Genome Sequencer Data Analysis Software Manual

Software Version 2.0.00, October 2008

Preface 15

About this Manual ... 15

Related Publications ... 17

Revision History ... 18

Intended Use ... 18

Notice to Purchaser ... 19

Trademarks ... 19

Safety ... 19

General Cautions ... 19

Assistance ... 20

Part A: Introduction

1. Overview of Genome Sequencer FLX System Data Processing 25

1.1 Data Processing and Data Analysis ... 26

1.2 Data Processing Options ... 30

1.3 Data Output and Folder Structure ... 33

1.3.1 Data Acquisition (GS Sequencer) Results: the Run Folder ... 34

1.3.2 Data Processing (GS Run Processor) Results: the Data Processing Folder ... 34

1.3.3 Data Analysis Applications Results ... 35

Part B: Data Processing

2. GS Sequencer (Output Only; Not a DataRig Application) 42

2.1 GS Sequencer Application Output 42
2.1.1 dataRunParams.parse 43
2.1.2 imageLog.parse 44
2.1.3 Raw Image Files 44

3. GS Run Processor Application 45

3.1 The Image Processing Step 46
3.1.1 Launching the Image Processing Step of Data Processing 46
3.1.2 Image Processing Step Output 48
3.1.2.1 *region*/dataRunParams.xml 49
3.1.2.2 *region*/regionnum.cwf 49
3.1.2.3 gsRunProcessor.log and gsRunProcessor_err.log 49
3.2 The Signal Processing Step 51
3.2.1 Introduction to Signal Processing 51
3.2.2 Quality Filtering and Trimming 52
3.2.2.1 Quality Filters 52
3.2.2.1.1 The Keypass Filter 52
3.2.2.1.2 The Dots Filter 52
3.2.2.1.3 The Mixed Filter 53
3.2.2.1.4 The Signal Intensity Filter 53
3.2.2.1.5 The Primer Filter 53
3.2.2.1.6 The TrimBack Valley Filter 54
3.2.2.1.7 The Quality Score Trimming Filter 55
3.2.2.2 Individual Base Quality Scores 56
3.2.2.3 Modifying the Stringency of Quality Filtering 56
3.2.3 Launching the Signal Processing Application 60
3.2.4 Signal Processing Step Output 62
3.2.4.1 *region*.cwf 64
3.2.4.2 *uaccnoRegion*.sff 64
3.2.4.3 gsRunProcessor.log and gsRunProcessor_err.log 64

4. GS Reporter Application 65

4.1 Launching the GS Reporter Application 65
4.2 GS Reporter Application Output Files 67
4.2.1 *region*.wells 70
4.2.2 *region*.trimInfo 70
4.2.3 *region.wells.key*.454RuntimeMetrics.txt 71
4.2.4 454QualityFilterMetrics.txt 71
4.2.5 454BaseCallerMetrics.txt 73
4.2.6 454RuntimeMetricsAll.txt 74
4.2.7 *region.key*.454Reads.fna 78
4.2.8 *region.key*.454Reads.qual 78

Part C: Data Analysis (Assembly and Mapping)

5. GS *De Novo* Assembler Application 86

5.1 Introduction to the GS *De Novo* Assembler ... 86

5.2 Using the GS *De Novo* Assembler Via the Graphical User Interface ... 87

5.2.1 Launching the Graphical User Interface Application ... 88
5.2.2 GS *De Novo* Assembler Interface Overview ... 89
5.2.2.1 The Main Buttons, Status Area, and Progress Box ... 90
5.2.2.2 The Quick Start and Documentation Text Links ... 90
5.2.3 Creating a New Assembly Project ... 91
5.2.4 Setting Assembly Project Parameters ... 92
5.2.5 Adding Read Data Sets to an Assembly Project ... 96
5.2.5.1 Adding GS Read Data Sets to an Assembly Project ... 97
5.2.5.2 Adding FASTA Read Data Sets to an Assembly Project ... 102
5.2.6 Computing an Assembly ... 103
5.2.6.1 Computing an Assembly for the First Time ... 103
5.2.6.2 Re-Computing an Assembly ... 103
5.2.7 Stopping an Assembly Project Computation in Progress ... 103
5.2.8 Viewing the Results of an Assembly Computation ... 103
5.2.8.1 Viewing the GS read and FASTA read metadata: the Project Tab ... 104
5.2.8.2 Viewing the Assembly Metrics Files: the Result Files Tab ... 105
5.2.8.3 Viewing the Alignment Data: the Alignment Results Tab ... 106
5.2.8.4 Viewing the Flowgram Data: the Flowgram Tab ... 108

5.3 Using the GS *De Novo* Assembler Via the Command Line Interface ... 112

5.3.1 One-Step Assembly: the runAssembly Command ... 112
5.3.2 Project-Based Assembly: the newAssembly and Related Commands ... 118
5.3.2.1 The newAssembly Command ... 118
5.3.2.2 The addRun Command ... 118
5.3.2.3 The removeRun Command ... 120
5.3.2.4 The runProject Command ... 121

5.4 GS *De Novo* Assembler Application Output ... 125

5.4.1 Project/Output Folder Structure ... 125
5.4.2 GS *De Novo* Assembler Output Files ... 126
5.4.2.1 454AllContigs.fna ... 127
5.4.2.2 454AllContigs.qual ... 127
5.4.2.3 454LargeContigs.fna ... 127
5.4.2.4 454LargeContigs.qual ... 127
5.4.2.5 454NewblerMetrics.txt ... 128
5.4.2.6 454NewblerProgress.txt ... 134
5.4.2.7 454AlignmentInfo.tsv ... 134
5.4.2.8 454ContigGraph.txt ... 135
5.4.2.9 454PairAlign.txt ... 136
5.4.2.10 454ReadStatus.txt ... 137
5.4.2.11 454TrimStatus.txt ... 137
5.4.2.12 454Contigs.ace or ace/ContigName.ace or consed/… ... 138
5.4.2.13 sff/sfffilename ... 140
5.4.2.14 454Scaffolds.fna ... 140
5.4.2.15 454Scaffolds.qual ... 140
5.4.2.16 454Scaffolds.txt ... 140
5.4.2.17 454PairStatus.txt ... 141
5.4.2.18 454TagPairAlign.txt ... 141

5.5 Assembling with Reads Obtained Using the Sanger Sequencing Method ... 142

5.5.1 Direct Incorporation into the Assembly Process ... 142
5.5.2 Sanger Reads with available Quality Scores ... 142
5.5.3 Sanger Read Annotations ... 143
5.5.3.1 The “template”, “dir”, and “ library” Annotations ... 143
5.5.3.2 The “trim” Annotation ... 144
5.5.3.3 The “scf” and “phd” Annotations ... 144
5.5.4 Recommended Method for including Sanger Reads ... 145

Part C: Data Analysis (Assembly and Mapping), continued

6. GS Reference Mapper Application 147

6.1 Introduction to the GS Reference Mapper 147

6.2 Using the GS Reference Mapper Via the Graphical User Interface 148

6.2.1 Launching the Graphical User Interface Application 148
6.2.2 GS Reference Mapper Interface Overview 149
6.2.2.1 The Main Buttons, Status Area, and Progress Box 149
6.2.2.2 The Quick Start and Documentation Text Links 150
6.2.3 Creating a New Mapper Project 150
6.2.4 Setting Reference Mapper Project Parameters 151
6.2.5 Adding Read Data to a Mapping Project 156
6.2.5.1 Adding GS Read Data Sets to a Mapping Project 157
6.2.5.2 Adding FASTA Read Data Sets to a Mapping Project 162
6.2.6 Adding Reference DNA Sequence Files to a Mapping Project 163
6.2.7 Computing the Read Alignments to the Reference 165
6.2.7.1 Computing the Alignments to the Reference for the First Time 165
6.2.7.2 Re-Computing the Alignments to the Reference 165
6.2.8. Stopping a Mapping Project Computation In Progress 165
6.2.9 Viewing the Results of a Mapping Computation 166
6.2.9.1 Viewing the Reference, GS read, and FASTA read metadata: the Project Tab 166
6.2.9.2 Viewing the Mapping Metrics Files: the Result Files Tab 168
6.2.9.3 Viewing the High Confidence Differences: the HCDiffs Tab 169
6.2.9.4 Viewing the Alignment Data: the Alignment Results Tab 171
6.2.9.5 Viewing the Flowgram Data: the Flowgrams Tab 173

6.3 Using the GS Reference Mapper Via the Command Line Interface 176

6.3.1 One-Step Mapping: the runMapping Command 176
6.3.2 Project-Based Mapping: the newMapping and Related Commands 184
6.3.2.1 The newMapping Command 184
6.3.2.2 The setRef Command 185
6.3.2.3 The addRun Command 186
6.3.2.4 The removeRun Command 188
6.3.2.5 The runProject Command 189

6.4 GS Reference Mapper Application Output 193

6.4.1 Project/Output Folder Structure 193
6.4.2 GS Reference Mapper Output Files 194
6.4.2.1 454AllContigs.fna 195
6.4.2.2 454AllContigs.qual 195
6.4.2.3 454LargeContigs.fna 195
6.4.2.4 454LargeContigs.qual 195
6.4.2.5 454AllDiffs.txt 196
6.4.2.6 454HCDiffs.txt 197
6.4.2.7 454NewblerMetrics.txt 198
6.4.2.8 454NewblerProgress.txt 204
6.4.2.9 454MappingQC.xls 205
6.4.2.10 454AlignmentInfo.tsv 208
6.4.2.11 454PairAlign.txt 209
6.4.2.12 454ReadStatus.txt 210
6.4.2.13 454RefStatus.txt 210
6.4.2.14 454TrimStatus.txt 211
6.4.2.15 454Contigs.ace or ace/refaccno.ace or consed/… 211
6.4.2.16 sff/sfffilename 212
6.4.2.17 454PairStatus.txt 212
6.4.2.18 454TagPairAlign.txt 212

Part C: Data Analysis (Assembly and Mapping), continued

7. SFF Tool Commands 213

7.1 **sfffile** 213
7.2 **sffinfo** 218
7.3 **sff2scf** 219
7.4 **fnafile** 221
7.5 **sffrescore** 224

8. GS Run Browser Application 225

8.1 **Introduction to the GS Run Browser** 225
8.1.1 Launching the GS Run Browser Application 226
8.1.2 GS Run Browser Interface Overview 228
8.1.2.1 The Main Buttons 229
8.1.2.2 The Tabs 229
8.1.2.3 Buttons and Plots 230
8.1.2.3.1 Scroll Bars 230
8.1.2.3.2 Navigation Buttons 230
8.1.2.3.3 Mousing Functions 232
8.2 **The Overview Tab** 233
8.2.1 General Description 233
8.2.2 Source of the Data 233
8.2.3 Features and Functionalities 234
8.3 **The Wells Tab** 235
8.3.1 General Description 235
8.3.2 Source of the Data 235
8.3.3 Features and Functionalities 236
8.3.4 Well Flowgrams (for Wells Generating Library Reads) 241
8.3.4.1 General Description 241
8.3.4.2 Features and Functionalities 242
8.3.5 Well Tri-Flowgrams (for Wells Generating Control DNA Reads) 243
8.3.5.1 General Description 243
8.3.5.2 Features and Functionalities 244
8.3.6 Location Flowgrams 245
8.3.6.1 General Description 245
8.3.6.2 Features and Functionalities 245
8.3.7 Subtraction Raw Flowgrams 246
8.3.7.1 General Description 246
8.3.7.2 Features and Functionalities 246
8.4 **The Signals Tab** 247
8.4.1 General Description 247
8.4.2 Source of the Data 248
8.4.3 Features and Functionalities 248
8.5 **The Reads Tab** 250
8.5.1 General Description 250
8.5.2 Source of the Data 250
8.5.3 Features and Functionalities 251

Part C: Data Analysis (Assembly and Mapping), continued

8.6 The Control DNA Tab 252

8.6.1 General Description 252

8.6.2 Source of the Data 252

8.6.3 Features and Functionalities 253

8.6.4 Control DNA Consensus Flowgrams 254

8.6.4.1 General Description 254

8.6.4.2 Features and Functionalities 255

8.7 The Filters Tab 256

8.7.1 General Description 256

8.7.2 Source of the Data 256

8.7.3 Features and Functionalities 257

Part D: Data Analysis (Amplicon Variant Analysis)

9. GS Amplicon Variant Analyzer Application 265

9.1 Introduction to the GS Amplicon Variant Analyzer Application 266
9.1.1 Definitions 266
9.1.1.1 Project 266
9.1.1.2 Reference Sequence 266
9.1.1.3 Amplicon and Target 267
9.1.1.4 Read Data Set and Read Group 267
9.1.1.5 Variant 268
9.1.1.6 Sample 268
9.1.1.7 MID and MID Group 269
9.1.1.8 Multiplexer 270
9.1.2 Launching the GS Amplicon Variant Analyzer Application 271
9.1.3 GS Amplicon Variant Analyzer Application Interface Overview 272
9.1.3.1 Main Buttons 272
9.1.3.2 The Tabs and Sub-Tabs 274
9.1.3.3 Buttons and Plots 276
9.1.3.3.1 Scroll Bars 277
9.1.3.3.2 Navigation Buttons 277
9.1.3.3.3 Mousing Functions 278
9.1.3.3.4 Progress bars 279
9.1.3.3.5 Special Action Buttons 279
9.1.3.4 File Browsing in Linux 279
9.2 The Overview Tab 280
9.3 The Project Tab 281
9.3.1 The “Project Tree” Sub-Tabs 283
9.3.1.1 The References Tree 286
9.3.1.2 The Read Data Tree 287
9.3.1.3 The Samples Tree 289
9.3.1.4 The MIDs Tree 290
9.3.2 The “Definition Table” Sub-Tabs 291
9.3.2.1 The References Definition Table 296
9.3.2.1.1 To Enter or Edit the DNA Sequence of a Reference Sequence 297
9.3.2.2 The Amplicons Definition Table 297
9.3.2.2.1 To Enter or Edit the Reference Sequence to which an Amplicon is associated 298
9.3.2.2.2 To Enter or Edit the Primer Sequences for the Amplicon 299
9.3.2.2.3 To Enter or Edit the Target Start and End Positions 300
9.3.2.3 The Read Data Definition Table 302
9.3.2.3.1 To Edit the Read Group of a Read Data Set 303
9.3.2.3.2 To Edit the “Active” status of a Read Data Set 303
9.3.2.4 The Samples Definition Table 304
9.3.2.5 The Variants Definition Table 305
9.3.2.5.1 To Enter or Edit the Reference Sequence to which a Variant is associated 306
9.3.2.5.2 To Enter or Edit the Pattern of a (Known) Variant 306
9.3.2.5.3 To Edit the Status of a Variant 309
9.3.2.6 The MIDs Definition Table 310
9.3.2.6.1 To Enter or Edit the DNA Sequence of an MID 312
9.3.2.6.2 To Edit the MID Group of an MID 313
9.3.2.7 The Multiplexers Definition Table 314
9.3.2.7.1 To Enter or Edit the Sample Encoding using Multiplexers 315
9.3.2.7.1.1 “Primer 1 MID” and “Primer 2 MID” Encoding 316
9.3.2.7.1.2 “Both” Encoding 316
9.3.2.7.1.3 “Either” Encoding 317
9.3.2.7.2 To Enter or Edit the Primer 1 MIDs and Primer 2 MIDs 317
9.3.2.7.3 To Enter or Edit the Samples Assignment 319
9.3.2.7.3.1 Sample Assignment with “Primer 1 MID” or “Primer 2 MID” Encoding 320
9.3.2.7.3.2 Sample Assignment with “Both” Encoding 321
9.3.2.7.3.3 Sample Assignment with “Either” Encoding 322
9.3.2.7.4 Using Multiplexers for more than one Read Data 325
9.4 The Computations Tab 326

Part D: Data Analysis (Amplicon Variant Analysis), continued

9.5 **The Variants Tab** 330
9.5.1 The Variants Frequency Table 331
9.5.1.1 General organization 331
9.5.1.2 Organizing Data in the Variants Frequency Table 333
9.5.1.2.1 Sort options 334
9.5.1.2.2 Ignore filters 335
9.5.1.2.3 Show filters 335
9.5.1.2.4 Option reversions 335
9.5.1.3 Populating the Global Align Tab from the Variants Tab 336
9.5.1.4 Defining a Haplotype from the Variants Tab 336
9.5.1.5 Editing/Removing Variants from the Variants Tab 338
9.5.1.6 The Mouse Tracker 338
9.5.2 Variant Data Display Controls 339
9.5.2.1 The Alignment Read Type Controls 339
9.5.2.2 The Show Values Controls 340
9.5.2.3 The Min / Max Filters 341
9.5.2.4 The Variant Status Filter 342
9.5.2.5 The Compact Table Checkbox 342
9.5.2.6 The Auto-Detected Variant Load Button 342
9.5.2.7 Variant Discovery Workflow 344
9.6 **The Global Align Tab** 346
9.6.1 Populating the Global Align Tab 347
9.6.2 The Variation Frequency Plot 348
9.6.3 The Multiple Alignment Display 349
9.6.3.1 The Reference Sequence 349
9.6.3.2 The Multi-Alignment 350
9.6.3.3 Special Function Buttons 352
9.6.4 Display Option Tools 354
9.6.4.1 Alignment Data 355
9.6.4.2 Read Type 356
9.6.4.3 Reported Frequency 356
9.6.4.4 Read Orientation 357
9.7 **The Consensus Align Tab** 358
9.7.1 Populating the Consensus Align Tab 358
9.7.2 The Variation Frequency Plot 359
9.7.3 The Multiple Alignment Display 359
9.7.4 Display Option Tools 359
9.8 **The Flowgrams Tab** 360
9.8.1 Populating the Flowgrams Tab 362
9.8.2 The Tri-Flowgram Plot 362
9.8.3 Navigation on the Flowgrams Tab 363

10. **Example Amplicon Project Design and Analysis** **364**

10.1 **Experimental Design** 365
10.2 **Project Setup in the AVA Software** 368
10.2.1 Launching the AVA Application 369
10.2.2 Creating a New Project 370
10.2.3 Defining the Reference Sequence 372
10.2.4 Defining the Amplicons 374
10.2.5 Defining the Sample 376
10.2.6 Defining the Known Variant 377
10.2.7 Importing the Read Data Set 379
10.3 **Analysis of Known Variants** 381
10.3.1 Compute the Project 381
10.3.2 Frequency of Known Variants 382
10.4 **Mining a Project for New Variants** 389

Part D: Data Analysis (Amplicon Variant Analysis), continued

10.5 **Important Factors in the Assessment of New Variants** 404
10.5.1 Above the Noise 404
10.5.2 Coverage 404
10.5.3 Bidirectional Support 404
10.5.4 Homopolymers 404
10.5.5 Flowgram Evidence 405
10.5.6 Read Length 405
10.6 **Other Issues of Special Interest** 406
10.6.1 What Does 'Sample' Mean? 406
10.6.2 How Should Your Project Be Organized? 407
10.6.3 Should Amplicons Share a Reference Sequence or Have Individual Ones? 407
10.6.4 When should MIDs be used? 408
10.6.5 What is the purpose of Multiplexers? 408
10.6.5.1 Non-Multiplexer Example 409
10.6.5.2 Multiplexer Example 410
10.6.5.3 Multiplexer Benefits Summary 412

11. **GS Amplicon Variant Analyzer Command Line Interface** **413**

11.1 **Purpose of the CLI** 413
11.1.1 Data Import 413
11.1.2 Data Export 414
11.1.3 Automating the Triggering of Computations 414
11.1.4 Result Reporting 414
11.2 **AVA-CLI Command Language Overview** 415
11.2.1 Entities 415
11.2.2 Available Commands 415
11.3 **AVA-CLI General Online Help** 417
11.3.1 Help 417
11.3.2 General Help 417
11.3.2.1 CommandLine Help 418
11.3.2.2 Parsing Help 419
11.3.2.3 Tabular Commands Help 420
11.3.2.4 Record Names Help 422
11.3.2.5 Abbreviations Help 422
11.3.2.6 File Paths Help 423
11.3.2.7 Multiplexing Help 424
11.4 **AVA-CLI Command Usage Statements** 425
11.4.1 associate 425
11.4.2 close 428
11.4.3 computation 429
11.4.3.1 computation start 429
11.4.3.2 computation stop 429
11.4.3.3 computation status 429
11.4.3.4 computation loadDetectedVariants 429
11.4.4 create 430
11.4.4.1 create amplicon 430
11.4.4.2 create mid 431
11.4.4.3 create midGroup 432
11.4.4.4 create multiplexer 432
11.4.4.5 create project 433
11.4.4.6 create readGroup 433
11.4.4.7 create reference 434
11.4.4.8 create sample 434
11.4.4.9 create variant 435
11.4.5 dissociate 436
11.4.6 exit 438

Part D: Data Analysis (Amplicon Variant Analysis), continued

11.4.7 list 438
11.4.7.1 list amplicon 439
11.4.7.2 list mid 439
11.4.7.3 list midGroup 439
11.4.7.4 list multiplexer 440
11.4.7.5 list project 440
11.4.7.6 list readData 440
11.4.7.7 list readGroup 441
11.4.7.8 list reference 441
11.4.7.9 list sample 441
11.4.7.10 list variant 442
11.4.8 load 442
11.4.9 open 443
11.4.10 remove 444
11.4.10.1 remove amplicon 444
11.4.10.2 remove mid 444
11.4.10.3 remove midGroup 445
11.4.10.4 remove multiplexer 445
11.4.10.5 remove readData 445
11.4.10.6 remove readGroup 445
11.4.10.7 remove reference 446
11.4.10.8 remove sample 446
11.4.10.9 remove variant 446
11.4.11 rename 447
11.4.11.1 rename amplicon 447
11.4.11.2 rename mid 447
11.4.11.3 rename midGroup 448
11.4.11.4 rename multiplexer 448
11.4.11.5 rename project 448
11.4.11.6 rename readData 448
11.4.11.7 rename readGroup 448
11.4.11.8 rename reference 449
11.4.11.9 rename sample 449
11.4.11.10 rename variant 449
11.4.12 report 450
11.4.12.1 report variantHits 450
11.4.13 save 450
11.4.14 set 451
11.4.14.1 set verbose 451
11.4.14.2 set onErrors 451
11.4.14.3 set currDir 451
11.4.15 Show 452
11.4.15.1 show environment 452
11.4.16 update 452
11.4.16.1 update amplicon 453
11.4.16.2 update mid 454
11.4.16.3 update midGroup 455
11.4.16.4 update multiplexer 455
11.4.16.5 update project 456
11.4.16.6 update readData 456
11.4.16.7 update readGroup 456
11.4.16.8 update reference 457
11.4.16.9 update sample 457
11.4.16.10 update variant 458
11.4.17 utility 458
11.4.17.1 utility validateNames 459
11.4.17.2 utility validateForComputation 459
11.4.17.3 utility makeSetupScript 460
11.4.17.4 utility clone 460
11.4.17.5 utility execute 461

Part D: Data Analysis (Amplicon Variant Analysis), continued

11.5 Creating and Computing a Project with the AVA-CLI 462
11.5.1 Setting CLI Parameters 462
11.5.2 Creating a New Project 463
11.5.3 Creating References 463
11.5.4 Creating Amplicons 466
11.5.5 Creating Variants 468
11.5.6 Creating Samples 469
11.5.7 Associating Samples with Amplicons 469
11.5.8 Loading Read Data Sets 471
11.5.9 Associating Read Data Sets with Samples 472
11.5.10 Editing Object Properties 473
11.5.10.1 Updating an Object 473
11.5.10.2 Renaming an Object 474
11.5.10.3 Removing an Object 474
11.5.10.4 Dissociating Relationships 475
11.5.11 Computation 476
11.5.11.1 Validating the Project before Computation 476
11.5.11.2 Managing the Computation 477
11.5.11.3 Loading Automatically Detected Variants 478
11.5.12 Reporting 478
11.5.13 Finishing Touches 479
11.5.13.1 save 479
11.5.13.2 close 479
11.5.13.3 exit 479
11.5.14 Exporting from the Project 480
11.5.14.1 utility makeSetupScript 480
11.5.14.2 utility clone 480
11.5.14.3 list 481
11.5.15 Integrated Project Script 481
11.6 Creating and Computing an MID Project with the AVA-CLI 485
11.6.1 Example MID Project Script 487

12. Amplicon Variant Analyzer – Special Topics 490

12.1 Addressing Simultaneous Multiple Users' Access to an Amplicon Project 490
12.2 Intelligent Variant Naming 493
12.2.1 Tier 1 Naming 493
12.2.2 Tier 2 Naming 494
12.2.3 Tier 3 Naming 494
12.2.4 Tier 4 Naming 495
12.2.5 Naming Example 495
12.3 Properties Windows for Global and Consensus Alignments 496
12.3.1 When is the Properties Information Useful? 496
12.3.2 Content of the Three "Properties" Window Types 496
12.3.2.1 Properties Window for a Consensus 497
12.3.2.2 Properties Window for a Forward Read 497
12.3.2.3 Properties Window for a Reverse Read 498
12.4 Automatic Project Initialization in the GUI 499
12.4.1 Default Initialization Script Location 499
12.4.2 Default Initialization Script Contents 500
12.4.2.1 Step 1: Loading the "Standard" 454 MIDs 500
12.4.2.2 Step 2: Running User-Customized Initialization Functions 500
12.4.3 Initialization Script Restrictions 501
12.4.4 Initialization Script Error Handling 501
12.5 Project Initialization and the CLI 502
12.6 Multiplex Amplicon Libraries 503

Part E: Appendix

13. Data Files and Formats 508

13.1 Directory Naming Conventions 509
13.2 Format Requirements for Input FASTA Files 509
13.3 Standard File Formats 510
13.3.1 Composite wells file format 510
13.3.1.1 mimetype 512
13.3.1.2 meta.xml 512
13.3.1.3 history.xml 513
13.3.1.4 location.idx 515
13.3.1.5 metrics.xml 516
13.3.1.6 sequences.xml 518
13.3.1.7 filters.xml 519
13.3.1.8 filterResults.uint8.dat 520
13.3.1.9 Base Called Data 522
13.3.1.10 Other Public Streams 522
13.3.1.11 Other Private Streams 523
13.3.1.12 Other Files 523
13.3.2 Parameters and Viewable Metrics Files: 454 Parser Format (.parse, .txt) 524
13.3.3 Image Files (.pif) 524
13.3.4 Well-Level Signal Data Files (.wells) 525
13.3.5 Exportable Metrics Files (.csv) 525
13.3.6 DNA Sequence (FASTA; .fna) and Base Quality Score (.qual) Files 526
13.3.7 454 “Universal” Accession Numbers 527
13.3.8 Standard Flowgram Files (.sff) 528
13.3.8.1 Common Header Section 529
13.3.8.2 Read Header Section 530
13.3.8.3 Read Data Section 531
13.3.8.4 Computing Lengths and Scanning the File 531
13.4 Example Files 532
13.4.1 454QualityFilterMetrics.txt 532
13.4.2 454BaseCallerMetrics.txt 534
13.4.3 454RuntimeMetricsAll.txt 545
13.4.4 *region.key*.454Reads.fna and *region.key*.454Reads.qual 552
13.4.5 Consensus Basecalled Contig Files 554
13.4.6 454NewblerMetrics.txt File for the GS *De Novo* Assembler Application 557
13.4.7 454ReadStatus.txt 562
13.4.8 454AllDiffs.txt 564
13.4.9 454NewblerMetrics.txt File for the GS Reference Mapper Application 567
13.4.10 454NewblerProgress.txt 571
13.4.11 454AlignmentInfo.tsv 572
13.4.12 454ReadStatus.txt 572
13.4.13 454RefStatus.txt 573
13.4.14 sffinfo summary 573
13.4.15 MID Configuration File 575

Preface

About this Manual

The *Genome Sequencer Data Analysis Software Manual* describes the software package version. 2.0.00 of the Genome Sequencer FLX System for DNA Sequencing, developed by 454 Life Sciences Corporation. This software is used to process the data acquired during one or more sequencing Runs on a Genome Sequencer FLX Instrument. Such data processing is divided into two main categories: data processing, whereby raw image data is converted into processed signals and basecalled reads; and data analysis whereby the final results of the sequencing experiment are produced, such as the generation of a consensus sequence for the DNA sample.

Each step in the acquisition, processing and analysis of sequencing data in the Genome Sequencer FLX System is governed by a specific application. The GS Sequencer governs the sequencing Run itself, whereby the raw images are acquired. The raw data are processed by the GS Run Processor, itself composed of two parts: image processing and signal processing. Acquisition and data processing are always carried out in a sequencing experiment. A separate application associated with data processing, the GS Reporter can extract data and metrics files, and present them in human-readable form to allow the examination of the results of a sequencing Run.

Three data analysis applications can be chosen, depending on your experimental set up and goals: the GS *De Novo* Assembler (with or without contig scaffolding using Paired End reads), the GS Reference Mapper, and the GS Amplicon Variant Analyzer (AVA). An additional application, the GS Run Browser, is an interactive Run browser/ troubleshooting tool which displays graphically the images, some intermediate data, and various output metrics from a sequencing Run. The software package also includes the SFF Tools commands for handling and using the data files (called Standard Flowgram Format or SFF files) that hold the sequencing trace data.

Following an overview of data processing and analysis in the Genome Sequencer FLX System, this manual provides a full description of these applications and commands, including how they are invoked through their Graphical User Interface (GUI) and at the UNIX command line level on a DataRig or a computer cluster resource, and information on the format of the output files of all the applications. In addition, a pair of DVDs is provided with this manual, which contains sample Runs that serve as examples of the Genome Sequencer FLX System output; these Runs can, in part, be viewed through the GS Run Browser or the AVA software.

For applications used on the Genome Sequencer FLX Instrument itself, see the *Genome Sequencer FLX Operator's Manual*. For details on the GS Sequencer application and how to use it to carry out a sequencing Run, see the *GS FLX Titanium Sequencing Method Manual*.

Important Note: October 2008 marks the first release of the new GS FLX Titanium series chemistry for the Genome Sequencer FLX System. For the time being, the system supports only the non-MID "General" (*e.g.* Shotgun) sequencing applications under the GS FLX Titanium chemistry. For Paired End or Amplicon sequencing, or for the preparation and sequencing of MID libraries of any type, users must continue to use the GS FLX standard series kits and procedures (last updated in December 2007).

However, the Genome Sequencer FLX Software version 2.0.00, associated with the October 2008 release, is fully backward-compatible with, and can process datasets generated with any of the Genome Sequencer System's chemistries (GS 20, GS FLX standard, and GS FLX Titanium). This manual describes **all** the functionalities of the 2.0.00 software, even those that apply **only** to datasets generated with older chemistries, such that **it completely replaces** the December 2007 issue of the *Genome Sequencer FLX Data Analysis Software Manual* (which addressed the software version 1.1.03).

The Genome Sequencer FLX Titanium series manuals are easily identified by their new cover graphics and distinctive tri-color stripes (reflecting the GS FLX Titanium kits packaging). All the methods, protocols and applications supported on the GS FLX standard chemistry will be enabled on the GS FLX Titanium chemistry in the near future.

Features supported only under the GS FLX standard chemistry:

- While the use of Paired End and of MID-tagged Shotgun (sstDNA) libraries is supported under the GS FLX standard chemistry, these are not currently supported under the GS FLX Titanium chemistry. Therefore, the information relative to the analysis of datasets from Paired End or from MID-tagged libraries that appears in Parts A and C of this manual applies only to libraries prepared under the GS FLX standard chemistry.
- While the Amplicon Variant Analyzer software v. 2.0.00 provides important new features, such as the support for MIDs with Amplicon libraries, those libraries are not currently supported under the GS FLX Titanium chemistry. Therefore, the entire content of Part D of this manual, including the new features, applies only to Amplicon libraries prepared under the GS FLX standard chemistry.

In this manual, the phrase "Genome Sequencer System" refers to whole system for DNA sequencing developed by 454 Life Sciences Corp., including the Genome Sequencer Instrument, all the kits for the preparation, amplification and sequencing of a DNA sample, the methods to use the kits as described in the Manuals and Guides, and the software provided to process and analyze the data from sequencing Runs. Likewise, "Genome Sequencer FLX System" refers to a Genome Sequencer System based on the Genome Sequencer FLX Instrument (as opposed to the Genome Sequencer 20 Instrument, which is now retired). Two versions of the Genome Sequencer FLX System have been released: the GS FLX standard series, last updated in December 2007, and the GS FLX Titanium series. 454 Life Sciences Corporation is a Roche company.

Related Publications

A full suite of publications are available that describe in detail the components and usage of the Genome Sequencer System:

- *Genome Sequencer FLX Operator's Manual (October 2008)*
- *Genome Sequencer FLX Titanium Applications and Methods Manual*, including:
    - *GS FLX Titanium General Library Preparation Method Manual*
    - *GS FLX Titanium emPCR Method Manual*
    - *GS FLX Titanium Sequencing Method Manual*
    - A Quick Guide version of each method is also included.
- *Genome Sequencer Data Analysis Software Manual* – this manual
- *Genome Sequencer System Site Preparation Guide (October 2008)*

Note also that some of the applications of the Genome Sequencer FLX standard System (December 2007) are not yet available for the GS FLX Titanium chemistry. These include the preparation and usage of Paired End and Amplicon libraries, and the usage of Multiplex Identifiers (MIDs). For these applications, the Genome Sequencer FLX standard System methods and kits must still be used. The December 2007 GS FLX manual set comprises the following:

- *Genome Sequencer FLX Operator's Manual (December 2007)*
- *Genome Sequencer FLX System Methods Manual*, including:
    - *GS FLX Shotgun DNA Library Preparation Method Manual*
    - *GS FLX Paired End DNA Library Preparation Method Manual*
    - *GS FLX Amplicon DNA Library Preparation Method Manual*
    - *GS FLX emPCR Method Manual*
    - *GS FLX Sequencing Method Manual*
    - A Quick Guide version of each method is also available.
- *Genome Sequencer FLX Data Analysis Software Manual*
- *Genome Sequencer System Site Preparation Guide (December 2007)*

For users who want to input sample information and Run settings to the Genome Sequencer FLX Instrument via a Laboratory Information Management System (LIMS), a document entitled "*GS LIMS Implementation Guide*" is available from your Roche Representative. This Guide explains how to implement the LIMS lookup feature of the Genome Sequencer System.

All Genome Sequencer Manuals, Guides and Bulletins are available three ways: in hardcopy form, on a CD from your Roche representative, or downloaded from the customer restricted access area of **www.genome-sequencing.com**.

Revision History

Manual Version	Instrument Version	Software ersion	Revision Date
FLX.01 – USM-00027.A	GS FLX	1.1.01	December 2006
FLX.02 – USM-00027.B	GS FLX	1.1.02	June 2007
FLX.03 – USM-00036.A	GS FLX	1.1.03	December 2007
FLX.Ti.00 – USM-00058.A	GS FLX	2.0.00	October 2008

454 Life Sciences Corporation
1 Commercial St.
Branford, CT
USA 06405

Intended Use

The Genome Sequencer FLX System data processing and data analysis software is intended for use on a Genome Sequencer FLX Instrument or on a DataRig or a computer cluster configured to be used as part of the Genome Sequencer FLX System. The data processing software is used to acquire the DNA sequencing data, process it, and output it in the form specified by the user. The data analysis software then transforms the processed data into the desired final output of the Genome Sequencer FLX System.

Notice to Purchaser

RESTRICTION ON USE: Purchaser is only authorized to use the Genome Sequencer Instrument with PicoTiterPlate devices supplied by 454 Life Sciences Corporation and in conformity with the operating procedures contained in the Genome Sequencer System manuals and guides.

Made in USA by 454 Life Sciences Corporation, Branford, CT, USA, a Roche company.

For life science research only. Not for use in diagnostic procedures.

Trademarks

Safety

General Cautions

Connection to computer networks contains an inherent risk of infection by viruses and worms and of malicious targeted attacks through the network. It is the customer's responsibility to protect the system against such threats, *i.e.* by keeping up-to-date the protection of any network to which the customer chooses to connect the Genome Sequencer FLX Instrument or any "DataRig" or computer cluster. This protection might include measures such as a firewall to separate these devices from uncontrolled networks, as well as measures to ensure that the connected network is free of malicious code.

Assistance

If you have questions or experience problems with the Genome Sequencer System, please call, write, fax, or email us.

When calling for assistance, be prepared to provide the serial number of your Genome Sequencer Instrument and/or lot number of the kit(s) you are using. The instrument's serial number is located on the label found on the back of the instrument cart.

If you are located in…	**Please contact Roche Applied Science Technical Support via:**	
USA or Canada	phone: 1-800-262-4911 (toll-free)	e-mail: **us.gssupport@roche.com**
Europe, Middle East, Asia Pacific, Mexico, South America or Africa	phone: +49-8856-60-6457 or toll-free +800SEQUENCE	e-mail: **service.sequencing@roche.com**
Japan	phone: +03-5443-5287	e-mail: **tokyo.biochemicals@roche.com**

Introduction

A

Part A: Introduction

1. Overview of Genome Sequencer FLX System Data Processing 25

1.1 Data Processing and Data Analysis 26

1.2 Data Processing Options 30

1.3 Data Output and Folder Structure 33

1.3.1 Data Acquisition (GS Sequencer) Results: the Run Folder 34

1.3.2 Data Processing (GS Run Processor) Results: the Data Processing Folder 34

1.3.3 Data Analysis Applications Results 35

Important Note: October 2008 marks the first release of the new GS FLX Titanium series chemistry for the Genome Sequencer FLX System. For the time being, the system supports only the non-MID "General" (*e.g.* Shotgun) sequencing applications under the GS FLX Titanium chemistry. For Paired End or Amplicon sequencing, or for the preparation and sequencing of MID libraries of any type, users must continue to use the standard Genome Sequencer FLX kits and procedures (last updated in December 2007).

However, the Genome Sequencer FLX Software version 2.0.00, associated with the October 2008 release, is fully backward-compatible with, and can process datasets generated under any of the Genome Sequencer System's chemistries (GS 20 chemistry, GS FLX standard chemistry, and GS FLX Titanium chemistry). This manual describes **all** the functionalities of the 2.0.00 software, even those that apply **only** to datasets generated with older chemistries, such that **it completely replaces** the December 2007 issue of the *Genome Sequencer FLX Data Analysis Software Manual* (which addressed the software version 1.1.03).

The Genome Sequencer FLX Titanium series manuals are easily identified by their new cover graphics and distinctive tri-color stripes (reflecting the GS FLX Titanium kits packaging). All the methods, protocols and applications supported on the GS FLX standard chemistry will be enabled on the GS FLX Titanium chemistry in the near future.

Features supported only under the GS FLX standard chemistry:
While the use of Paired End and of MID-tagged Shotgun (sstDNA) libraries is supported under the GS FLX standard chemistry, these are not currently supported under the GS FLX Titanium chemistry. Therefore, the information relative to the analysis of datasets from Paired End or from MID-tagged libraries that appears in Part A of this manual applies only to libraries prepared under the GS FLX standard chemistry.

1. Overview of Genome Sequencer FLX System Data Processing

The Genome Sequencer FLX System, developed by 454 Life Sciences Corporation, a Roche company, is an ultra-high-throughput automated DNA sequencing system capable of carrying out and monitoring sequencing reactions in a massively parallel fashion (hundreds of thousands of simultaneous reactions, in the "wells" of a PicoTiterPlate device). During DNA-directed DNA synthesis, pyrophosphate (PPi) is released with each nucleotide addition; the instrument generates an amount of light commensurate with the amount of PPi released; this light is captured by a charge-coupled device camera and converted into a digital signal. (For more information on the basics of the Genome Sequencer FLX System, please refer to the *Genome Sequencer FLX Operator's Manual*).

The raw data resulting from a sequencing Run consists of a series of digital images captured by the camera. The images are a representation of the surface of the PicoTiterPlate device over which the sequencing reactions are taking place; and each image corresponds to one reagent flow over that surface, as defined by the Run script. If the sample DNA fragment present in a given PicoTiterPlate well is extended during a nucleotide flow, light is emitted from the well and captured on the image corresponding to that flow. Furthermore, the amount of light emitted is proportional to the number of nucleotides extended. Knowledge of the nucleotide flowed while each image is being captured (from the Run script), of the location on the PicoTiterPlate device where light is being emitted (coordinates of each pixel on the images), and of the amount of light emitted during each flow (brightness of the pixels in the corresponding images) allows the software to identify PicoTiterPlate wells that contain a DNA library fragment and determine the sequence of the DNA fragments present in each well.

1.1 Data Processing and Data Analysis

After a sequencing Run has completed on the Genome Sequencer FLX Instrument, the manipulation of the raw data (the images captured by the GS Sequencer application) consists of two main phases: "data processing" and "data analysis". Data processing is carried out by the GS Run Processor application and encompasses all the steps required to go from raw image data to base-called results suitable for use by downstream "data analysis" applications. Data processing is done in two main steps, image processing and signal processing. The signal processing step, in turn, exists in three "options": standard for general (*e.g.* Shotgun) sequencing, a set of options for Paired End sequencing, and an alternative set of algorithms especially tuned to process data from Amplicon sequencing Runs (or test Runs performed with only Control DNA Beads.).

The data analysis phase offers a choice of several downstream analysis paths to generate the desired final output: a consensus sequence of the DNA sample generated by the assembly of reads into contigs, with or without Paired End analysis to order and orient the contigs into scaffolds (GS *De Novo* Assembler); a consensus sequence along with a list of high-confidence mutations obtained by mapping the reads to a known reference sequence (GS Reference Mapper); or the identification and quantitation of sequence variants by the Ultra Deep Sequencing of amplicons (GS Amplicon Variant Analyzer). All data analysis outputs also include base-per-base quality scores (Phred-equivalent) and other specific metric files.

Table 1–1 lists the inputs and outputs of the three main components of data handling, from data acquisition through data processing, as well as the individual functions carried out by each application. It is important to note that the two steps of data processing are typically executed independently, running the image processing step on-board the instrument concurrently with the sequencing Run, and the more time consuming signal processing step on a dedicated data processing computer (cluster) (see section 1.2 for a description of data processing pipeline options). In general,

1. the GS Sequencer application records a set of raw digital images representing the light detected over the surface of the PicoTiterPlate device, during each reagent flow of the sequencing Run (this application is not described in the present manual, except for file output information; see the *Genome Sequencer FLX Operator's Manual* and the *GS FLX Titanium Sequencing Method Manual* for details);
2. the first step of the GS Run Processor application, image processing, performs initial pixel-level calculations, and then groups pixels from the image set into a representation of the PicoTiterPlate wells where sequencing reactions were detected; and
3. the second step of the GS Run Processor application, signal processing, performs well-level calculations across the whole series of images to generate well "flowgrams" (and the basecalls of the DNA fragments being sequenced in all the active PicoTiterPlate wells; "reads").

Application	Input	Output	Main processing steps
GS Sequencer	Light	Raw images	▶ Image acquisition and storage to disk
GS Run Processor (image processing step)	Raw images	Composite Wells Format (CWF) Files	▶ Subtract background and normalize the images (at the pixel level) ▶ Find the active PicoTiterPlate wells ▶ Extract the raw signals for each flow, in each active well ▶ Write the resulting flow signals into "composite wells format" (CWF) files
GS Run Processor (signal processing step)	Composite Wells Format (CWF) Files	Corrected CWF files, and SFF files containing read basecalls and per-base quality scores	▶ Filter out lower signal ghost wells (Amplicon sequencing pipeline only) ▶ Correct for crosstalk between neighboring wells ▶ Correct for known "out-of phase" errors (incomplete extension and carry-forward) ▶ Correct for signal droop and perform residual background subtraction ▶ Filter out any residual ghost wells (Amplicon sequencing pipeline only) ▶ Filter (pass or fail) the processed reads based on signal quality ▶ Trim read ends for low quality and primer sequence ▶ Update the CWF files with the fully processed data ▶ Generate Standard Flowgram Format (SFF) files containing the basecalled read sequences and per-base quality scores

Table 1–1: The 3 main components of data handling, from data acquisition through data processing, in the Genome Sequencer FLX System, with their inputs, outputs, and main processing steps. They are performed in succession, in the order indicated; the SFF files output by the signal processing step of the GS Run Processor application are used as input to either of the data analysis applications (see Table 1–2). For a description of the data processing pipeline options, see section 1.2.

The data analysis applications use the fully processed and "trimmed" read basecalls of a sequencing Run, or of a pool of Runs, to produce initial alignments to the reference sequence (or read-to-read overlaps for the GS *De Novo* Assembler); then they use a combination of nucleotide and flowgram information for consensus-calling of the contigs and determination of quality values for the contig sequences. Table 1–2 lists the specific outputs of the 3 data analysis applications as well as the individual functions carried out by each one. The final system output choices are the following:

1. The GS *De Novo* Assembler application generates a consensus sequence of the whole DNA sample, by assembling the reads into contigs (*de novo* shotgun assembly). An option allows the use of one or more sequencing Runs performed on a Paired End library (any type, or even a combination of Paired End library types) prepared from the same DNA sample, to be analyzed together with Shotgun sequencing Run(s) and help order and orient the resulting contigs into scaffolds. (Long Paired End reads do not necessarily need to be analyzed together with Shotgun reads.)
2. The GS Reference Mapper application generates the consensus DNA sequence by mapping, or alignment, of the reads to a reference sequence; as well as a list of high-confidence mutations (individual bases or blocks of bases that differ between the consensus DNA sequence of the sample and the reference sequence).
3. The GS Amplicon Variant Analyzer application compares reads from an Amplicon library to corresponding reference sequences, and allows the user to detect, identify and quantitate the prevalence of sequence variants.

It may take multiple Runs to generate enough data for a given sequencing project (*e.g.* a project requiring several-fold depth sequencing of a large genome). In such cases, the data sets of all the Runs can be combined at the time of data analysis. Furthermore, contig consensus-calling in mapping and assembly are carried out in "flowspace" (*i.e.* they operate directly on the processed signals measured from the wells), followed by basecalling to produce a consensus sequence for the sample.

The final output of the Genome Sequencer FLX System thus varies depending on whether Assembly, Mapping or Amplicon Variant Analysis analysis (or no analysis of any kind) is performed. In all cases, however, the output DNA sequence is supplied as a set of FASTA files, with associated "Quality Scores" and other Run and data metrics files useful for library titrations, troubleshooting, and determining the overall quality of the sequencing Run. ACE-formatted files are also produced by each of the data analysis applications to allow users to view alignment results using third-party software tools.

The software package described in this manual also includes the GS Reporter and the GS Run Browser applications, available both on the Genome Sequencer FLX Instrument and for off-instrument use (on a cluster or DataRig) and used to view and troubleshoot the results of a completed sequencing Run; and the SFFTools, a set of commands used to create, manipulate and access sequencing trace data from SFF files. However, these applications and commands are not required steps of data processing and analysis.

Application	Input	Output	Main processing steps
GS *De Novo* Assembler	SFF files, from one or multiple sequencing Runs, containing read flowgrams and basecalls, and per-base quality scores	Sample consensus sequence, assembled *de novo* (and scaffold information, with Paired End option)	▶ Identify pairwise overlaps between reads, in nucleotide space ▶ Construct multiple alignments of reads that tile together (*i.e.* form contigs), based on the pairwise overlaps ▶ Generate consensus basecalls of the contigs by averaging the processed flow signals for each nucleotide flow included in the alignment, in flowspace ▶ Output the contig consensus sequences and corresponding quality scores, along with an ACE file of the multiple alignments and assembly metrics files Additional steps with Paired End option: ▶ Identify pairwise overlaps between Paired End tags and the shotgun contigs ▶ Organize the contigs into scaffolds (order, orientation, and approximate distance) ▶ Output the scaffolded consensus sequences and corresponding quality scores, along with an AGP file of the scaffolds and specific metrics Tables
GS Reference Mapper		Sample consensus sequence, mapped to a reference sequence; and list of mutations	▶ For each read, search for alignment(s) to the reference sequence, in nucleotide space ▶ Construct contigs and compute a consensus basecall sequence from the signals of the aligned reads (flowspace) ▶ Identify the positions where the consensus or subsets of the reads that comprise it differ from the reference sequence (or reads from one another); these are the "putative differences" ▶ Evaluate the putative differences to identify high-confidence differences ▶ Output contig consensus sequence(s) and corresponding quality scores, an ACE file of the multiple alignments of the reads and contigs to the reference, the list of identified differences, and mapping metrics files
GS Amplicon Variant Analyzer		Identity and quantitation of sequence variants	▶ Trim reads (remove primer sequences) ▶ Assign reads to "Samples" (Demultiplex datasets) ▶ Align Sample reads to their reference sequences ▶ Quantitate variant frequency for each Sample

Table 1–2: The 3 applications of the data analysis phase of the Genome Sequencer FLX System, with their inputs, outputs, and main processing steps. Note that all data analysis applications use as input the reads and flowgrams output in SFF format by the data processing (GS Run Processor application). For a description of the data processing pipeline options, see section 1.2. For a full description of the various applications, see the corresponding chapters in this manual.

1.2 Data Processing Options

The Genome Sequencer FLX System gives the user five options for processing the data from a sequencing Run, by selecting among the available processing types during sequencing Run set up (see Figure 1–1; for more detail on processing types and sequencing Run set up, see the *GS FLX Titanium Sequencing Method Manual*):

- In most cases, the user will perform the image processing step of data processing on the instrument, copy the partially processed data to a dedicated computational node, and carry out the time-consuming signal processing step there. To do this, the user would select the "Image processing only" processing type. This minimizes the amount of time the Genome Sequencer FLX Instrument is busy processing and unavailable for more experiments: when included in a sequencing Run, the signal processing step could continue for up to 80 hours after the end of image acquisition for a 200 cycle Run performed with the GS FLX Titanium chemistry. The image processing step, by contrast, takes place concurrently with image acquisition.
- In some cases, the user will elect to include both the image and signal processing steps in the sequencing Run (select one of the "Full" processing types). This is the simplest option from the standpoint of instrument operation, as all data processing up to the generation of read flowgrams and basecalling of the reads is carried out during the Run, without user intervention. However, it can take up to 80 hours to process a GS FLX Titanium chemistry Run on-instrument. The user then proceeds with the post-Run analysis step(s) (which must await the acquisition of all the read data), as appropriate for the experiment.
- An alternate "Full processing for Amplicons" processing type exists in the Genome Sequencer FLX System, which uses data processing algorithms that are specially tuned for Amplicon sequencing; this option should be selected if you are sequencing an Amplicon library and you want to carry out the full Run-time data processing on it. It is also used for test Runs that use only Control DNA beads. If "Image processing only" is carried out during the sequencing Run, a signal processing for Amplicons option is available off-instrument.
- Similarly, another alternate "Full processing for Paired End" processing type also exists, to fully process sequencing Runs of Paired End libraries. If "Image processing only" is carried out during the sequencing Run, a signal processing for Paired End option is available off-instrument.
- Finally, the user can choose to only acquire the raw images during the Run (select the "No processing" processing type). This option should be selected, for example, for procedures that do not generate sequencing data, such as pre-wash and maintenance wash Runs. If selected for a sequencing Run, all other steps of data processing and analysis can be carried out separately afterwards. This provides users with maximum flexibility in data handling to conform to their desired processing architecture.

As shown in Figure 1–1, data processing or data analysis applications that are carried out separately from the sequencing Run (*i.e.* not determined by the processing type selection) are invoked via the GS Run Browser or at the UNIX "command line" level. This must be done off-instrument, on a separate computer (called a "DataRig") or on a computer cluster. The GS Run Processor will produce comparable results whether it is run on-instrument or off-instrument. <u>This Manual is written from the perspective of the off-instrument versions of all the applications.</u>

The GS *De Novo* Assembler, GS Reference Mapper and GS Amplicon Variant Analyzer applications, and the SFFTools commands, are *always* run separately from the sequencing Run (*i.e.* off-instrument, on a DataRig or a computer cluster). The rationale for this is that these applications either are not usually applied to individual Runs but rather draw on multiple Runs, or require additional information beyond the Run data (such as a reference genome against which to map the sequencing reads). In addition, these applications can take hours to complete, during which time the Genome Sequencer FLX Instrument would not be available for another Run if it were tied up in data processing or data analysis. Run troubleshooting may also be time intensive, so the GS Reporter and the GS Run Browser applications are provided both on the Genome Sequencer FLX Instrument and as separate applications that can be run on a DataRig or the computer cluster.

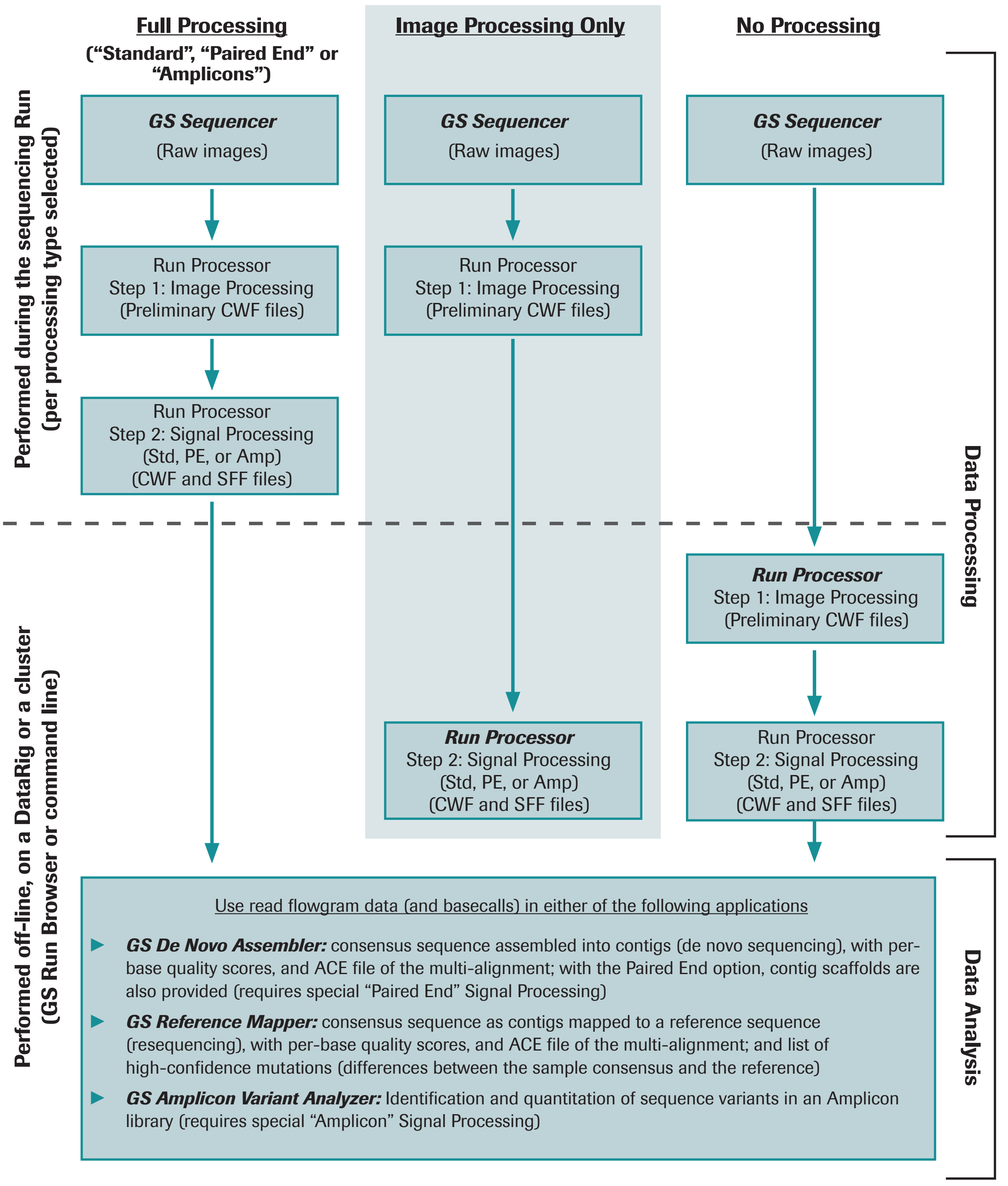

Full Processing
("Standard", "Paired End" or "Amplicons")
Image Processing Only
No Processing
Performed during the sequencing Run (per processing type selected)
Performed off-line, on a DataRig or a cluster (GS Run Browser or command line)
GS Sequencer (Raw images)
Run Processor Step 1: Image Processing (Preliminary CWF files)
Run Processor Step 2: Signal Processing (Std, PE, or Amp) (CWF and SFF files)
Data Processing
Data Analysis
Use read flowgram data (and basecalls) in either of the following applications
GS De Novo Assembler: consensus sequence assembled into contigs (de novo sequencing), with per-base quality scores, and ACE file of the multi-alignment; with the Paired End option, contig scaffolds are also provided (requires special "Paired End" Signal Processing)
GS Reference Mapper: consensus sequence as contigs mapped to a reference sequence (resequencing), with per-base quality scores, and ACE file of the multi-alignment; and list of high-confidence mutations (differences between the sample consensus and the reference)
GS Amplicon Variant Analyzer: Identification and quantitation of sequence variants in an Amplicon library (requires special "Amplicon" Signal Processing)

Figure 1–1: Data processing options in the Genome Sequencer FLX System. The blocks identify the various data processing or data analysis applications (or steps) and their outputs. The raw images are captured as part of the sequencing Run. Depending on the processing type selected during set up of the sequencing Run (top of each column), the image processing and signal processing steps can either be performed as part of the sequencing Run (above the dotted line); or they can be carried out separately following the Run, using the off-instrument version of the software, on a DataRig or a cluster (below the dotted line). Note the three options for signal processing: the standard algorithm is for General (*e.g.* shotgun) sequencing, and alternative ones exist for Paired End and for Amplicon sequencing (and test Runs). The preferred data processing path is highlighted. Data analysis applications (GS *De Novo* Assembler, GS Reference Mapper and GS Amplicon Variant Analyzer), on the other hand, are always carried out separately, and are supported via a Graphical User Interface in the GS Run Browser, or at the command line level. Applications launched by direct user input are shown in *Bold-Italics* (others are embedded in the Run). The GS Run Browser application (not shown) can be invoked either on the Genome Sequencer FLX Instrument or on a DataRig or a cluster to view raw images and other Run data, *e.g.* for troubleshooting purposes or to reanalyze the dataset using different settings; this is done after a Run completes (or is aborted).

1.3 Data Output and Folder Structure

This section provides an overview of the listing and organization of the files that are generated at each step of data processing or data analysis, and made available to the user.

After a sequencing Run, results can be found in the "*/data*" directory on the Genome Sequencer FLX Instrument, grouped by date, where each "date" folder contains individual Run folders. Each Run folder is identified by the Run name specified during Run set up (*i.e.* with an 'R_' prefix, denoting 'R'un; see first "Note" below). Run folders contain the raw data of the sequencing Run (*i.e.* the results of the GS Sequencer application), as well as any Data Processing folder(s) for that Run (identified by a 'D_' prefix, for 'D'ata). Data Processing folders, in turn, contain the results of the image processing and/or signal processing steps.

"tmp" sub-directory required: The */data* directory in the on-instrument computer contains a tmp sub-directory. Do not delete the tmp sub-directory as it is required for the proper functioning of the instrument software.

Because the data analysis applications are typically performed on a pool of sequencing Runs rather than on any single Run, their results are not associated with a specific sequencing Run or data processing invocation; rather, assembly, mapping and amplicon variant analysis results are deposited in the "current working directory" at the time the application is launched, or written to a directory specified by the user (see section 1.3.3). The SFFTools commands also usually deposit their output into the "current working directory" or a directory specified on the command line.

- The user enters a unique name for the Run during set up. A good choice for a unique name could be structured as follows: "PicoTiterPlate device size and barcode #, kit lot #, genome, run #"; *e.g.* "70×75_123456_081708_ecoli_Run1". To form the complete name of the Run files, the date stamp, time stamp, instrument name, and user name will be added automatically in front of this unique Run name. The name structure is:

  "R_yyyy_mm_dd_hh_min_sec_machineName_userName_uniqueRunName".

- Before any data processing or data analysis application can be run on a DataRig or a cluster (GS Run Processor, GS *De Novo* Assembler, GS Reference Mapper, or GS Amplicon Variant Analyzer), the necessary input data (see Table 1–1 and Table 1–2) for the sequencing Run(s) being processed or analyzed must be made available on that DataRig or cluster. The Genome Sequencer FLX Instrument can be configured to automatically transfer the output files that result from the GS Run Processor application to a remote disk. (See the section on data transfer scripts in the *Genome Sequencer System Site Preparation Guide*) Alternatively, the Data tab of the GS Sequencer application can be used on the Genome Sequencer FLX Instrument to transfer the raw images of sequencing Run(s) to a pre-configured destination. (For more information on the Data tab of the GS Sequencer application, see the *Genome Sequencer FLX Operator's Manual.*)

1.3.1 Data Acquisition (GS Sequencer) Results: the Run Folder

The organization of a generic Run ('R_') folder is depicted in Figure 1–2. All the raw data (raw images, log files, *etc.*) remain in temporary local storage on the Genome Sequencer FLX Instrument in case the user chooses to re-analyze them (*e.g.* using the reanalysis function of the GS Run Browser; see section 8). In addition, if "Backup" is selected during Run set up, raw and processed data files from the Run can be transferred to a network location specified by the *System Administrator*, for long-term storage. See section 2 for full file descriptions.

```
R_yyyy_mm_dd_hh_min_sec_machineName_userName_uniqueRunName/
              dataRunParams.parse
              imageLog.parse
              PTP_flowOrder_cycleCount.icl
              runLog.parse
              aaLog.txt
              rawImages/
                     00001.pif
                     00002.pif
                     00003.pif
                     …
```

Figure 1–2: General organization of a "Run" folder. On the Genome Sequencer FLX Instrument, this is located inside /data/*date*; while on a DataRig or a cluster, it can be placed anywhere. Words in *italics* are generic

1.3.2 Data Processing (GS Run Processor) Results: the Data Processing Folder

The organization of a generic Data Processing folder ('D_') is depicted in Figure 1–3. D_ folders are created by the GS Run Processor application, within the R_ folder of the sequencing Run whose data is being processed. Since a dataset can be re-processed multiple times (via the GS Run Browser; see section 8), a given R_ folder can contain multiple D_ folders. To the extent that they are generated on-instrument, per the processing type selected (see section 1.2), all the processed data (basecalls and quality scores, Run metrics, log files, *etc.*) remain in temporary local storage on the Genome Sequencer FLX Instrument. In addition, if "Backup" is selected during Run set up, raw and processed data files from the Run can be transferred to a network location specified by the *System Administrator*, for long-term storage. See section 3 for full file descriptions and for more information on the GS Run Processor application.

The Genome Sequencer FLX Instrument processes the sequencing data "on-the-fly," *i.e.* the data is processed (to the extent specified in the processing type selected) and deposited in the Data Processing folder, concurrently with the Run. When the "Run Completed" window appears on the screen, the processing of the sequencing Run has completed and the results are ready for further processing or transfer.

```
R_yyyy_mm_dd_hh_min_sec_machineName_userName_uniqueRunName/1
                [...]
                D_yyyy_mm_dd_hh_min_sec_machineName_analysisType/2
                        gsRunProcessor.log
                        gsRunProcessor_err.log
                        dataRunParams.xml
                        regions/2
                                region.cwf2
                        sff/2
                                uaccnoRegion.sff2
```

Figure 1–3: General organization of a "Data Processing" folder. Data Processing ('D_') folders are created within the corresponding Run's 'R_' folder; an R_ folder can contain multiple D_ folders, if the dataset was re-processed. Words in *italics* are generic. The superscripts indicate the application by which the folders and files are generated: GS Sequencer^1, GS Run Processor^2. The set of SFF files are generated during the signal processing step of the GS Run Processor, using the "universal" accession prefix described in section 13.3.7. See section 3 for full file descriptions.

Note on Software v. 2.0: The file output of the data processing pipeline has changed considerably in version 2.0 of the software. The main differences are that all output files from the processing pipeline are now stored in a "Composite Wells File" (.cwf) and that each invocation of the processing pipeline generates a new D_ directory. A separate application, the GS Reporter, can still generate the usual metrics and FASTA files, and is indeed evoked by default by all the "Full Processing" data processing types (section 1.2).

1.3.3 Data Analysis Applications Results

As indicated above, the data analysis applications are often performed on a pool of sequencing Runs rather than on any single Run, and/or can require additional information beyond the Run data. For this reason, they are carried out off-instrument, via a Graphical User Interface (GUI) or from the UNIX command line on a DataRig, rather than on the Genome Sequencer FLX Instrument.

As a consequence, the result files generated by these applications are not deposited in a Run folder. For the GS *De Novo* Assembler and the GS Reference Mapper, rather, one of the following will apply:

- A folder with a 'P_' prefix (for 'P'ost-Run Analysis) is created to receive them, in the user's current working directory on the DataRig at the time the application is launched, or written to a directory specified by the user via the applications' GUI or on the command line.
- Mapping and Assembly can also be carried out in a "project-based" fashion, whereby datasets can be added to existing results (or a new reference sequence can be specified) for an existing Assembly or Mapping "project". This uses the corresponding applications' GUI (or can be done using the newAssembly, newMapping, and associated commands), and the data is then stored in a "Project" folder. A Project folder is identified by the "454Project.xml" file it contains.

The folder and file structures generated for each of these commands (or GUI equivalents) are shown in the Figures below: the "one-step" runAssembly (Figure 1–4) or runMapping (Figure 1–5) commands, for the "standard", non-incremental assembly or mapping of one or more Runs; and newAssembly (Figure 1–6) or newMapping (Figure 1–7), for the "project-based", incremental assembly or mapping of one or more Runs. See sections 5 and 6 for full file descriptions and for more information on the GS *De Novo* Assembler and GS Reference Mapper applications. Both assembly and mapping can also be performed using a graphical interface, as described in detail in sections 5 and 6 of this document.

The GS Amplicon Variant Analyzer application can also operate either via a GUI or via its own Command Line Interface (the AVA-CLI), but its output is not structured like that of the other two data analysis applications. See sections 9 through 12 for details on this application.

```
$[current_working_directory] or [directory_specified]
       P_yyyy_mm_dd_hh_min_sec_runAssembly/
              454AllContigs.fna
              454AllContigs.qual
              454LargeContigs.fna
              454LargeContigs.qual
              454NewblerMetrics.txt
              454NewblerProgress.txt
              454AlignmentInfo.tsv
              454ContigGraph.txt
              454PairAlign.txt
              454ReadStatus.txt
              454Contigs.ace or ace/ContigName.ace or consed/…
              sff/
              *.sff
              454Scaffolds.fna¹
              454Scaffolds.qual¹
              454Scaffolds.txt¹
```

Figure 1–4: File output of the GS *De Novo* Assembler application, using the runAssembly command (or its GUI equivalent). All result files (specifying the actual *contig names*) are placed in a folder with a 'P_' prefix, within the user's current working directory when running the command or in a directory specified by the user. All input SFF files used in the assembly are organized in the sff sub-directory. [1]These files are produced only when the Paired End option is used; the Paired End option also adds sections to the 454NewblerMetrics.txt file. See section 5 for full file descriptions.

```
$[current_working_directory] or [directory_specified]
       P_yyyy_mm_dd_hh_min_sec_runMapping/
              454AllContigs.fna
              454AllContigs.qual
              454LargeContigs.fna
              454LargeContigs.qual
              454AllDiffs.txt
              454HCDiffs.txt
              454NewblerMetrics.txt
              454NewblerProgress.txt
              454MappingQC.xls
              454AlignmentInfo.tsv
              454PairAlign.txt
              454ReadStatus.txt
              454RefStatus.txt
              454Contigs.ace or ace/refaccno.ace or consed/…
              sff/
                     *.sff
```

Figure 1–5: File output of the GS Reference Mapper application, when using the runMapping command (or its GUI equivalent). All result files (specifying the actual *reference sequence accession numbers*) are placed in a folder with a 'P_' prefix, within the user's current working directory when running the command or in a directory specified by the user. All input SFF files used in the mapping are organized in the sff sub-directory. See section 6 for full file descriptions.

```
$[current_working_directory] or [directory_specified]
       P_yyyy_mm_dd_hh_min_sec_runAssembly/
              assembly/
                     454AllContigs.fna
                     454AllContigs.qual
                     454LargeContigs.fna
                     454LargeContigs.qual
                     454NewblerMetrics.txt
                     454NewblerProgress.txt
                     454AlignmentInfo.tsv
                     454ContigGraph.txt
                     454PairAlign.txt
                     454ReadStatus.txt
                     454Contigs.ace or ace/ContigName.ace or consed/…
                     454AssemblyProject.xml
                     454Scaffolds.fna[1]
                     454Scaffolds.qual[1]
                     454Scaffolds.txt[1]
              sff/
                     *.sff
              454Project.xml
```

Figure 1–6: File output of the GS *De Novo* Assembler application, when using the newAssembly and related commands (or their GUI equivalents) for "project-based" assembly. All result files (specifying the actual *contig names*) are placed in a folder within the user's current working directory when running the command or in a directory specified by the user. This is identified as a "Project" folder by the presence of a 454Project.xml file within it. All assembly status and result files are organized in the assembly sub-folder; and all input SFF files used in the assembly (or symbolic links to them) are organized in the sff sub-folder. [1]These files are produced only when the Paired End option is used; the Paired End option also adds sections to the 454NewblerMetrics.txt file. See section 5 for full file descriptions.

```
$[current_working_directory] or [directory_specified]
       P_yyyy_mm_dd_hh_min_sec_runMapping/
              mapping/
                     454AllContigs.fna
                     454AllContigs.qual
                     454LargeContigs.fna
                     454LargeContigs.qual
                     454AllDiffs.txt
                     454HCDiffs.txt
                     454NewblerMetrics.txt
                     454NewblerProgress.txt
                     454MappingQC.xls
                     454AlignmentInfo.tsv
                     454PairAlign.txt
                     454ReadStatus.txt
                     454RefStatus.txt
                     454Contigs.ace or ace/refaccno.ace or consed/…
                     454MappingProject.xml
              sff/
                     *.sff
              454Project.xml
```

Figure 1–7: File output of the GS Reference Mapper application, when using the newMapping and related commands (or their GUI equivalents) for "project-based" mapping. All result files (specifying the actual *reference sequence accession numbers*) are placed in a folder within the user's current working directory when running the command or in a directory specified by the user. This is identified as a "Project" folder by the presence of a 454Project.xml file within it. All mapping status and result files are organized in the mapping sub-folder; and all input SFF files used in the mapping (or symbolic links to them) are organized in the sff sub-folder. See section 6 for full file descriptions.

The rest of this manual describes the individual software applications and commands of the Genome Sequencer FLX System, from the standpoint of their off-instrument usage, on a DataRig or a cluster. (Note that GS Run Processor can also be performed as part of a sequencing Run, using the GS Sequencer application or the reanalysis function of the GS Run Browser application on the Genome Sequencer FLX Instrument; see Figure 1–1 and the *Genome Sequencer FLX Operator's Manual*.)

The commands, command line options (or GUI equivalents), and general inputs and outputs, as well as some detail of the purpose and function of each application are provided. This includes the raw data output of the sequencing Run (section 2) even though it is acquired on the Genome Sequencer FLX Instrument, because it is the first possible input for a off-instrument application (image processing step of the GS Run Processor; see Figure 1–1). Also described are the "SFF Tools" (section 7), a set of commands used to create, manipulate and access sequencing trace data from SFF files; and the GS Reporter (section 4) and the GS Run Browser applications (section 8), used to view and troubleshoot the results of a completed sequencing Run which are available both on the instrument and in the off-instrument software package. Note, however, that the GS Reporter and GS Run Browser applications and the SFF Tool commands are not required steps of data processing.

The sections are as follows:

Part B: Data Processing

- Section 2: GS Sequencer (Output Only; Not a DataRig Application)
- Section 3: GS Run Processor Application
- Section 4: GS Reporter Application

Part C: Data Analysis (Assembly and Mapping)

- Section 5: GS ***De Novo*** Assembler Application
- Section 6: GS Reference Mapper Application
- Section 7: SFF Tool Commands
- Section 8: GS Run Browser Application

Part D: Data Analysis (Amplicon Variant Analysis)

- Section 9: GS Amplicon Variant Analyzer Application
- Section 10: Example Amplicon Project Design and Analysis
- Section 11: GS Amplicon Variant Analyzer Command Line Interface
- Section 12: GS Amplicon Variant Analyzer – Special Topics

The "output" sub-sections below define the files produced by the data processing and data analysis applications described in this manual (except for the GS Amplicon Variant Analyzer, whose output structure is different), and provide the exact syntax of the file names that would be generated by a sequencing Run with 2 analysis regions and 2 sequencing keys. This would be the case, for example, for a Run using a 70×75 mm PicoTiterPlate device (2 large regions gasket), with a General DNA library sample spiked with Control DNA Beads (each with its sequencing key). See the *Genome Sequencer FLX Operator's Manual* for more information.

Data Processing

B

Part B: Data Processing

2. GS Sequencer (Output Only; Not a DataRig Application) 42

2.1 GS Sequencer Application Output ... 42
2.1.1 dataRunParams.parse ... 43
2.1.2 imageLog.parse ... 44
2.1.3 Raw Image Files ... 44

3. GS Run Processor Application 45

3.1 The Image Processing Step ... 46
3.1.1 Launching the Image Processing Step of Data Processing ... 46
3.1.2 Image Processing Step Output ... 48
3.1.2.1 *region*/dataRunParams.xml ... 49
3.1.2.2 *region*/regionnum.cwf ... 49
3.1.2.3 gsRunProcessor.log and gsRunProcessor_err.log ... 49
3.2 The Signal Processing Step ... 51
3.2.1 Introduction to Signal Processing ... 51
3.2.2 Quality Filtering and Trimming ... 52
3.2.2.1 Quality Filters ... 52
3.2.2.1.1 The Keypass Filter ... 52
3.2.2.1.2 The Dots Filter ... 52
3.2.2.1.3 The Mixed Filter ... 53
3.2.2.1.4 The Signal Intensity Filter ... 53
3.2.2.1.5 The Primer Filter ... 53
3.2.2.1.6 The TrimBack Valley Filter ... 54
3.2.2.1.7 The Quality Score Trimming Filter ... 55
3.2.2.2 Individual Base Quality Scores ... 56
3.2.2.3 Modifying the Stringency of Quality Filtering ... 56
3.2.3 Launching the Signal Processing Application ... 60
3.2.4 Signal Processing Step Output ... 62
3.2.4.1 *region*.cwf ... 64
3.2.4.2 *uaccnoRegion*.sff ... 64
3.2.4.3 gsRunProcessor.log and gsRunProcessor_err.log ... 64

4. GS Reporter Application 65

4.1 Launching the GS Reporter Application ... 65
4.2 GS Reporter Application Output Files ... 67
4.2.1 *region*.wells ... 70
4.2.2 *region*.trimInfo ... 70
4.2.3 *region.wells.key*.454RuntimeMetrics.txt ... 71
4.2.4 454QualityFilterMetrics.txt ... 71
4.2.5 454BaseCallerMetrics.txt ... 73
4.2.6 454RuntimeMetricsAll.txt ... 74
4.2.7 *region.key*.454Reads.fna ... 78
4.2.8 *region.key*.454Reads.qual ... 78

2. GS Sequencer (Output Only; Not a DataRig Application)

The raw images are acquired using the GS Sequencer application on the Genome Sequencer FLX Instrument. The GS Sequencer application must be run directly on the Genome Sequencer FLX Instrument as it requires direct communication with the on-board camera and microcontrollers. For more information on the GS Sequencer application, see the *Genome Sequencer FLX Operator's Manual* and the *GS FLX Titanium Sequencing Method Manual*.

2.1 GS Sequencer Application Output

The files generated by the GS Sequencer application include general Run and image set information, and a set of raw images (".pif" format, in the 'rawImages' sub-folder; see Table 2–1). These are placed in a Run folder ('R_'; see Figure 1–2). The number of images captured during the Run is determined by the Run script. You can view the output files from the sample datasets provided in the accompanying DVD, using the GS Run Browser application.

File Name Structure	Definition
dataRunParams.parse	Metadata about the sequencing Run, such as the user name, group name, Run unique name, PicoTiterPlate barcode, software revision numbers, and kit identifier. This file is required in order to perform the image processing step of data processing. The dataRunParams.parse file also contains information about the bead loading gasket used for the sequencing Run, including the number and exact location of regions. This file can also contains a list of the Control DNA sequences and key sequenced used in the sequencing Run.
imageLog.parse	Log of all the images of the data set, including the reagent that was being flowed during the capture of each image. This file is required in order to perform the image processing step of data processing.
#####.pif	Raw image files, showing photon counts for each pixel (in a rawImages sub-folder). These files are required in order to perform the image processing step of data processing.

Table 2–1: Files generated by the GS Sequencer application

For any sequencing Run (with any number of analysis regions), the files produced by the GS Sequencer application will have the following names (for a general description of the ".parse" and ".pif" file formats, see section 13):

- dataRunParams.parse
- imageLog.parse
- ptpImage.pif[1]
- runlog.parse[1]
- aaLog.txt[1]
- *copy_of_instrument_control_file*.icl[1]
- rawImages (folder)
  - 00001.pif
  - 00002.pif
  - 00003.pif
  - …

[1] [These files are generated for debugging purposes. While not strictly required for further analysis, it is recommended that these be archived with the raw data for possible debugging.]

2.1.1 dataRunParams.parse

The information in the dataRunParams.parse file comes from the settings entered in the Run Wizard's windows of the GS Sequencer application (see the *GS FLX Titanium Sequencing Method Manual*). Additional entries are derived from data sources contained on-board the instrument. It contains the following important values:

1. runParameters group
   a. runId – The "unique name" for the Run as entered by the operator during Run setup. This is a free-form field.
   b. runName – The full Run name. See Note in section 1.3 for information on the makeup of the runName.
   c. operatorName – The name of the operator and the operator's e-mail address.
   d. groupName – The group of experiments of which this Run is a part. This is selected by the operator during Run setup.
   e. ptpBarcode – The PicoTiterPlate device identification number.
   f. ptpType – The dimensions of the PicoTiterPlate device type used in the Run, in millimeters. The possible values are "70×75" or "25×75".
   g. padLayout – Name of the layout describing the number of regions for the Run. This is kept for legacy purposes and is superseded by the "regionsToProcess" group below.
   h. imageWidth – The pixel width of each PIF file, in pixels
   i. imageHeight – The pixel height of each PIF file, in pixels
   j. postRunProcessing – This field is used by the LIMS Lookup feature. See the GS *LIMS Implementation Guide* for more information.
   k. kitId – The type of GS Sequencing kit used during the experiment. If the operator selected a "Custom" kit in the Run setup wizard, the word "custom" will appear here.
   l. ICMVersion – The version number of the software on the instrument used to collect the original (image) data.
   m. instrumentSerialNumber – The serial number of the instrument used to collect the data.
   n. ptpPitch – The pitch of the PicoTiterPlate device used to collect the data.

2. regionsToProcess group
   a. region – There should be one "region" group below each regionsToProcess group for each region loaded on the PTP device.
      I. name – The region's name. This parameter is reserved for future use; it will always be set to "Region" and the region number.
      II. location – Location of the region, in format "startX, startY, endX, endY" where each value is in pixels. All pixel values are relative to the camera's 4096 × 4096 imaging area. This is true even if the actual image dimensions are different, as in the case of 25×75 mm PTP devices, where the images are 2048 pixels wide.
      III. analysisType – Reserved for future use. Always set to "analysis".

3. sequencesToSearchFor group – Identifies the pool of Control DNA keys used during the sequencing Run.
   a. useKitDefaults – Should be set to "true" for most cases.

The data contained in the regionsToProcess group and the sequencesToSearchFor group is taken from a default template file on the Genome Sequencer FLX Instrument, one for each region layout.

For a general description of the ".parse" file format, see section 13.3.2. Several examples of parser files are shown in section 13.4.

2.1.2 imageLog.parse

The imageLog.parse file contains data about the images that were captured during the sequencing Run and stored in the rawImages directory. This file is mainly used to distinguish the nucleotide flow images, the PPi flow images, the adaptive apyrase adjustment images, and the wash flow images. This file contains the following information:

1. imageSuperStep group
   a. image – One quoted, comma-separated line of summary information about each image, consisting of:
      I. Date and time stamp
      II. Full path to the image's PIF file
      III. System-level timestamp
      IV. The image type, where "A", "C", "G" and "T" are the nucleotide flows, "PPI" is the PPi flow, and all others are images used internally for instrument operation.

For a general description of the ".parse" file format, see section 13.3.2. Several examples of parser files are shown in section 13.4.

2.1.3 Raw Image Files

The images themselves are a record of the photon counts that the camera captured during each flow period. The PIF file format is a 454-defined format (see section 13). Note that the camera capture system adds a 500 count offset to the value of each pixel, so the value used in processing is 500 less than the value stored in the PIF file. For a general description of the ".pif" file format, see section 13.3.3.

The approximate size of the set of images created during a sequencing Run for each sequencing kit type and Run script (assuming the largest bead loading regions available on the PTP device) is as follows:

Sequencing Kit	Number of Cycles	Raw Images
XLR70	200	28 Gb
XLR70	150	21 Gb
XLR70	100	14 Gb
LR70	100	14 Gb
LR70	42	5 Gb
LR25	100	6 Gb
LR25	42	3 Gb
SR70	42	5 Gb

3. GS Run Processor Application

The GS Run Processor application carries its work in two steps: image processing and signal processing. The image processing step extracts the raw signals for each nucleotide flow, in each active well of the PicoTiterPlate device, from the raw images captured by the Genome Sequencer FLX Instrument during the sequencing Run. The signal processing step corrects these flow signals for optical and chemical artifacts, and produces basecalls. The GS Run Processor can call a user-provided script for automatically transferring its output files for further processing or archiving. This is described in the *Genome Sequencer System Site Preparation Guide*.

The output of the GS Run Processor is packaged into a set of Composite Wells Format (.cwf) files and Standard Flowgram Format (.sff) files. The GS Reporter application (described in section 4) is evoked by default by all the “Full Processing” data processing types (see section 1.2), to extract information from the CWF and SFF files output by the GS Run Processor, and generate a set of metrics and FASTA files that can be used to assess the quality of a Run. It is also important to note that the GS Reporter application can generate output files in formats similar to these produced by previous version of the data processing software. Each of the two steps of data processing generates its own D_ directory.

Table 3–1 shows the approximate amount of disk space needed to store the various types of files produced by the GS Run Processor application. If the storage space available on the data processing computer is not sufficient, a warning will appear on screen.

Sequencing Kit	Number of Cycles	Raw Images Only	Raw Images Plus Image Processing	Raw Images Plus Full Processing
XLR70	200	28 Gb	30 Gb	37 Gb
XLR70	150	21 Gb	23 Gb	28 Gb
XLR70	100	14 Gb	16 Gb	18 Gb
LR70	100	14 Gb	15 Gb	16 Gb
LR70	42	5 Gb	6 Gb	7 Gb
LR25	100	6 Gb	7 Gb	8 Gb
LR25	42	3 Gb	4 Gb	4 Gb
SR70	42	5 Gb	6 Gb	7 Gb

Table 3–1: Approximate amount of data generated per sequencing Run by each sequencing kit type, Run script, and data processing option. These numbers assume that the bead loading gasket with the largest possible regions is used.

3.1 The Image Processing Step

The image processing step of the GS Run Processor application takes the captured, or "raw" images produced during a sequencing Run, and performs the following operations (see also Table 1–1):

- Subtract background and normalize the images (at the pixel level)
- Find the centers of the active wells in the PicoTiterPlate device (*i.e.* the locations where sequencing reactions are taking place)
- For each active well, extract the raw signals from the images corresponding to all nucleotide or PPi flows
- Write the resulting flow signals to disk for further processing.
- As part of the image processing, the data corresponding to the different loading regions of the PicoTiterPlate device are separated into different files, and the data from each region is processed separately throughout image and signal processing.

There is usually no reason to call the image processing step separately. Most users can skip to section 3.2, on the signal processing step. Normally, image processing is performed in real-time, during the sequencing Run, as the images are being captured (*i.e.* without adding any time to the Run). Even if the "No Processing" processing type was selected during Run set up, the user can call the runAnalysisPipe, runAnalysisPipePairedEnd, or runAnalysisPipeAmplicons command and pass the R_ directory as a parameter to perform the entire data processing in one call.

3.1.1 Launching the Image Processing Step of Data Processing

There are three ways to launch the image processing step of the GS Run Processor application:

- from the command line,
- from the GS Run Browser application, and
- as part of an automated pipeline processing system (preferred).

This section describes the command line method. See section 8 of this manual, on the GS Run Browser application for more information on the second method. Information on setting up an automated processing environment for your sequencing Runs can be found in the *Genome Sequencer System Site Preparation Guide.*

Launching the image processing step of the GS Run Processor consists of running a single command, **runImagePipe**, which has the following command line structure:

```
runImagePipe RUN_DIRECTORY [options]
```

Command	Description
runImagePipe	This command runs the algorithms to process the raw data contained in the images into the signal data of each nucleotide flow of the sequencing Run, for each well of the PicoTiterPlate device.

Argument	Description
RUN_DIRECTORY	Path to the Run directory which must contain the dataRunParams.parse file, the imageLog.parse file and the complete contents of the rawImages directory.

Most users will not need to set any additional options, as these parameters are usually passed automatically to the underlying GS Run Processor application. See the section entitled "Data Processing Infrastructure for Software 2.0" in the *Genome Sequencer System Site Preparation Guide* for a description of all available options. Nonetheless, some useful options worth mentioning are:

Option	Description
--progress	Display real-time progress of the job's processing
--pipe=FILENAME	Specify a pipeline with custom parameters
--reg=REGION_NUM	Only process data from a particular region.

Note on Software v. 2.0: It is no longer necessary, or even possible, to specify the region layout or PTP device size on the command line to this function. For data sets generated by version 2.0 of the instrument software, the processing application obtains this information from the dataRunParams.parse file. If the user passes a directory containing a data set generated with a version of the Genome Sequencer software anterior to 2.0, the processing application will use a set of "legacy DN" files to attempt to determine the correct region layout.

Depending on your site's configuration, the runImagePipe command will either begin to process the data immediately or, if the instrument/dataRig/compute cluster is busy, will report that the instrument is busy and to try again later. If the job is able to start, progress information is displayed to the terminal.

The runImagePipe command performs the following actions:

- Creates a new output D_ directory
- Performs a region finding step to determine the exact position and span of all the regions on the images, as well as detecting empty (non-loaded) regions.
- Based on the previous two steps, the image processing step generates a new file in the "regions" subdirectory called "dataRunParams.xml". This file contains the results of the region finding, a list of the Control DNA sequences detected in the data set, and the meta-data about the sequencing Run.
- A number of GS Run Processor sub-processes are launched, which use the rawImages and the dataRunParams.xml files to create .cwf files.

3.1.2 Image Processing Step Output

Whether the image processing is carried out on-instrument as part of the "on-the-fly" analysis of a sequencing Run, or later by launching the image processing step explicitly on a DataRig or a cluster, a Data Processing folder (identified by a 'D_' prefix) is created within the Run folder (Figure 1–2). A new Data Processing folder is created for each invocation of the pipeline (a Run folder can thus contain multiple Data Processing folders) and given its own software-generated full name, in the same format as Run folders (see "Note" in section 1.3) but with the current date and time stamp, *etc.*, and the pipeline name *e.g.*:

```
"D_yyyy_mm_dd_hh_min_sec_machineName_userName_imageProcessing"
```

Note on Software v. 2.0: This is fundamentally different from the behavior of the versions of the image processing software anterior to 2.0. In versions prior to 2.0, the on-the-fly image processing created a single D_ directory with the suffix "OTFAnalysis". When the user performed the signal processing part of the data processing, the original image processing results were copied into a subdirectory within the D_ directory called "regions.orig", and the signal processing results were placed in the regions directory. Each subsequent invocation of the signal processing pipeline overwrote the previous one.

With the software version 2.0, each invocation of the image processing or signal processing pipeline generates its own D_ directory. A regions.orig directory is no longer created, as the original results are stored in the first D_ directory (which takes the "imageProcessingOnly" suffix), with no risk of overwriting.

Within the D_ directory, the GS Run Processor application creates the files and folders listed in Table 3–2. For examples, you can view the output files from the sample datasets provided in the accompanying DVD, using the GS Run Browser application. Note that version 2.0 or later of the GS Run Browser application is REQUIRED to view data sets generated by version 2.0 of the software.

File Name Structure	Definition
region/dataRunParams.xml	This file contains an XML digest of the meta-data that will be used to process the data from the sequencing Run. This file is for reference only and may be deleted at any time.
region/*regionnum*.cwf	The "composite wells format file." This single file (per region) contains the processed flowgrams, Run metrics, Run meta-data, and all intermediate results. The GS Reporter application uses the CWF file to extract various reports (see section 4).
gsRunProcessor.log	This file contains a log of the progress of processing. Users may view this file during the Run to monitor progress. This file is for reference only and may be deleted at any time
gsRunProcessor_err.log	This file contains a list of errors generated during data processing. It is only generated if errors were encountered. This file is for reference only and may be deleted at any time

Table 3–2: Files generated by the image processing step of the GS Run Processor application. Words in *italics* are generic.

For a sequencing Run with 2 analysis regions where the region names are "1" and "2", the files produced will have the following names:

- regions (folder)
  - dataRunParams.xml
  - 1.cwf
  - 2.cwf
- gsRunProcessor.log
- sRunProcessor_err.log

3.1.2.1 ***region/*dataRunParams.xml**

The dataRunParams.xml file is used as an intermediate file within the pipeline. Its purpose is to provide a consistent interface between the various versions of dataset types (produced by the GS 20 chemistry, the GS FLX standard chemistry, or the GS FLX Titanium chemistry) and the pipeline itself. This file can be considered transient and may be deleted as soon as the work of the GS Run Processor is complete. As an internal file, the contents of this file will most likely change with each revision of software and, therefore, should not be parsed by the customer.

3.1.2.2 ***region/*regionnum.cwf**

The data processing pipeline generates one Composite Wells Format (CWF) file per region, each with a .cwf extension. The CWF file is a "package" format that contains not only the uncorrected flowgrams, but also meta data about the sequencing Run and other intermediate metrics and data from the data processing. These flowgrams are further corrected by the signal processing step of the GS Run Processor application (see section 3.2.4.1). The GS Reporter application depends on the data generated by the signal processing step to generate its output reports. Therefore, running the GS Reporter application with CWF files generated by the image processing step of the pipeline is not allowed. The format of the CWF file is described in section 13.3.1.

Note on Software v. 2.0: The composite wells format file is a new introduction for version 2.0 of the software. It is basically a ZIP file containing the compressed wells information as well as all the data required to generate any of the output artifacts. This compression is useful given the increased volume of data generated by the new GS FLX Titanium chemistry kits.

3.1.2.3 ***gsRunProcessor*.log and *gsRunProcessor_err*.log**

This pair of files provides the complete set of logging information from the GS Run Processor application. Each line is stamped with the time, severity and source of the message. An excerpt of the file is shown below.

```
[Mon Jul 07 17:04:02 2008][Information][]
gsRunProcessor 20080308172036 (Build 99)  Starting

[Mon Jul 07 17:04:05 2008][Information][]
Detected processor speed: 2992 MHz.

[Mon Jul 07 17:04:05 2008][Notice][ProcessingEngine]
Starting job 39e09a4e-4c68-11dd-8026-001d096bafd0.

[Mon Jul 07 17:04:05 2008][Information][ProcessingEngine]
Using memory-only storage for flowgrams.

[Mon Jul 07 17:04:05 2008][Notice][ProcessingEngine]
Processing Group 0 : Loading data.

[Mon Jul 07 17:04:05 2008][Notice][ProcessingEngine]
Processing Group 0 : Done loading data for region(s): 1.

[Mon Jul 07 17:04:05 2008][Notice][ProcessingEngine]
Processing Group 0 : Starting WellFinder.

[Mon Jul 07 17:04:13 2008][Information][WellFinder]
Region 1 : Found 45644 raw wells.

[Mon Jul 07 17:04:13 2008][Notice][ProcessingEngine]
Processing Group 0 : WellFinder finished.

[Mon Jul 07 17:04:13 2008][Notice][ProcessingEngine]
Processing Group 0 : Starting WellBuilder.

[Mon Jul 07 17:04:23 2008][Information][WellBuilder]
```

Region 1 : 3107 control key pass, 41941 sample key pass.

Messages marked as "Notice" or "Information" are presented for the user's benefit and are normally generated as the data processing progresses. Messages marked as "Warning" are generated when an unexpected condition is encountered. While not necessarily fatal, messages marked as "Warning" should be investigated. Conditions that may result in warning messages but can be safely ignored include:

- An unloaded / empty region
- A region without reads prefixed with the library key (for example, a "Control DNA Only" experiment)
- A missing postAnalysisScript.sh (see the *Genome Sequencer System Site Preparation Guide* for information on setting up automated analysis)

Messages marked as "Error" are generated when the GS Run Processor encounters conditions from which it cannot automatically recover. Regions for which error conditions are encountered will probably produce sub-standard or unusable results.

Messages marked as "Fatal" are generated when the GS Run Processor encounters errors that can prevent the GS Run Processor from proceeding. Conditions like missing files or improper configurations are marked as "Fatal". The GS Run Processor will exit after generating "Fatal" messages and delete any partially generated output files.

For the user's convenience, all messages marked as "Warning", "Error" or "Fatal" are placed in a separate file named "gsRunProcessor_err.log" in addition to being placed in the gsRunProcessor.log file. If no messages of this type are generated, the gsRunProcessor_err.log file will not be created.

3.2 The Signal Processing Step

3.2.1 Introduction to Signal Processing

The signal processing step of the GS Run Processor application analyzes the signal data for each flow for all active wells of each PicoTiterPlate device's loading region, using the data generated during the image processing step and stored in the .cwf files. The Genome Sequencer FLX System comprises three parallel paths for signal processing: the "standard" signal processing is used for regular shotgun sequencing, one for Paired End sequencing; an alternative configuration uses an additional algorithm to remove "ghost wells" and is especially tuned for Amplicon sequencing (or for test Runs using only Control DNA Beads) (see "Note", below).

Following a series of normalization, correction, and quality filtering algorithms, the application threads the remaining (high quality) signals into "flowgrams" for each well (reads). The application also generates basecalls with associated quality scores for the individual reads, and outputs this data as Standard Flowgram Format (or SFF) files that contain all the sequence trace data for the reads. All the data analysis applications (GS *De Novo* Assembler, GS Reference Mapper or GS Amplicon Variant Analyzer) use these SFF files as input. In this process, the software performs the following operations (see also Table 1–1):

- Screen out lower signal ghost wells (see definition in the Glossary, in the *Genome Sequencer FLX Operator's Manual*); this step is included only in the Amplicon sequencing pipeline
- Correct for interwell cross-talk between neighboring wells
- Correct for known "out-of phase" errors (incomplete extension and carry forward; see entries for these terms in the Glossary, in the *Genome Sequencer FLX Operator's Manual*)
- Correct for signal droop and subtract residual background signal
- Screen out any remaining ghost wells (in Amplicon sequencing pipeline only)
- Filter (pass or fail) the processed reads based on signal quality (see section 3.2.2)
- Trim read ends for low quality and primer sequence (see section 3.2.2)
- Generate "flowgrams" and basecalled sequences with corresponding quality scores for all individual, high quality reads (*i.e.* those which passed all filters). This information is stored as Standard Flowgram Format (SFF) files, to be used as input to the data analysis applications.

The special "Amplicons" configuration of this application is important because Amplicon libraries tend to generate much stronger signals than other library types, perhaps because they often consist of shorter fragments that amplify to a greater extent during emPCR. Very strong signals, coupled with the fact that many wells in Runs with Amplicon libraries contain similar sequences, can result in "ghost wells", areas where light is detected but that do not truly correspond to DNA sequencing events. The special signal processing configuration for Amplicons includes two additional well screening steps (see above) that help remove these false wells. This configuration is triggered by using the "runAnalysisPipeAmplicons" command line tool or by selecting "Full Processing for Amplicons" or "Signal Processing for Amplicons" from the GS Run Browser application.

3.2.2 Quality Filtering and Trimming

Following signal normalization and phase correction (and removal of ghost wells if appropriate), the signal processing step of the GS Run Processor application runs the resulting processed reads through a series of quality filters. Filters identify high quality reads or sections of reads that can be used for downstream data analysis (assembly, mapping, or rare variant analysis). There are two categories of quality filters. The first category assesses the quality of whole reads: these are the "Keypass filter", the "Dots filter", and the "Mixed filter"; failing any of these tests results in the rejection of the entire read. The second category assesses the distal end of the reads (end opposite the sequencing primer, represented by the later nucleotide flows of the sequencing Run): these are the "Signal Intensity filter", the "Primer filter", and "TrimBack Valley Filter"; these filters result in the trimming of any low quality flows at the 3' end of each read and, if the minimum threshold trimmed length of 84 nucleotide flows is reached, rejection of the read. Filtering occurs in "flowspace", as opposed to "nucleotide space".

A summary of the filtering results is stored within the .cwf file generated by the signal processing step of the pipeline. Specific "metrics" files (fully defined in section 4.1.2.4, below) can be extracted using the GS Reporter application. In particular, the number of reads that pass all filters is reported as "totalPassedFiltering", and the total of all bases associated with these reads is reported as filtered bases in "totalBases" in the output file "454BaseCallerMetrics". In addition, each individual filter has a corresponding output metric that is reported in the "454QualityFilterMetrics" file.

Filtering and trimming assesses the quality of the signals that comprise the reads, but no filtering is performed on the individual nucleotides basecalled for these reads. This means that certain nucleotides retained for basecalling, especially those near the end of long homopolymer stretches, may be included as high quality bases even though the Quality Score (Phred score) would be low for the individual base. For this reason, the basecalling algorithm of the signal processing step of the GS Run Processor application also calculates individual base Quality Scores (see section 3.2.2.2).

3.2.2.1 Quality Filters

3.2.2.1.1 The Keypass Filter

This filter verifies that the sequence in the well contains a valid key sequence, qualifying it as a valid sample library read or Control DNA read. The output metric is "numKeyPass".

3.2.2.1.2 The Dots Filter

A "dot" is an instance of 3 successive nucleotide flows that record no incorporation (or 4 negative flows if one considers the very first nucleotide flow of the Run). The Dots filter works in two phases. In its first phase, it finds the last flow with a positive signal; if that flow is before flow 84, the entire read is rejected. In the second phase, the Dots filter checks if more than 5% of the flows before the last positive flow in a read are dots; if this is the case, the entire read is considered low quality and is rejected. This can occur if the signal intensity in a well is generally low (*e.g.* due to poor chemistry, or if the well contains a mixture of DNA molecules resulting in poor signal intensity for any one flow). The output metric is "numDotFailed".

3.2.2.1.3 The Mixed Filter

The Mixed filter also operates in two phases. In its first phase, it tests to see if more than 70% of nucleotide flows before the last positive flow or before the 168th nucleotide flow, whichever comes first, appear to record an incorporation; if so, the read is considered low quality and is rejected because it is probably the result of simultaneously sequencing a mixture of different DNA molecules. In its second phase, the Mixed filter classifies the flow signals as having no incorporation, having one or more incorporations, or occurring in a "middle ground" (approximately between 0.45 and 0.75 incorporations, which is where the bulk of the signals for a mixed well occur); the read will be rejected if it has more than 20% of the flows in the middle ground, less than 30% of the flows above the middle ground, or less than 30% of the flows below the middle ground. Although it is possible for certain sequences to legitimately have greater than 70% positive flows, or have fewer than 30% positive flows or 30% negative flows, such instances are relatively rare and this filter primarily eliminates mixed reads. The output metric is "numMixedFailed".

3.2.2.1.4 The Signal Intensity Filter

The Signal Intensity filter defines a range of flow signals from 0 (no incorporation) to 1 (nucleotide incorporated), with the intensity range between 0.5 and 0.7 considered borderline positive (determined empirically using multiple datasets generated at 454 Life Sciences Corporation). If more than 3% of the flows in a read fall in the borderline region, the filter trims the read, starting from later flows and working backwards until either (1) the trimmed read has fewer than 3% borderline flows; or (2) the trimmed read is shorter than 84 flows (21 cycles, or approximately 50 nucleotides), whichever occurs first. Then, the filter trims the end of the read further to remove very poor quality ends (that may occur after a very high quality beginning); to do this, the filter checks a small window of flows just prior to the trim point, and if that area contains dots or borderline signals, the filter shifts the trim point to the first dot or borderline signal and then repeats the check on the new end-window. Trimmed reads shorter than 84 flows are rejected. The output metric for all the reads that fail (*i.e.* are rejected by) the Signal Intensity Filter is "numTrimmedTooShortQuality".

The Signal Intensity Filter assumes single base nucleotide signals. If a genome has many long homopolymer stretches, this approach may not be optimal: reads with high homopolymer content would likely pass this filter regardless of quality because the homopolymer signals will always be above the signal ranges defined for borderline calls.

3.2.2.1.5 The Primer Filter

The ends of all processed reads are scanned for similarity to Genome Sequencer System Adaptor sequences. Any flows corresponding to Adaptor sequences are trimmed from the read. If a trimmed read is shorter than 84 flows (21 cycles, or approximately 50 nucleotides), it is rejected. The output metric for these failed reads is "numTrimmedTooShortPrimer".

3.2.2.1.6 The TrimBack Valley Filter

The TrimBack Valley Filter first uses the trimmed flow signals to compute the signal distribution of all the reads that have passed all the other filters. Then, it identifies the valleys between the 0-mer and 1-mer, the 1-mer and 2-mer, and the 2-mer and 3-mer peaks. Finally, it scans the length of each read (in reverse order, from 3' to 5'), assigning each flow signal a score that attains a maximum when the flow value is at a valley minimum, and decreases to 0 when the flow signal is ± 0.1 away of the valley minimum. The score value represents the probability of the flow signal being an ambiguous, or a "valley flow", for the read. The cumulative sum of the scores are referred to the "valley score" for the read.

The filter then calculates the ratio of the valley scores to a set number of test flows in the reads. This ratio, representing the chance to see this number of "bad flows" over the entire read, is compared to the specified Valley Flow Threshold ratio, which is calculated as follows (see Figure 3–1 below for more information):

$$\text{Valley Flow Threshold ratio} = \frac{\text{vfBadFlowThreshold}}{\text{vfLastFlowToTest}}$$

The default values for calculating the threshold are 4 valley flows and 320 nucleotide test flows (*i.e.* 80 cycles of four successive nucleotide flows), which produces as a 1.25% Valley Flow Threshold ratio. These default Valley Filter parameters can be changed to either increase or decrease the stringency of the Valley Filter (see section 3.2.2.3).

Reads that exceed the specified Valley Flow Threshold ratio are trimmed back, starting from the 3'-end, to the nucleotide just before the first valley flow that exceeds the threshold. For example, with the default settings, a read from a sequencing Run carried out using the GS FLX Titanium chemistry that has incurred no previous trimming (full 800 flows before the Valley Filter) will be left intact if it has fewer than 10 valley flows (1.25%), but will be trimmed back from its 3'-end if it has 10 or more. (Again, trimmed reads shorter than 84 flows are rejected.) This procedure thus allows the retention of the "good part" of reads that have a lot of valley flows near their 3'-end.

The reads that fail (*i.e.* are rejected by) the TrimBack Valley filter are reported together with those that fail the Signal Intensity filter, in the output metric "numTrimmedTooShortQuality".

- The TrimBack functionality of the Valley Filter typically provides a significant number of additional high quality bases in the sequencing Run (~10–20%), while maintaining an average inferred read error rate close to or better than the original Valley Filter (without TrimBack), at the same stringency setting. In addition, the TrimBack filter generally produces a similar number or fewer contigs in assembly and mapping, compared to the non-TrimBack mode.

- However, use of the TrimBack mode results in an *apparent* ~10–20% decrease in read length, reported as Average Read Length for the Run, compared to the non-TrimBack mode. This is because the *additional* high quality reads passed by the TrimBack Valley Filter are shorter on average, since they are the product of trimming. Although most of the reads that would pass the non-TrimBack algorithm are unaffected by the TrimBack Valley Filter, this results in a decrease in the *average* read length for the whole Run.

- The TrimBack mode of the Valley Filter is not compatible with the processing of Run data for Amplicon Sequencing, and is not part of the "Full Processing for Amplicons" processing type.

- For data processing of General (*e.g.* Shotgun) and Paired End Runs, the TrimBack mode can also be turned off manually (*e.g.* with the `runAnalysisPipe` or `runAnalysisFilter` commands; see section 3.2.3). This is not normally recommended, but can be done by setting the doValleyFilterTrimBack parameter to "false" in the <*qualityFilter*> section of the <pipeline> section of the filterOnly template file (see section 3.2.2.3)

- The Valley Flow parameters apply to both the TrimBack and the non-TrimBack modes. However, when TrimBack is off, the Valley Filter counts the valley flows only over the test number of flows rather than over the full length of the read, so valley flows near the 3'-end of the read (beyond the test flows) will not be taken into consideration by the filter. Reads whose valley flows count is equal to or larger than the threshold over the test number of flows are simply rejected.

3.2.2.1.7 The Quality Score Trimming Filter

After the quality filters above have been applied to the data, the filtered and trimmed reads are processed through basecalling and computing the individual base quality scores (section 3.2.2.2). After quality scores are computed, an additional trimming is applied on the called nucleotide sequences, based on their scoring.

Trimming with the quality scores first scans a window of nucleotide sequence from 3'end of the reads (the default window size is 10 nucleotides), and computes the average quality score in the nucleotide window. If the average score is lower than Q20, the last nucleotide sequence is trimmed at the 3', the window moves forward (3' to 5'), and the average score is recomputed for the new (10-base) nucleotide window. The trimming stops when the average score satisfies the "Q20 test".

The remaining nucleotide sequence is considered high quality (good for use in Assembly, Mapping, or other processing), and output as called bases to the SFF files (and FASTA files, via the GS Reporter application).

3.2.2.2 Individual Base Quality Scores

It is important to note that sequenced bases are not filtered individually; rather, all signals contained in the reads that passed filtering and trimming are considered high quality (good for use in Assembly, Mapping, or other processing), and output as called bases to the CWF and SFF files (and optionally to a FASTA format file, via the GS Reporter application).

Quality scores for individual called bases are determined by a method developed in collaboration with the Broad Institute, whereby the methodology described by Ewing and Green (*Genome Research*, 8: 186-194, 1998) for the creation of quality scores as part of the Phred basecalling algorithm is applied to generating quality scores for 454 reads. The quality scores computed for each called base are written to the CWF and SFF files (and optionally to a file paralleling the basecall FASTA file). Briefly, the method compares the properties of each base's flowgram signals against properties that have been found to correlate with accurate and/or error-prone signal information, using training sets of read data. A multivariate analysis of those properties determines the sets of property values that best describe "bins" of basecalls, then assigns the training set accuracy rates of the basecalls in each bin as a quality score using the following scale:

$$Q = -10 \log_{10} (\text{accuracy})$$

Then, when the basecalling of a new read is performed, the trace properties for each read are used to determine the bin in which each base falls, and the quality score associated with that bin is assigned to the basecall. For 454 reads, the property metrics based on the following characteristics are used for the training and the quality score generation:

- Base position of the base in the read sequence
- Flow signal intensity for the flow where the base is called
- Flow signal intensity one cycle before and/or one cycle after the flow where the base is called
- Local variation of the flowgram signals in a window surrounding the flow where the base is called (*i.e.*, how far are those signals from the ideal 0.0, 1.0, 2.0, 3.0, … signals). This provides an estimate of the homopolymer accuracy.
- Overall "separation" of the flowgram signals (*i.e.*, how greatly separated are the 0-mer signals from 1-mer signals)

Training sets using multiple datasets from different sample Runs were used for training the quality scores, and lookup tables were generated from those datasets. Separate training was performed for GS 20 chemistry, GS FLX standard chemistry, and GS FLX Titanium chemistry data, to account for the slightly different error tendencies of the three chemistries. These lookup tables are used to generate the base quality scores.

3.2.2.3 Modifying the Stringency of Quality Filtering

Certain parameters of the TrimBack Valley filter (section 3.2.2.1.6) or of the Quality Score Trimming filter (section 3.2.2.1.7) can be adjusted to control the overall stringency of the read quality filtering. Increasing the stringency will tend to improve the average accuracy of the filtered reads, but will yield fewer reads; while decreasing the stringency will tend to increase the number of passed filter high quality reads at the expense of overall accuracy.

The TrimBack Valley filter parameters that can be used for this are:

- whether the read trimming function of the TrimBack Valley filter should be active;
- the number of test flows used to calculate the Valley Flow Threshold ratio: using more flows gives more chance of reaching the rejection threshold, and hence increases the stringency;
- the number of valley flows used as rejection threshold: again, a lower threshold is easier to meet, and hence provides higher stringency;
- the maximum percentage of flows that fail the TrimBack Valley filter before the read is rejected entirely; and
- the minimum trimmed length of a read below which the read is rejected entirely.

The Quality Score Trimming filter parameters that can be used for this are:

- whether the Quality Score Trimming filter should be active;
- the size of the window in which the test average accuracy must be maintained: requiring that a given accuracy be maintained in a larger window is more stringent; and
- the average accuracy in the window below which the read will be trimmed by one nucleotide: a higher the average accuracy requirement (smaller value for this parameter) is more stringent.

This section provides instructions for modifying the stringency of quality filtering, for a sequencing Run whose data has already been processed, by changing these Valley Filter or Quality Score Trimming filter parameters. These are stored in the "pipeline configuration file". For off-instrument computing resources (DataRig or cluster), templates for this file are stored in /etc/gsRunProcessor/. (For on-instrument signal processing, these are stored in $RIGDIR/config/pipelines, but performing signal processing on-instrument is not normally recommended.)

1. Identify the Data Processing folder ('D_') of the sequencing Run whose reads you want to re-filter. There may be multiple 'D_' folders in the Run ('R_') folder, if the data has already been processed more than once; in that case, make sure to identify the one you want to use as the basis for your filter stringency adjustment.

2. Generate a template file in the current directory by typing the following command:

   ```
   gsRunProcessor --template=filterOnly > filterTemplate.xml
   ```

   This will create a template file in the D_ directory. There are additional templates for paired end and amplicons experiments, which can be generated by using the words "filterOnlyPairedEnd" and "filterOnlyAmplicons" in the command above. The filename to the right of the ">" symbol can be selected by the user. The name of the file will be needed in steps 3 and 15.

3. Open the file with a text editor. An easy-to-use text editor called "nedit" is present on the Genome Sequencer FLX Instrument. To use nedit to edit the file, type the following command:

   ```
   nedit filterTemplate.xml
   ```

▶▶▶

4 Within the template, scroll down to the <qualityFilter> and <baseCaller> sections (Figure 3–1). These sections contain all the user-adjustable Quality Filter parameters that govern the filtering of high quality reads for the sequencing Run. As you do your adjustments, in the next steps, bear in mind that:

a. increasing a stringency setting will reduce the number of filtered high quality reads by eliminating those reads from among the previously filtered reads that have the lowest quality. This will result in a higher average accuracy of the filtered high quality reads; while

b. decreasing a stringency setting will reduce overall accuracy by allowing lower quality reads to pass through the quality filter, but increase the number of passed filter high quality reads.

```
<pipeline>
          <!-- Metrics generator.  There are no user modifiable parameters here. -->
          <metricsGenerator/>

          <!-- Quality filter.-->
          <qualityFilter>

                    <doValleyFilterTrimBack>true</doValleyFilterTrimBack>
                    <vfBadFlowThreshold>4</vfBadFlowThreshold>
                    <vfLastFlowToTest>320</vfLastFlowToTest>
                    <vfMaxFailedPercent>100</vfMaxFailedPercent>
                    <vfTrimBackMinimumLength>84</vfTrimBackMinimumLength>

          </qualityFilter>

          <baseCaller>

                    <doQscoreWindowTrim>true</ doQscoreWindowTrim>
                    <nucleotideQscoreWindowTrim>10</nucleotideQscoreWindowTrim>
                    <errorQscoreWindowTrim>0.01</errorQscoreWindowTrim>

          </baseCaller>
</pipeline>
```

Figure 3–1: The <pipeline> section of the filterOnly pipeline template. It is important that the <metricsGenerator/> <qualityFilter> and <baseCaller> sections of the pipeline stay in the same order. Only edit the parameters highlighted above.

5 The first parameter that can be changed is “doValleyFilterTrimBack”. The default value is “true” for the General (shotgun) and Paired End processing types (but “false” for Amplicons; trimming would be incompatible with Amplicon experiments). Changing this to “false” turns off the trimming function of the TrimBack Valley filter and turns it into a pass-or-fail filter (using the other filter’s parameters, below). In a sense, the “false” value might be considered more stringent because reads that fail the test are rejected entirely rather than just trimmed. On the other hand, the value “true” rejects only the low quality parts of the reads, resulting in a higher overall quality (and in the salvage of good parts of many reads).

6 The second parameter that can be changed is “vfLastFlowToTest”. (If the TrimBack mode is turned off, this value represents the number of flows over which the reads are monitored for the presence of valley flows, in a pass-or-fail test.) The default value of “320” can be either increased or decreased, to make the filter more stringent or less stringent, respectively. The maximum value for this parameter is equal to the number of nucleotide flows in the Run (most stringent), and the minimum value is “0” (least stringent; this would effectively turn off the Valley Filter altogether). The following settings for either increasing or decreasing stringency are recommended:

For increased stringency:	<vfLastFlowToTest>400</vfLastFlowToTest>
For decreased stringency:	<vfLastFlowToTest>168</vfLastFlowToTest>

▶▶▶

7 The third parameter that can be changed is "vfBadFlowThreshold". (If the TrimBack mode is turned off, this value represents the threshold number of valley flows within the flows monitored, at which a read is rejected due to its low quality.) The default value of "4" can be either decreased or increased, to make the filter more stringent or less stringent, respectively (opposite of the vfLastFlowToTest parameter). The following settings for either increasing or decreasing stringency are recommended:

Highest stringency	<vfBadFlowThreshold>2</vfBadFlowThreshold>
Higher stringency	<vfBadFlowThreshold>3</vfBadFlowThreshold>
Default stringency	<vfBadFlowThreshold>4</vfBadFlowThreshold>
Lower stringency	<vfBadFlowThreshold>5</vfBadFlowThreshold>
Lowest stringency	<vfBadFlowThreshold>6</vfBadFlowThreshold>

8 The fourth parameter that can be changed is "vfMaxFailedPercent". The default value of "100" can be decreased to make the filter more stringent. When the percentage of "valley" flows in a read reach this percentage, the read is rejected entirely. This parameter can be used to fine tune the failed percent. A range between "90" and "100" is recommended.

9 The fifth parameter that can be changed is "vfTrimBackMinimumLength". The default value of "84" can be either decreased or increased, to make the filter more or less sensitive to short reads, respectively. When a read is trimmed such that it would be shorter than the value of this parameter, the read is rejected entirely. Users may set this parameter to a lower number if they are attempting to sequence very short reads.

10 The sixth parameter that can be changed is "doQscoreWindowTrim". The default value is "true". Changing this to "false" turns off the Quality Score Trimming filter, which is less stringent.

11 The seventh parameter that can be changed is "nucleotideQscoreWindowTrim". The default value of "10" can be either increased or decreased, to make the filter more stringent or less stringent, respectively. The following settings for either increasing or decreasing stringency are recommended:

For increased stringency:
<nucleotideQscoreWindowTrim>20</nucleotideQscoreWindowTrim>

For decreased stringency:
<nucleotideQscoreWindowTrim>5</nucleotideQscoreWindowTrim>

12 The eighth and last parameter that can be changed is "errorQscoreWindowTrim". The default value of "0.01" represents a Q20 accuracy, and can be either decreased or increased, to make the filter more stringent or less stringent, respectively. However, the value cannot be zero (which would mean an infinite accuracy) or a negative number The following settings for either increasing or decreasing stringency are recommended:

For increased stringency:
<errorQscoreWindowTrim>0.005</errorQscoreWindowTrim>

For decreased stringency:
<errorQscoreWindowTrim>0.02</errorQscoreWindowTrim>

13 When you have made your changes, save them and exit the text editor. For example, the nedit editor has a File pull-down menu with standard file saving and renaming options, as well as the close and exit selections for closing a file or exiting the application.

14 Change directories to exit the Data Processing folder, by typing the command:

```
cd ..
```

15 Finally, run the modified processing script on data from the current Data Processing folder, using the following command (see section 3.2.3):

```
runAnalysisFilter –pipe=<name of Data Processing folder>/filterTemplate.xml
```

A new Data Processing directory will be created, containing the new results. It is not necessary to generate the template file for each new re-processing. If a user modifies a template file that generates satisfactory results, it can be used for subsequent rounds of data processing by specifying the full path of the template file as the parameter to the --pipe command in the runAnalysisFilter command (section 3.2.3). Users may copy useful pipeline files *e.g.* to their home directories to make them easy to locate in the future.

3.2.3 Launching the Signal Processing Application

There are three commands that can be used to launch the signal processing step of the GS Run Processor application. These are:

```
runAnalysisPipe [options] SOURCE_DIRECTORY
    runAnalysisPipePairedEnd [options] SOURCE_DIRECTORY
    runAnalysisPipeAmplicons [options] SOURCE_DIRECTORY
```

The Source Directory can be either an 'R_' directory or a 'D_' directory (see section 1.3). If an 'R_' directory is specified, the runAnalysisPipe command will run both the image processing and signal processing steps on the data set. If a 'D_' directory is specified, the command will run only the signal processing step. This requires that the 'D_' directory contain only image processing results; if the data present in a 'D_' directory has also been signal processed, the runAnalysisPipe will produce the following warning message (note that this is only a warning; processing will still proceed):

```
This data set has already been corrected. Further processing
will likely distort the results. Ensure you provide a directory
containing uncorrected data. Directories containing unprocessed
data usually have an “imageProcessingOnly” suffix. Proceeding,
but the results will probably be incorrect.
```

Note on Software v. 2.0: Unlike previous version of the Genome Sequencer System software, the runAnalysisPipe command needs a pipeline file to be identified. See section 3.2.2.3 for more details on how to create a custom pipeline.

An additional command can be used to launch solely the filter portion of the signal processing step. This command requires that the data already be processed through a previous iteration of the signal processing step. The command is:

```
runAnalysisFilter --pipe= datadirectory
```

Command	Description
runAnalysisPipe, runAnalysisPipePairedEnd, or runAnalysisPipeAmplicons	These commands run the complete set of signal processing algorithms, beginning with either raw images or the output of the image processing step (the .cwf files). The outputs of these commands are: ▶ Composite Wells Format (CWF) files, containing the corrected flowgrams and Run metrics. ▶ Standard Flowgram Format (SFF) files, which will be used as input to the data analysis applications. This command executes the same set of algorithms that are performed when the “Full Processing”, “Full Processing for Paired End”, or “Full Processing for Amplicons” processing types are included in a sequencing Run (see the *GS FLX Titanium Sequencing Method Manual.*) These include the invocation of the GS Reporter application, whose default out put includes: ▶ FASTA and the corresponding quality files ▶ Various metrics files
runAnalysisFilter	This command executes only the filtering, trimming and base-calling parts of the processing. It allows the rapid generation of flowgrams and basecalled sequences from already pro-cessed data sets.

Argument	Description
SOURCE_DIRECTORY	Path to an ‘R_’ or a ‘D_’ directory. These are created by the GS Sequencer application, or by the image processing step of the GS Run Processor application, respectively (see sec-tion 1.3).

Option	Description
--pipe	Specifies a custom pipeline script to be used to process the run
--reg	Only process specified regions. Regions can be specified as a range or as a comma-separated list (“1-4” or “1,2,5,9”)

3.2.4 Signal Processing Step Output

The signal processing commands described in the previous section can also be invoked using the reanalysis function of the GS Run Browser application, on the Genome Sequencer FLX Instrument or on a DataRig or cluster.

The signal processing step always generates a separate D_directory from the one produced by the image processing step. This application generates 2 general categories of results:

- processed "*region*.cwf" data, similar to but more refined than the output of the image processing application, deposited in the "regions" sub-folder
- the collected sequence trace data, comprising the processed read flowgrams, basecalls and per-base quality scores, stored as SFF files

In addition, if the GS Reporter application is run, the output can include:

- various "metrics" files from the main sub-steps of the signal processing for the Run as a whole, and optionally at the region level, useful in troubleshooting
- the basecalled reads, per region and segregated for each sequencing key (if applicable), with associated per-base quality scores stored as FASTA files

The exact set of files created by the signal processing step of the GS Run Processor application (either pipeline) is as shown in Table 3–3.

File Name Structure	Definition
region.cwf	Located in the "regions" folder, these files contain the fully processed flow-by-flow signal data for all the wells in any given PicoTiterPlate loading region, plus all the metrics and meta data information about the sequencing Run. These are written over the corresponding preliminary files generated by the image processing step of the GS Run Processor application (one file per region).
uaccnoRegion.sff	Standard Flowgram Format (SFF) files containing the sequence trace data for the high quality reads.

Table 3–3: Files generated by the signal processing step of the GS Run Processor application. Words in *italics* are generic; real file names would have the actual name of the 'region' or the string identifying the Run and region using the universal accession naming convention (see section 13.3.7).

For a sequencing Run with 2 analysis regions where the region names are "1" and "2", and with a universal accession Run prefix C3U5GNB, the files produced will have the following names:

- regions (folder)
  - 1.cwf
  - 2.cwf
- sff (folder)
  - C3U5GNB01.sff
  - C3U5GNB02.sff
- gsRunProcessor.log
- gsRunProcessor_err.log

Full data processing default output: By default all the "Full Processing" data processing types that can be chosen during set up of a sequencing Run (see section 1.2), as well as the three corresponding runAnalysisPipe commands that run the signal processing algorithms (this section) evoke the GS Reporter application (section 4) to generate a number of human-readable data and metrics files describing the Run results. For a sequencing Run with 2 analysis regions where the region names are "1" and "2", 2 different keys (with sequences "TCAG" and "CATG", where "TCAG" is used for the sample library and "CATG" is used for the Control DNA Beads1), and with a universal accession Run prefix C3U5GNB, the full output of this default setting would be as follows:

Output File / Folder	Generated by
▶ gsRunProcessor.log	GS Run Processor
▶ gsRunProcessor_err.log	GS Run Processor
▶ regions (folder) ▶ 1.cwf ▶ 2.cwf	Image processing step, then updated by the signal processing step of the GS Run Processor
▶ 454QualityFilterMetrics.txt	GS Reporter
▶ 454QualityFilterMetrics.csv	GS Reporter
▶ 454BaseCallerMetrics.txt	GS Reporter
▶ 454BaseCallerMetrics.csv	GS Reporter
▶ 454RuntimeMetricsAll.txt	GS Reporter
▶ 454RuntimeMetricsAll.csv	GS Reporter
▶ 1.ATG.454Reads.fna	GS Reporter
▶ 1.ATG.454Reads.qual	GS Reporter
▶ 1. TCA.454Reads.fna	GS Reporter
▶ 1. TCA.454Reads.qual	GS Reporter
▶ 2.ATG.454Reads.fna	GS Reporter
▶ 2.ATG.454Reads.qual	GS Reporter
▶ 2. TCA.454Reads.fna	GS Reporter
▶ 2. TCA.454Reads.qual	GS Reporter
▶ sff (folder) ▶ C3U5GNB01.sff ▶ C3U5GNB02.sff	Signal processing step of the GS Run Processor

[1] **The sequencing keys for Control DNA reads are ATGC or CATG, for GS FLX standard and GS FLX Titanium chemistries, respectively.**

3.2.4.1 *region*.cwf

The signal processing step of the GS Run Processor application reads the uncorrected flowgrams from the Composite Wells Format (CWF) files (one per region of the PicoTiterPlate device) generated during the image processing step, performs a number of corrections, and writes the corrected flowgrams into a new set of CWF files, in a separate D_ directory. The called bases, quality scores, sequence trim points, and runtime metrics are written into the CWF file package as well. This data is used by the GS Reporter application to generate output artifacts such as the FASTA files and the 454 text (.txt) and comma-separated value (.csv) reports (see section 13.3). The GS Reporter can also extract information from the CWF files to generate legacy compatibility files. The format of the CWF files is described in section 13.3.1.

3.2.4.2 *uaccnoRegion*.sff

This set of NCBI-compatible Standard Flowgram Format (or SFF) files contain the read flowgram (trace) data, basecalls and quality scores for all the high quality (passed all filters) sample library reads. For a general description of the Standard Flowgram Format, see section 13.3.8. For a description of the universal accession naming convention, which is used to name the SFF files, see section 13.3.7. For a text sample of the information stored in an SFF file for a read, see the output of the sffinfo command in section 13.4.14.

3.2.4.3 *gsRunProcessor.log* and *gsRunProcessor_err*.log

This pair of files is in the same format as the corresponding files produced by the image processing step of the GS Run Processor application (see section 3.1.2.3).

4. GS Reporter Application

The GS Reporter is a separate application that can extract Run metrics and other Run information from the CWF files produced by the GS Run Processor, and generate various human-readable files which can be used to examine the results of a sequencing Run. These GS Reporter output files are modeled after the various files that were output by previous versions of the data processing software: FASTA and .qual files for the sequences and corresponding quality scores, various metrics files in .txt and .csv format, *etc.* The GS Reporter can even generate files that are no longer part of the normal data processing output, such as the .well files, by using the '--legacy' option.

The GS Reporter application is evoked by default by all the "Full Processing" data processing types that can be chosen during set up of a sequencing Run (see section 1.2), as well as by the three corresponding runAnalysisPipe commands (section 3.2.3). The output of this default setting is identified by shaded rows in Table 4–1, below.

4.1 Launching the GS Reporter Application

The gsReporter command line has the following form:

```
    gsReporter [options] INPUT_CWF [INPUT_CWF ...]
```

Command	**Description**
gsReporter	Application that extracts results from the .cwf file of a sequencing Run, and presents them in the form of data and metrics files.

Argument	**Description**
INPUT_CWF [INPUT_CWF ...]	Path to one or more .cwf files that contain the processed sequencing Run results from which to prepare a report.

Many options are also available to determine what files will be produced by the gsReporter command, and where they will be output, as listed in Table 4–1. Multiple options may be specified on a single line. Note that if an option is specified, the gsReporter will attempt to generate the corresponding output file even if the option does not make sense in the context. For example, using the --fna option in the creation of a report on data that was subject to only image processing will result in an empty .fna file. In this case a warning will be generated to the console. See section 4.1.2 for a description of the files that can be generated.

Option	Description
--console	Makes the command write its results to standard output. This is useful in cases where the gsReporter command's output will be used as input to another program.
--out	Allows the user to specify an alternate (non-default) output location for its results. (By default, the GS Reporter application attempts to write its output files to the locations they would have been placed by earlier versions of the software.)
--dump	Dumps all XML in the CWF as a single document.
--info	Shows summary information about the CWF file.
--legacy	Simulate the output of a GS FLX sequencing Run processed with pre-2.0 software.
-a, --all	Generates all output files.
--fna	Generates one FNA file per key per region.
--qual	Generates one qual file per key per region.
--meta	Extracts the meta information about a region's data as an XML file.
--metrics	Extracts the metrics information about a region's data as an XML file.
--wells	Generates a .wells file per region.
--cfValues	Generates one cfValues file per region. This is a binary file that contains information about the CAFIE correction data. This file is used by the gsSupportTool for run diagnostics.
--454RuntimeMetricsAll.txt	Generates one 454RuntimeMetricsAll.txt file for the Run.
--454RuntimeMetricsAll.csv	Generates one a 454RuntimeMetricsAll.csv file for the Run.
--454BaseCallerMetrics.txt	Generates one 454BaseCallerMetrics.txt file for the Run.
--454BaseCallerMetrics.csv	Generates one 454BaseCallerMetrics.csv file for the Run.
--454RuntimeMetrics.txt	Generates one 454RuntimeMetrics.txt per key per region.
--454RuntimeMetrics.csv	Generates one 454RuntimeMetrics.csv per key per region.
--454QualityFilterMetrics.txt	Generates one 454QualityFilterMetrics.txt file for the Run.
--454QualityFilterMetrics.csv	Generates one 454QualityFilterMetrics.csv file for the Run.
--cafieMetrics	Generates one cafieMetrics.csv file per region.
--droopEstimate	Generates one droopEstimate file per region.
--trimInfo	Generates one trimInfo file per region.
--xy	Generates a text list of well locations.
--csv	Generates a file with the flows written out as comma separated values.
--analysisParms.parse	Generates an approximation of a pre-2.0 analysisParms. parse.
--revisedRegions.parse	Generates a single revisedRegions.parse file.

Table 4–1: The various options available to specify the files to be generated by the GS Reporter application. The options shaded are part of the default configurations.

4.2 GS Reporter Application Output Files

The files that are output by the GS Reporter application are directly controlled by the options used on the command line (section 4.1.1). The main ones are listed in Table 4–2.

By default, the GS Reporter application attempts to write its output files to the locations they would have been placed by earlier versions of the software. This can be overridden by the use of the '--console' or the '--out' options (section 4.1.1).

Note on Software v. 2.0:

- The data processing pipeline no longer creates .wells files by default. These, and most other output files from pre-2.0 releases of the software, can be created by the GS Reporter application. To create all the pre-2.0 files in their proper locations, set the GS Reporter option at the end of the pipeline scripts to '--legacy'. Note that wells files from Runs carried out using the GS FLX Titanium chemistry can be very large (over 2.4 GB) and may not be accessible by software without "Large File Support" (LFS).
- Nonetheless, the GS Run Processor can read .wells files from previous versions of the software. In order to determine the correct context for the flowgrams, the program searches for an analysisParms.parse file and a .trimInfo file in the directory containing the input .wells files and its parent.

File Name Structure	Definition
region.wells	Located in the "regions" folder, these files contain the fully processed flow-by-flow signal data for all the wells in any given loading region of the PicoTiter-Plate device and are written over the corresponding preliminary files initially generated by the Image Processing software (one file per region).
region.trimInfo	These files parallel the *region*.wells files, above, and list the pass/fail and trim point results from the 5 read quality filters, for the reads of each well in the *region*.wells files. Together with the data in the *region*.wells files, this defines the well "flowgrams".
region.wells. *key*.454RuntimeMetrics.txt	This set of files provides various runtime metrics, including performance metrics for Control DNA Beads if they are present in the Run; one file for each specific region and key combination.
454QualityFilterMetrics.txt	This file provides various quality filter metrics, including the analysis name, number of raw wells, keypass wells, dot wells, mixed wells, filtered read information, and trimmed read information.
454BaseCallerMetrics.txt	This file provides various basecaller metrics, including the number of high quality filtered reads, average read length, and information by region.
454RuntimeMetricsAll.txt	This file provides various runtime metrics, including performance metrics for Control DNA Beads if they are contained in the Run, for all the regions and for all the keys indicated.
region.key.454Reads.fna	FASTA file of the basecalled reads, after full processing including filtering and trimming, for each specific region and each specific key (with the leading sequence key omitted).
region.key.454Reads.qual	Corresponding quality score values for each base in the reads in the *region.key*.454Reads.fna file.

Table 4–2: Files generated by the Signal Processing application (version of the data processing software anterior to v. 2.0. Words in *italics* are generic; real file names would have the actual name of the 'region', the first three nucleotides of the 'key' sequence, or the string identifying the Run and region using the universal accession naming convention (see section 13.3.7). The four *.txt files listed also have a companion *.csv file containing the same information, in the comma-separated values file format that can be imported into a spreadsheet application such as Microsoft Excel.

For a sequencing Run with 2 analysis regions where the region names are "1" and "2", 2 different keys (with sequences "TCAG" and "CATG", where "TCAG" is used for the sample library and "CATG" is used for the Control DNA Beads[1]), and with a universal accession Run prefix C3U5GNB, the main legacy files that can be produced by the GS Reporter will have the following names:

- regions (folder)
  - 1.wells
  - 2.wells
  - 1.trimInfo
  - 2.trimInfo
  - 1.wells.ATG.454RuntimeMetrics.txt
  - 1.wells.ATG.454RuntimeMetrics.csv
  - 1.wells.CAT.454RuntimeMetrics.txt
  - 1.wells.CAT.454RuntimeMetrics.csv
  - 2.wells.ATG.454RuntimeMetrics.txt
  - 2.wells.ATG.454RuntimeMetrics.csv
  - 2.wells.CAT.454RuntimeMetrics.txt
  - 2.wells.CAT.454RuntimeMetrics.csv
- 454QualityFilterMetrics.txt
- 454QualityFilterMetrics.csv
- 454BaseCallerMetrics.txt
- 454BaseCallerMetrics.csv
- 454RuntimeMetricsAll.txt
- 454RuntimeMetricsAll.csv
- 1.ATG.454Reads.fna
- 1.ATG.454Reads.qual
- 1. CAT.454Reads.fna
- 1. CAT.454Reads.qual
- 2.ATG.454Reads.fna
- 2.ATG.454Reads.qual
- 2. CAT.454Reads.fna
- 2. CAT.454Reads.qual

[1] The sequencing keys for Control DNA reads are ATGC or CATG, for GS FLX standard and GS FLX Titanium chemistries, respectively.

As an aid for users familiar with the files output by the previous versions of the data processing software, Table 4–3 lists all the files that can be output by the data processing applications of the Genome Sequencer System, and how they can be generated (or not) in the software v. 2.0.

Organization of a Data Processing Directory ("D_")							
File name	Generated By gsRunProcessor During Processing	New for 2.0	Generated by Default	Can Be Generated Directly By gsReporter From CWF Files	Generated By gsReporter's "--legacy" Option	No Longer Generated	Notes
gsRunProcessor.log	✓	✓	✓				
gsRunProcessor.err	✓	✓	✓				1
454DataProcessingDir.xml	✓	✓	✓				2
X.KEY.454Reads.fna			✓	✓	✓		3, 7
X.KEY.454Reads.qual			✓	✓	✓		3,7
454BaseCallerMetrics.csv			✓	✓	✓		3, 4, 7
454BaseCallerMetrics.txt			✓	✓	✓		3, 7
454QualityFilterMetrics.csv			✓	✓	✓		3, 4, 7
454QualityFilterMetrics.txt			✓	✓	✓		3, 7
454RuntimeMetricsAll.csv			✓	✓	✓		3, 4, 7
454RuntimeMetricsAll.txt			✓	✓	✓		3, 7
analysisParms.parse				✓	✓		3, 5
revisedRegions.parse				✓	✓		4
454BaseCallerThresholds.txt						✓	
error.baseCaller2						✓	
error.bbcSelfTrain						✓	
error.cafieCorrection						✓	
regions/	✓		✓				
X.cwf	✓	✓	✓				
X.metrics.xml		✓		✓			
X.meta.xml		✓		✓			
X.processingHistory.xml		✓		✓			
X.wells				✓	✓		
X.wells.KEY.454RuntimeMetrics.csv				✓	✓		3, 4, 7
X.wells.KEY.454RuntimeMetrics.txt				✓	✓		3, 7
X.wells.cafieMetrics.csv				✓	✓		3, 4, 7
X.wells.cfValues				✓	✓		3, 7
X.wells.droopEstimate				✓	✓		3, 7
X.wells.incValues				✓	✓		3, 7
X.wells.mleCorrectionInfo				✓	✓		3, 7
X.wells.trimInfo				✓	✓		3, 7
sff/	✓		✓				
ACCNOPREX.sff	✓		✓				
control-sff/	✓	✓					6
ACCNOPRE0X.sff	✓	✓					6
failed-sff/	✓	✓					6
ACCNOPRE0X.sff	✓	✓					6

Table 4–3: Organization, in the 'D_' directory, of the files output by the data processing algorithms of the Genome Sequencer System software, and the application that can generate them. The following codes are used in the file names: X is the region number, ØX is the zero padded region number, KEY is the 3 or 4 letter sequencing key, ACCNOPRE is the accession number prefix.

Notes:

1 - This file is only generated if there are warnings or errors generated during processing. Otherwise, it will not be created.

2 - This file is only created after the base calling step is run and signifies to the data analysis software that the sff directory contains files suitable for processing.

3- The default generation of these files can be controlled by adjusting the gsReporter options in the pipeline configuration files.

4 - These files are generated for legacy purposes only. It is recommended that new applications use X.metrics.xml and X.meta.xml extracted from the CWF file via gsRunProcessor for report generating purposes.

5 - This file is deprecated and generated for legacy purposes only. X.processingHistory.xml and X.meta.xml are the canonical record of parameters used to process a Run and a Run's metadata, respectively.

6 - The sff files containing control keys and failed Runs are not generated by default, but can be triggered by adjusting the pipeline configuration file.

7 - These files may only be generated after all processing is complete. Specifically, the data to create these files are not available after the image processing only step of gsRunProcessor.

4.2.1 *region.*wells

The *region.*wells legacy files contain the signal data for all nucleotide and PPi flows of the Run, from all the active wells of the PicoTiterPlate device identified in the sequencing Run. The signals have been corrected for neighboring well cross-talk effects, for potential "out-of phase" errors (incomplete extension and carry-forward), signal droop, *etc.* This constitutes the final signal data in pre-2.0 software data processing.

To generate the "read flowgrams", this processed signal data is threaded for each well across all the flows of the Run, and then further restricted by the quality filters (whose results are contained in the *region.*trimInfo files) to ensure that only high quality reads/signals are incorporated into flowgrams, and input into the data analysis applications. For a general description of the ".wells" file format, see section 13.3.4. As these are binary-format files, an example cannot be provided in this text-based document.

4.2.2 *region.*trimInfo

In the pre-2.0 software, the results of the filtering and trimming operations are stored in the *region*.trimInfo files, presented as a list of trimming endpoints for all the wells (reads) in the corresponding *region*.wells files (the two file sets are parallel, well by well). Each read endpoint is written in the following form:

```
    unsigned short trimEnd;
```

- If the read was rejected by any of the five pre-2.0 filters, trimEnd takes the value 0.
- If the read passed all 5 filters, trimEnd takes the value of the last flow before the trimming point, using a 0-based index of the flows (*i.e.* the flows are numbered from 0 to "numFlows-1", and the trimming point is between flow "trimEnd" and "trimEnd+1"); if the trimming would result in a read of less than 84 flows, the read fails (*i.e.* trimEnd = 0).
  - Note: The trimEnd value gives the last flow before the trimming point, *counting only the nucleotide flows*. Since the well record includes signals for PPi flows, the correct method of identifying the well record flow corresponding to the trim point is to scan the well record flows, skipping PPi flows and counting non-PPi flows, to the point where the counter equals trimEnd.

This information restricts the data present in the *region.*wells files used to form the read flowgrams, which constitute the input for data analysis processing; or for the basecalling of the reads. For example, only the wells with a positive trimEnd value appear in the 454Reads.fna and 454Reads.qual file, and the sequence in the FASTA file starts with the first character after the key and ends with the character called in the trimEnd flow. As these are binary-format files, an example cannot be provided in this text-based document.

4.2.3 *region.wells.key.*454RuntimeMetrics.txt

This set of files provide basic Run and signal metrics, as a separate file for each specific region and sequencing key combination, including performance metrics for Control DNA Beads if they are present in the Run. These metrics are the same as those presented for the whole Run (all regions, all keys) in the 454RuntimeMetricsAll.txt file. See section 4.1.2.6 for details.

The four *.txt files output by the Signal Processing application also have companion *.csv files containing the same information, in the comma-separated values file format convenient for importing data into a spreadsheet.

4.2.4 454QualityFilterMetrics.txt

The 454QualityFilterMetrics.txt is a 454 parser file that provides the results of all the quality filters applied to the Run, for each region of the PicoTiterPlate device and each sequencing key. An equivalent *.csv file contains the same information in a format suitable for importing into a spreadsheet. Below is a description of all the sections present in this file, with their named groups and keywords. For a general description of the 454 parser file format, see section 13.3.2. For a sample 454QualityFilterMetrics.txt file, see section 13.4.1.

1. Comment Header – The comment header contains the version of the software, the full name of the sequencing Run ('R_'), and the name of the Analysis ('D_'). The header also includes the date and time the file was created in the form: YYYY/MM/DD HH: MM:SS.

2. runConditions group – This group contains the date, the unique name of the Run, and the name of the data analysis. The keyword parameters contained in this group are:

a. dateOfFile – The date/time when the file was created, in the form:

YYYY/MM/DD hh:mm

…where:

YYYY	- four digit year
MM	- two digit month (01 to 12)
DD	- two digit day (01 to 31 depending on month)
hh	- two digit hour (00 to 23)
mm	- two digit minutes (00 to 59)

b. rigRunName – The unique name for this Run. The full Run name is constructed by concatenating multiple pieces of information separated by "_" characters, and put in the form:

R_YYYY_MM_DD_hh_mm_rigName_userName_freeForm

…where:

R	- stands for "Run" and is always the first character of the Run name
YYYY	- four digit year of when the Run was executed
MM	- two digit month (01 to 12) of when the Run was executed
DD	- two digit day (01 to 31 depending on month) of when the Run was executed
hh	- two digit hour (00 to 23) of when the Run was executed
mm	- two digit minute (00 to 59) of when the Run was executed
rigName	- network host name of the computer in the Genome Sequencer FLX Instrument used to execute this Run
userName	- name of the user who executed the Run (from the login username)
freeForm	- any alpha numeric string ("Unique Run name") that was entered by the user

▶▶▶

2 c. analysisName – The unique name for the data analysis that was executed. The full data analysis name is constructed by concatenating multiple pieces of information separated by "_" characters, and put in the form:

D_YYYY_MM_DD_hh_mm_rigName_userName_freeForm

…where:

D	– stands for "Data Analysis" and is always the first character of the analysis name
YYYY	– four digit year of when the data analysis was executed
MM	– two digit month (01 to 12) of when the data analysis was executed
DD	– two digit day (01 to 31 depending on month) of when the data analysis was executed
hh	– two digit hour (00 to 23) of when the analysis was executed
mm	– two digit minute (00 to 59) of when the analysis was executed
rigName	– network host name of the computer on which the data analysis was executed (*e.g.* a Genome Sequencer FLX Instrument) (not included if the analysis was performed with the off-instrument version of the software)
userName	– name of the user who executed the data analysis (from the login username) (not included if the analysis was performed with the off-instrument version of the software)

3 region group – There is one region group for each loading region of the PicoTiterPlate device. This group contains all the metric results for the corresponding region. The region group also contains key subgroups (one subgroup for each sequencing key defined for the region). The keywords contained in the region group are:

a. name – The name of the region. Each region is automatically assigned a sequential number starting with 1.

b. totalRawWells – The number of raw wells detected in this region of the PicoTiterPlate device. These are wells that are generating enough signal to be considered for further processing.

c. totalKeyPass – The number of wells where the first four bases called match any defined key, detected in this region of the PicoTiterPlate device. A keypass well is assumed to contain a legitimate DNA read. The keypass wells are the wells that are further processed in the downstream data analysis pipeline.

d. key group – The key subgroup is contained in the region group. The key sequence is a known sequence that will be the first four nucleotides sequenced on any read. For each region, two key subgroups will be displayed, one for sample library reads (TCAG) and one for Control DNA reads (ATGC); this is dictated by the keys present (1) on Adaptors used to generate the DNA library and (2) on the Control DNA Beads provided in the GS Sequencing Kits. The keywords contained in the key group are:

   I. keySequence – The actual key sequence. Note that although the keys are four nucleotides long, only the first three are guaranteed to produce signals corresponding to a single nucleotide incorporation, so only these are used for signal normalization (while all four nucleotides are used for the identification of sample library or Control DNA reads).
   II. numKeyPass – The number of wells identified on the PicoTiterPlate device that contained the valid key sequence indicated by the keySequence keyword. A keypass well is assumed to contain a legitimate DNA read. The keypass wells are the wells that are further processed in the downstream data analysis applications.
   III. numDotFailed – The number of reads that failed the "dot" filter. See section 3.2.2 for a description of the quality filters.
   IV. numMixedFailed – The number of reads that failed the "mixed" filter. See section 3.2.2 for a description of the quality filters.
   V. numTrimmedTooShortQuality – The number of reads that failed the length test because of quality trimming. See section 3.2.2 for a description of the quality filters.
   VI. numTrimmedTooShortPrimer – The number of reads that failed the length test because of Primer sequence trimming. See section 3.2.2 for a description of the quality filters.
   VII. totalPassedFiltering – The number of wells that passed all five filters in the Signal Processing application. These are the wells that contain high quality DNA reads that are used (as trimmed) as input for data analysis post-Run processing, or for basecalling of the reads.

4.2.5 454BaseCallerMetrics.txt

The 454BaseCallerMetrics.txt is a 454 parser file that provides general information on the reads and basecalls of the Run, for each combination of PicoTiterPlate device region and sequencing key. An equivalent *.csv file contains the same information in a format suitable for importing into a spreadsheet. Below is a description of all the sections present in this file, with their named groups and keywords. For a general description of the 454 parser file format, see section 13.3.2. For a sample 454BaseCallerMetrics.txt file, see section 13.4.2

1 Comment Header – The comment header contains the version of the software and the date and time the file was created in the form: YYYY/MM/DD HH:MM:SS.

2 runParameters group – This group contains the rig results directory name. The keyword parameter contained in this group is:

a. rigResultDirectory – This keyword indicates the full path to the location of the Data Analysis directory on the computer (either on the Genome Sequencer FLX Instrument or on the DataRig).

3 basecallResults group – The Basecall Results provide a summary of all nucleotides called in all regions for all sequencing keys from the sequencing Run. These numbers are the final read results obtained after the reads were processed through the Genome Sequencer Filters executed during Signal Processing. The keyword parameters contained in this group are:

a. numReads – The total number of DNA reads that were retained (passed) after all five pre-2.0 quality filters were applied. See section 3.2.2 for a description of the quality filters.
b. aveLength – The average length of the DNA reads after quality filter trimming.
c. stdDev – The standard deviation of the value displayed in aveLength.

4 regionKey group – Results are summarized for each PicoTiterPlate device region and read key combination, *i.e.* if there are multiple regions and/or sequencing keys, there will be multiple regionKey groups. (Note that this is different from the display of the 454RuntimeMetricsAll.txt file, which shows "region" groups with "key" subgroups; see section 4.1.2.6.)) Normally, there are two keys, one for sample library reads (TCAG) and one for Control DNA reads (ATGC); this is dictated by the keys present (1) on Adaptors used to generate the DNA library and (2) on the Control DNA Beads provided in the GS Sequencing Kits. The keywords contained in the regionKey group are:

a. region – The name of the region whose metrics are given in this group.
b. key – The key sequence for which this group is providing the metrics.
c. numReads – The total number of DNA reads that were retained (passed) after all five pre-2.0 quality filters were applied in this region with the specified key. See section 3.2.2 for a description of the quality filters.
d. totalBases – The total number of DNA bases in these filtered reads (after trimming).
e. averageLength – A comma-separated list providing the average length of the DNA reads after filtering and trimming, and the corresponding standard deviation.
f. averageQuality – A comma-separated list providing the average quality score of all the DNA bases in these filtered reads (after trimming), and the corresponding standard deviation. For an explanation of individual base quality scores, see section 3.2.2.2.
g. lengthHistogram group – This subgroup is located within the regionKey group. It provides a comma-separated list of the number of filtered reads at each read length observed in this region and with this sequencing key. These values can be used to create a histogram of the read count at each read length. The group consists of multiple instances of the same keyword followed by values as described below:
   I. lengthCount – On each line, the read length (for the abscissa of the histogram), followed by the number of filtered reads that had this trimmed length.
h. qualityHistogram group – This subgroup is located within the regionKey group, and provides a comma-separated list of the number of bases called at each quality score observed in this region and with this sequencing key. These values can be used to create a histogram of the base count at a particular quality value. The group consists of multiple instances of the same keyword followed by values as described below:
   I. qualityBinCount – On each line, the quality score (for the abscissa of the histogram), followed by the number of bases that had this quality score.

4.2.6 454RuntimeMetricsAll.txt

The 454RuntimeMetricsAll.txt file is a 454 parser file that summarizes some performance metrics of the Run. An equivalent *.csv file contains the same information in a format suitable for importing into a spreadsheet. The software returns similar files for each analysis region on the PicoTiterPlate device, and for each sequencing key defined in these regions (*region*.wells.*key*.454RuntimeMetrics.txt; see section 4.1.2.3). Below is a description of all the sections present in these files, with their named groups and keywords. For a general description of the parser file format, see section 13.3.2. For a sample 454RuntimeMetricsAll.txt file, see section 13.4.3.

1. Comment Header – The comment header contains the version of the software, the name of the Run, the name of the analysis, the region name, and the key sequences included. In the "454RuntimeMetricsAll.txt" file, the Region Name and Key Sequence are always "All", by definition; by contrast, the similar "*region*.wells.*key*.454RuntimeMetrics.txt" files (section 4.1.2.3) specify a region and a key sequence. The header also includes the date and time the file was created in the form: YYYY/MM/DD HH:MM:SS.

2. runConditions group – This group contains the physical parameters of the sequencing Run such as the PicoTiterPlate type and number of loading regions on the PicoTiterPlate device. The number of flows and flow order as well as the unique name of the sequencing Run and the subsequent data analysis are also contained in this group. The keyword parameters contained in this group are:

   a. dateOfFile – The date/time when the file was created, in the form:

   YYYY/MM/DD hh:mm

   …where:

YYYY	- four digit year
MM	- two digit month (01 to 12)
DD	- two digit day (01 to 31 depending on month)
hh	- two digit hour (00 to 23)
mm	- two digit minutes (00 to 59)

   b. runName – The unique name of this Run. The full Run name is constructed by concatenating multiple pieces of information separated by "_" characters, and put in the form:

   R_YYYY_MM_DD_hh_mm_rigName_userName_freeForm

   …where:

R	- stands for "Run" and is always the first character of the Run name
YYYY	- four digit year of when the Run was executed
MM	- two digit month (01 to 12) of when the Run was executed
DD	- two digit day (01 to 31 depending on month) of when the Run was executed
hh	- two digit hour (00 to 23) of when the Run was executed
mm	- two digit minute (00 to 59) of when the Run was executed
rigName	- network host name of the computer in the Genome Sequencer FLX Instrument used to execute this Run
userName	- name of the user who executed the Run (from the login username)
freeForm	- any alpha numeric string ("Unique Run name") that was entered by the user

▶▶▶

2 c. analysisName – The unique name of the data analysis that was executed. The full data analysis name is constructed by concatenating multiple pieces of information separated by "_" characters, and put in the form:

D_YYYY_MM_DD_hh_mm_rigName_userName_freeForm

…where:

D	- stands for "Data Analysis" and is always the first character of the analysis name
YYYY	- four digit year of when the data analysis was executed
MM	- two digit month (01 to 12) of when the data analysis was executed
DD	- two digit day (01 to 31 depending on month) of when the data analysis was executed
hh	- two digit hour (00 to 23) of when the analysis was executed
mm	- two digit minute (00 to 59) of when the analysis was executed
rigName	- network host name of the computer on which the data analysis was executed (*e.g.* a Genome Sequencer FLX Instrument; not included if the analysis was performed with the off-instrument version of the software)
userName	- name of the user who executed the data analysis (from the login username; not included if the analysis was performed with the off-instrument version of the software)
freeForm	- one of the following: (1) a string identifying the data processing type that was run. "imageProcessingOnly", for example, contains the image processed results. (2) any alpha-numeric string that was entered by the user as "Unique Analysis name", if the Image Processing portion of data processing was performed using the reanalysis function of the GS Run Browser application.

d. PTPType – A placeholder for the type (size) of the PicoTiterPlate device that was used for the sequencing Run. In future versions of the Genome Sequencer FLX System software, the type of the PicoTiterPlate device entered by the user during the sequencing Run setup will appear here. (Currently set to "unknown".)

e. analysisType – A placeholder for the analysis pipeline that was used to process the data. In future versions of the Genome Sequencer FLX System software, the value set by the user during the sequencing Run setup will appear here. (Currently set to "<analysisType goes here>".)

f. numberOfRegions – The number of physically separate regions present on the PicoTiterPlate device (as defined by the Bead Deposition Gasket used during the preparation of the PicoTiterPlate device). This value was set by the user during the sequencing Run setup. Data from each region is processed separately, and each region has a complete set of metrics.

g. numberOfCycles – A placeholder for the number of nucleotide flow cycles that made up the sequencing Run. In future versions of the Genome Sequencer FLX System software, the value determined by the Run Script file (*.icl; Instrument Control Language) that was chosen by the user during the sequencing Run setup will appear here. (Currently set to -1.)

h. flowOrder – A placeholder for the flow order of the nucleotides for each cycle of the Run. In future versions of the Genome Sequencer FLX System software, the order determined by the Run Script file (*.icl) that was chosen by the user during the sequencing Run setup will appear here. (Currently set to "<flowOrder goes here>".)

▶▶▶

3 region group – There is one region group for each loading region on the PicoTiter-Plate device. This group contains all the metric results for the corresponding region. The region group also contains key subgroups (one subgroup for each key sequence defined for the region). The keywords contained in the region group are:

a. name – The name of the region. The region name corresponds to the location of the region on the PicoTiterPlate device, starting from the left.

b. rawWells – The number of raw wells detected in this region of the PicoTiterPlate device. These are wells that are generating enough signal to be considered for further processing.

c. keyPassWells – The total number of wells where the first four bases called match any defined key, detected in this region of the PicoTiterPlate device. A keypass well is assumed to contain a legitimate DNA read. The keypass wells are the wells that are further processed in the downstream data analysis pipeline.

d. key group – The key subgroup is contained in the region group. The key sequence is a known sequence that will be the first four nucleotides in any read. For each region, two key subgroups will be displayed, one for sample library reads (TCAG) and one for Control DNA reads (ATGC or CATG, for GS FLX standard and GS FLX Titanium chemistries, respectively); this is dictated by the keys present (1) on Adaptors used to generate the DNA library and (2) on the Control DNA Beads provided in the GS Sequencing Kits. The Control DNA read metrics are found in the matchSequenceMetrics subgroup, within the ATGC or CATG key group (see below). The keywords contained in the key group are:
   - I. keySequence – The actual key sequence. Note that although the keys are four nucleotides long, only the first three are guaranteed to produce signals corres-ponding to a single nucleotide incorporation, so only these are used for signal normalization (while all four nucleotides are used for the identification of sample library or Control DNA reads).
   - II. keyPassWells – The number of wells identified on the PicoTiterPlate device that contained the valid key sequence indicated by the keySequence keyword. A keypass well is assumed to contain a legitimate DNA read. The keypass wells are the wells that are further processed in the downstream data analysis pipeline.
   - III. keySignalPerBase – A comma-separated list providing the average signal per base for the key signals and the corresponding standard deviation.
   - IV. ppi1, ppi2, ppi3 – A set of comma-separated lists providing the average signal and the corresponding standard deviation, for each PPi flow during the sequencing Run.

▶▶▶

3 The following subgroups are present only within the ATGC or CATG (Control DNA reads, for GS FLX standard and GS FLX Titanium chemistries, respectively) key group:

V. singletA, singletT, singletG, singletC, singletN – A set of comma-separated lists providing the normalized singlet nucleotide signal and standard deviation of each signal, averaged over all the wells. The singlet 'N' is the average normalized signal of all the nucleotide singlet signals from all the wells.

VI. matchSequenceMetrics group – The matchSequenceMetrics subgroup contains the performance metrics for Control DNA read sequences. Control DNA reads are from the known pieces of DNA introduced in the sequencing Run as DNA Control Beads (provided in the GS Sequencing Kits); they are used to gauge the performance of the Run. There are no keywords in this group, only sequence subgroups.

01. sequence group – This subgroup contains the metrics for a specific Control DNA sequence. The keywords in the sequence group are:

a. name – The name of the Control DNA sequence these metrics are for. For the "all" name, the metrics are combined for all the Control DNA sequences.

b. wellsAtQuality – There will be multiple wellsAtQuality entries, providing a comma-separated range of values in format:
accuracy, length, percent keypass, # wells

…where

accuracy	– specifies the minimum percentage agreement between the known Control DNA sequences and a basecalled read
length	– indicates the number of bases over which this accuracy is maintained, starting at the first base of the Control DNA sequence
percent keypass	– indicates what percent of keypass wells had a sequence that matched the known sequence with at least the specified accuracy over the length specified [typically over 80% keypass wells with this sequencing key (ATG; Control DNA reads), at 98% accuracy, over the first 200 bases]
# wells	– indicates how many wells had a sequence that matched the known sequence with at least the specified accuracy over the length specified

There are *caveats* to the % keypass benchmark provided above for the Control DNA reads (ATG or CAT sequencing key, for the GS FLX standard and GS FLX Titanium chemistries, respectively). If the sample DNA library reads have a very high signal per base, they may overcome those from the Control DNA reads, which may then not read well and miss their % keypass benchmark. Also, if the number of Control DNA wells is low, the %keypass for those wells will not be reliable. In such cases, a low % keypass for the Control DNA reads would not necessarily be indicative of poor sample DNA library reads or of a generally poor quality sequencing Run.

4.2.7 *region.key*.454Reads.fna

The (trimmed) nucleotide sequences for the filtered reads of that region and key. For a general description of the ".fna" (FASTA) file format, see section 13.3.6.. For a sample *region.key*.454Reads.fna file, see section 13.4.4.

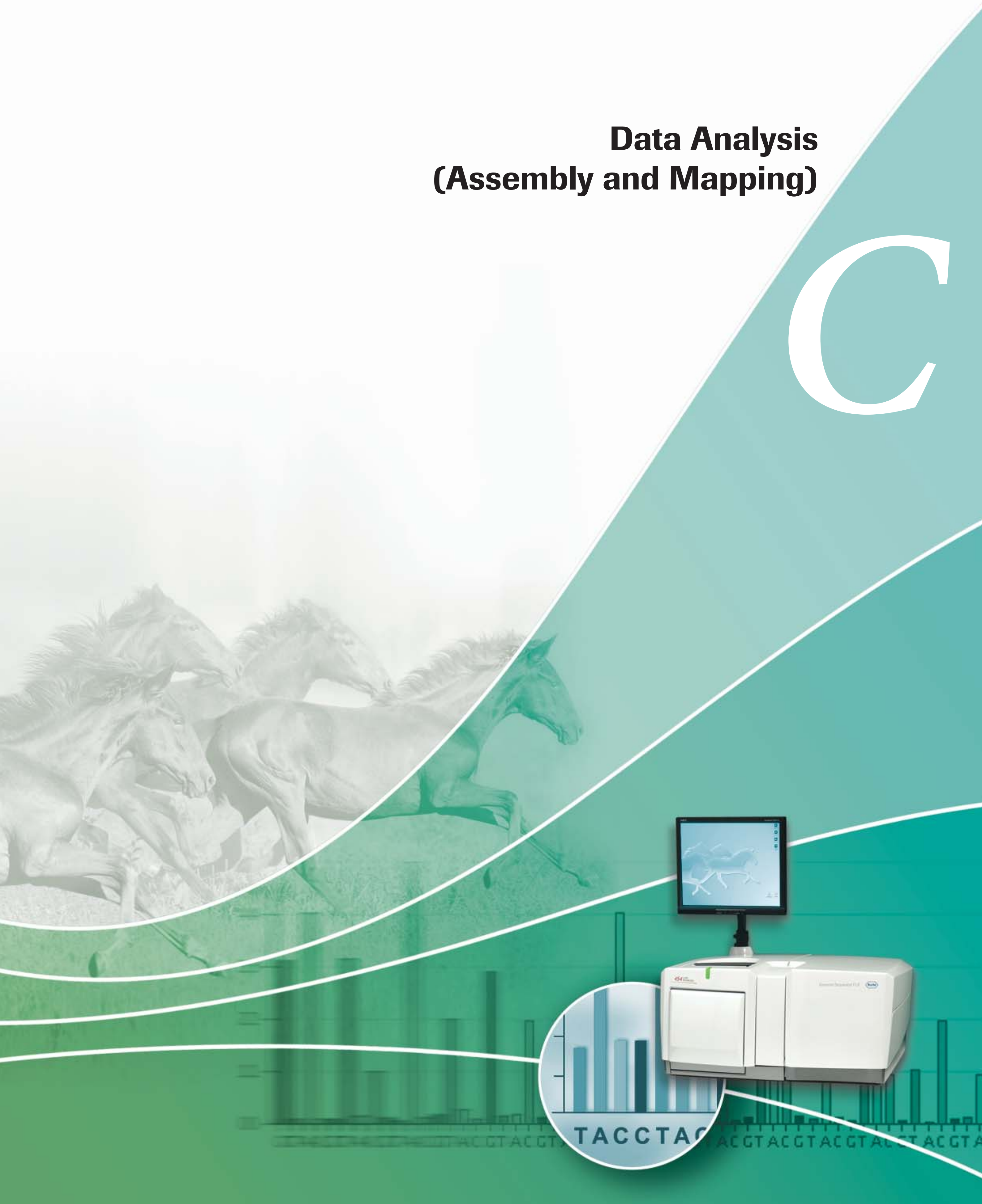
These files are named using only the first three characters of the key because although the actual keys are four nucleotides long, only the first three are guaranteed to produce signals corresponding to a single nucleotide incorporation, so only these are used for signal normalization (though all four nucleotides are used for the identification of sample library or Control DNA reads). So, for region "1" and key "TCAG", the filename would be "1.TCA.454Reads.fna".

4.2.8 *region.key*.454Reads.qual

This file contains the nucleotide quality scores (Phred-equivalent) for the high quality (filtered and trimmed) reads of that region and key. For a description of the process to compute individual base quality scores, see section 3.2.2.2. For a general description of the ".qual" file format, see section 13.3.6. For a sample *region.key*.454Reads.qual file, see section 13.4.4.

Data Analysis (Assembly and Mapping)

C



Part C: Data Analysis (Assembly and Mapping)

5. GS *De Novo* Assembler Application 86

5.1 Introduction to the GS *De Novo* Assembler 86

5.2 Using the GS *De Novo* Assembler Via the Graphical User Interface 87

5.2.1 Launching the Graphical User Interface Application 88

5.2.2 GS *De Novo* Assembler Interface Overview 89

5.2.2.1 The Main Buttons, Status Area, and Progress Box 90

5.2.2.2 The Quick Start and Documentation Text Links 90

5.2.3 Creating a New Assembly Project 91

5.2.4 Setting Assembly Project Parameters 92

5.2.5 Adding Read Data Sets to an Assembly Project 96

5.2.5.1 Adding GS Read Data Sets to an Assembly Project 97

5.2.5.2 Adding FASTA Read Data Sets to an Assembly Project 102

5.2.6 Computing an Assembly 103

5.2.6.1 Computing an Assembly for the First Time 103

5.2.6.2 Re-Computing an Assembly 103

5.2.7 Stopping an Assembly Project Computation in Progress 103

5.2.8 Viewing the Results of an Assembly Computation 103

5.2.8.1 Viewing the GS read and FASTA read metadata: the Project Tab 104

5.2.8.2 Viewing the Assembly Metrics Files: the Result Files Tab 105

5.2.8.3 Viewing the Alignment Data: the Alignment Results Tab 106

5.2.8.4 Viewing the Flowgram Data: the Flowgram Tab 108

5.3 Using the GS *De Novo* Assembler Via the Command Line Interface 112

5.3.1 One-Step Assembly: the runAssembly Command 112

5.3.2 Project-Based Assembly: the newAssembly and Related Commands 118

5.3.2.1 The newAssembly Command 118

5.3.2.2 The addRun Command 118

5.3.2.3 The removeRun Command 120

5.3.2.4 The runProject Command 121

5.4 GS *De Novo* Assembler Application Output 125

5.4.1 Project/Output Folder Structure 125

5.4.2 GS *De Novo* Assembler Output Files 126

5.4.2.1 454AllContigs.fna 127

5.4.2.2 454AllContigs.qual 127

5.4.2.3 454LargeContigs.fna 127

5.4.2.4 454LargeContigs.qual 127

5.4.2.5 454NewblerMetrics.txt 128

5.4.2.6 454NewblerProgress.txt 134

5.4.2.7 454AlignmentInfo.tsv 134

5.4.2.8 454ContigGraph.txt 135

5.4.2.9 454PairAlign.txt 136

5.4.2.10 454ReadStatus.txt 137

5.4.2.11 454TrimStatus.txt 137

5.4.2.12 454Contigs.ace or ace/ContigName.ace or consed/… 138

5.4.2.13 sff/sfffilename 140

5.4.2.14 454Scaffolds.fna 140

5.4.2.15 454Scaffolds.qual 140

5.4.2.16 454Scaffolds.txt 140

5.4.2.17 454PairStatus.txt 141

5.4.2.18 454TagPairAlign.txt 141

5.5 Assembling with Reads Obtained Using the Sanger Sequencing Method 142

5.5.1 Direct Incorporation into the Assembly Process 142

5.5.2 Sanger Reads with available Quality Scores 142

5.5.3 Sanger Read Annotations 143

5.5.3.1 The “template”, “dir”, and “ library” Annotations 143

5.5.3.2 The “trim” Annotation 144

5.5.3.3 The “scf” and “phd” Annotations 144

5.5.4 Recommended Method for including Sanger Reads 145

Part C: Data Analysis (Assembly and Mapping), continued

6. GS Reference Mapper Application 147

6.1 Introduction to the GS Reference Mapper 147

6.2 Using the GS Reference Mapper Via the Graphical User Interface 148

6.2.1 Launching the Graphical User Interface Application 148
6.2.2 GS Reference Mapper Interface Overview 149
6.2.2.1 The Main Buttons, Status Area, and Progress Box 149
6.2.2.2 The Quick Start and Documentation Text Links 150
6.2.3 Creating a New Mapper Project 150
6.2.4 Setting Reference Mapper Project Parameters 151
6.2.5 Adding Read Data to a Mapping Project 156
6.2.5.1 Adding GS Read Data Sets to a Mapping Project 157
6.2.5.2 Adding FASTA Read Data Sets to a Mapping Project 162
6.2.6 Adding Reference DNA Sequence Files to a Mapping Project 163
6.2.7 Computing the Read Alignments to the Reference 165
6.2.7.1 Computing the Alignments to the Reference for the First Time 165
6.2.7.2 Re-Computing the Alignments to the Reference 165
6.2.8. Stopping a Mapping Project Computation In Progress 165
6.2.9 Viewing the Results of a Mapping Computation 166
6.2.9.1 Viewing the Reference, GS read, and FASTA read metadata: the Project Tab 166
6.2.9.2 Viewing the Mapping Metrics Files: the Result Files Tab 168
6.2.9.3 Viewing the High Confidence Differences: the HCDiffs Tab 169
6.2.9.4 Viewing the Alignment Data: the Alignment Results Tab 171
6.2.9.5 Viewing the Flowgram Data: the Flowgrams Tab 173

6.3 Using the GS Reference Mapper Via the Command Line Interface 176

6.3.1 One-Step Mapping: the runMapping Command 176
6.3.2 Project-Based Mapping: the newMapping and Related Commands 184
6.3.2.1 The newMapping Command 184
6.3.2.2 The setRef Command 185
6.3.2.3 The addRun Command 186
6.3.2.4 The removeRun Command 188
6.3.2.5 The runProject Command 189

6.4 GS Reference Mapper Application Output 193

6.4.1 Project/Output Folder Structure 193
6.4.2 GS Reference Mapper Output Files 194
6.4.2.1 454AllContigs.fna 195
6.4.2.2 454AllContigs.qual 195
6.4.2.3 454LargeContigs.fna 195
6.4.2.4 454LargeContigs.qual 195
6.4.2.5 454AllDiffs.txt 196
6.4.2.6 454HCDiffs.txt 197
6.4.2.7 454NewblerMetrics.txt 198
6.4.2.8 454NewblerProgress.txt 204
6.4.2.9 454MappingQC.xls 205
6.4.2.10 454AlignmentInfo.tsv 208
6.4.2.11 454PairAlign.txt 209
6.4.2.12 454ReadStatus.txt 210
6.4.2.13 454RefStatus.txt 210
6.4.2.14 454TrimStatus.txt 211
6.4.2.15 454Contigs.ace or ace/refaccno.ace or consed/… 211
6.4.2.16 sff/sfffilename 212
6.4.2.17 454PairStatus.txt 212
6.4.2.18 454TagPairAlign.txt 212

Part C: Data Analysis (Assembly and Mapping), continued

7. SFF Tool Commands 213

7.1 sfffile 213
7.2 sffinfo 218
7.3 sff2scf 219
7.4 fnafile 221
7.5 sffrescore 224

8. GS Run Browser Application 225

8.1 Introduction to the GS Run Browser 225
8.1.1 Launching the GS Run Browser Application 226
8.1.2 GS Run Browser Interface Overview 228
8.1.2.1 The Main Buttons 229
8.1.2.2 The Tabs 229
8.1.2.3 Buttons and Plots 230
8.1.2.3.1 Scroll Bars 230
8.1.2.3.2 Navigation Buttons 230
8.1.2.3.3 Mousing Functions 232
8.2 The Overview Tab 233
8.2.1 General Description 233
8.2.2 Source of the Data 233
8.2.3 Features and Functionalities 234
8.3 The Wells Tab 235
8.3.1 General Description 235
8.3.2 Source of the Data 235
8.3.3 Features and Functionalities 236
8.3.4 Well Flowgrams (for Wells Generating Library Reads) 241
8.3.4.1 General Description 241
8.3.4.2 Features and Functionalities 242
8.3.5 Well Tri-Flowgrams (for Wells Generating Control DNA Reads) 243
8.3.5.1 General Description 243
8.3.5.2 Features and Functionalities 244
8.3.6 Location Flowgrams 245
8.3.6.1 General Description 245
8.3.6.2 Features and Functionalities 245
8.3.7 Subtraction Raw Flowgrams 246
8.3.7.1 General Description 246
8.3.7.2 Features and Functionalities 246
8.4 The Signals Tab 247
8.4.1 General Description 247
8.4.2 Source of the Data 248
8.4.3 Features and Functionalities 248
8.5 The Reads Tab 250
8.5.1 General Description 250
8.5.2 Source of the Data 250
8.5.3 Features and Functionalities 251

Part C: Data Analysis (Assembly and Mapping), continued

8.6 The Control DNA Tab 252
8.6.1 General Description 252
8.6.2 Source of the Data 252
8.6.3 Features and Functionalities 253
8.6.4 Control DNA Consensus Flowgrams 254
8.6.4.1 General Description 254
8.6.4.2 Features and Functionalities 255
8.7 The Filters Tab 256
8.7.1 General Description 256
8.7.2 Source of the Data 256
8.7.3 Features and Functionalities 257

Important Note: October 2008 marks the first release of the new GS FLX Titanium series chemistry for the Genome Sequencer FLX System. For the time being, the system supports only the non-MID "General" (*e.g.* Shotgun) sequencing applications under the GS FLX Titanium chemistry. For Paired End or Amplicon sequencing, or for the preparation and sequencing of MID libraries of any type, users must continue to use the GS FLX standard series kits and procedures (last updated in December 2007).

However, the Genome Sequencer FLX Software version 2.0.00, associated with the October 2008 release, is fully backward-compatible with, and can process datasets generated with any of the Genome Sequencer System's chemistries (GS 20 chemistry, GS FLX standard chemistry, and GS FLX Titanium chemistry). This manual describes **all** the functionalities of the 2.0.00 software, even those that apply **only** to datasets generated with older chemistries, such that **it completely replaces** the December 2007 issue of the *Genome Sequencer FLX Data Analysis Software Manual* (which addressed the software version 1.1.03).

The Genome Sequencer FLX Titanium series manuals are easily identified by their new cover graphics and distinctive tri-color stripes (reflecting the GS FLX Titanium kits packaging). All the methods, protocols and applications supported on the GS FLX standard chemistry will be enabled on the GS FLX Titanium chemistry in the near future.

Features supported only under the GS FLX standard chemistry:

- While the use of Paired End and of MID-tagged Shotgun (sstDNA) libraries is supported under the GS FLX standard chemistry, these are not currently supported under the GS FLX Titanium chemistry. Therefore, the information relative to the analysis of datasets from Paired End or from MID-tagged libraries that appears in Parts A and C of this manual applies only to libraries prepared under the GS FLX standard chemistry.
- While the Amplicon Variant Analyzer software v. 2.0.00 provides important new features, such as the support for MIDs with Amplicon libraries, those libraries are not currently supported under the GS FLX Titanium chemistry. Therefore, the entire content of Part D of this manual, including the new features, applies only to Amplicon libraries prepared under the GS FLX standard chemistry.

5. GS *De Novo* Assembler Application

5.1 Introduction to the GS *De Novo* Assembler

The GS *De Novo* Assembler software is an interactive application that allows the user to create assembly projects, add and remove reads from the project, run the assembly algorithms on the project data, and view the output produced by the assembly computations. The Assembler operates via a Graphical User Interface (GUI; section 5.2) or from the command line interface (section 5.3).

Assembly projects allow the construction of *de novo* assemblies of the reads from one or more sequencing Runs, using as input the "read flowgrams" (as SFF files) from some or all the regions of the Run(s) of interest, and generates a consensus sequence of the sample DNA library (in one or more contigs). As described in sections 3.2.1 and 3.2.2 the read information provided in the SFF files contains the high quality data to be used for assembly.

The assembler allows the inclusion of one or more Paired End Runs into the analysis, enabling the ordering and orientation of the assembled contigs from shotgun sequencing Runs into scaffolds; this requires the preparation of a separate Paired End library from the same DNA material that was used to prepare an sstDNA library used for shotgun sequencing. In the case of Long Paired End libraries, however, the presence of reads from a Shotgun library in the project is not strictly required, as Long Paired End reads can themselves be assembled with each other.

Also, the assembler software allows the inclusion of one or more FASTA files of reads, such as reads obtained using the Sanger sequencing method ("Sanger reads") in the analysis; the longer read length and paired end information in these files can then be used to improve the contigs and scaffolds.

In the assembly process, the software performs the following operations (see also Table 1–2):

- Identify pairwise overlaps between reads
- Construct multiple alignments of reads that tile together (forming contigs), based on the pairwise overlaps
- Generate consensus basecalls of the contigs by averaging the processed flow signals for each nucleotide flow included in the alignment
- Output the contig consensus sequences and corresponding quality scores, along with an ACE file of the multiple alignments and assembly metrics files.

Additional steps with the Paired End option:

- Identify pairwise overlaps between Paired End reads and the shotgun contigs
- Organize the contigs into scaffolds (order, orientation, and approximate distance)
- Output the scaffolded consensus sequences and corresponding quality scores, along with an AGP file of the scaffolds and specific metrics Tables.

While the read overlaps and multiple alignments are made in "nucleotide" space, the consensus basecalling and quality value determination for contigs are performed in "flowspace". Work in flowspace allows the averaging of processed flow signals (a continuous variable) at each nucleotide flow of the sequencing Run(s) rather than called bases (a discontinuous variable) at each nucleotide position along the alignment. It also allows the use of information from the "negative flows", *i.e.* flows where no nucleotide incorporation is detected, which would obviously not be possible in nucleotide space. Alignment in flowspace thus results in an improved accuracy for the final basecalls.

The GS *De novo* Assembler application is not available on the Genome Sequencer FLX Instrument and must be run on a DataRig.

5.2 Using the GS *De Novo* Assembler Via the Graphical User Interface

Assemblies can be performed either using the Graphical User Interface described in this section, or using the Linux command line, as described in section 5.3. The application includes graphical interfaces to:

- create a new assembly project
- open an existing assembly project
- carry out an assembly computation on an open project
- view the results of a completed assembly computation
- view the progress and logging information of a completed assembly computation

The GS *De Novo* Assembler GUI application is written in Java, and requires Java version 1.5. Java 1.5 is included as part of the off-instrument software package to be installed on your DataRig(s). In the course of its operation, the GS *De Novo* Assembler GUI application makes use of C and C++ components, all of which can be used to perform command line assemblies separately from the graphic application, as described in sections 5.3 and 7.

5.2.1 Launching the Graphical User Interface Application

The GS *De novo* Assembler GUI application is launched via a single command, whose command line structure is the following:

```
gsAssembler
```

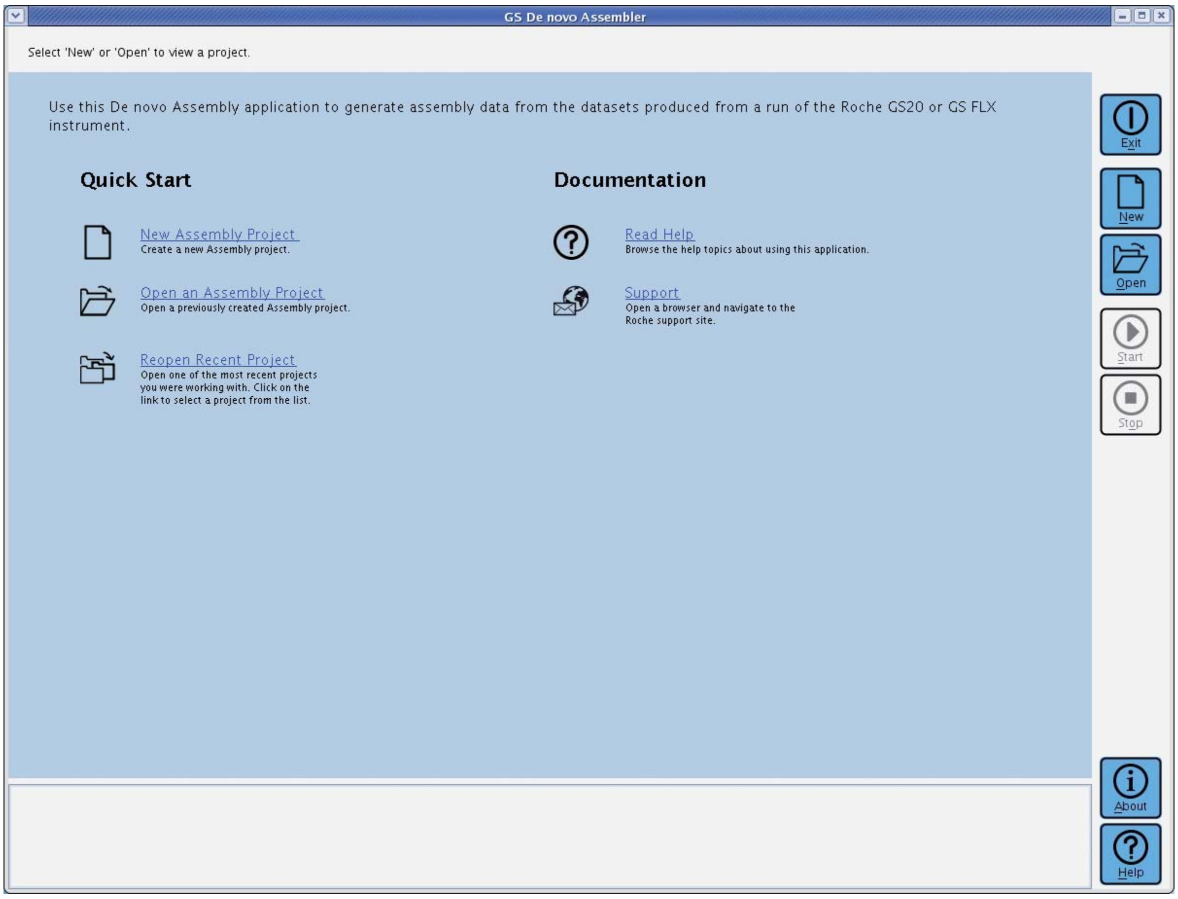
This will open a splash screen (Figure 5–1) and the main application windows (Figure 5–2, below). The splash screen can be dismissed by clicking anywhere on it, but it will also disappear on its own after approximately 6 seconds.

GS De novo Assembler
Version 2.0.00
Roche
www.roche-applied-science.com
454 SEQUENCING
(c) 454 Life Sciences 2006-2008. All rights reserved.

Figure 5–1: The GS *De Novo* Assembler application splash screen

5.2.2 GS *De Novo* Assembler Interface Overview

The GS *De Novo* Assembler entry window (Figure 5–2) comprises a vertical toolbar of main function buttons; a menu area with three "Text Links" for creating and opening assembly projects, and two "Text Links" for help and support; as well as a status area along its top edge and a progress box at the bottom.



Select 'New' or 'Open' to view a project.

Use this De novo Assembly application to generate assembly data from the datasets produced from a run of the Roche GS20 or GS FLX instrument.

Quick Start

New Assembly Project
Create a new Assembly project.

Open an Assembly Project
Open a previously created Assembly project.

Reopen Recent Project
Open one of the most recent projects you were working with. Click on the link to select a project from the list.

Documentation

Read Help
Browse the help topics about using this application.

Support
Open a browser and navigate to the Roche support site.

Figure 5–2: The GS *De Novo* Assembler application's main window, just after the application is launched

The progress information field at the bottom of the window can be resized by dragging its top frame separator. If the field is not visible, it is probably simply collapsed; you can view it by dragging up the edge of the collapsed frame.

5.2.2.1 The Main Buttons, Status Area, and Progress Box

Seven main buttons are always visible on the toolbar, along the right-hand side of the GS *De Novo* Assembler's main window:

- The **Exit** button closes the GS *De Novo* Assembler application
- The **New** button allows you to create a new assembly project
- The **Open** button allows you to open an existing assembly project
- The **Start** button begins the assembly (computation) of an open assembly project
- The **Stop** button halts the execution of an ongoing assembly computation
- The **About** button shows the GS *De novo* Assembler splash screen
- The **Help** button shows a help dialog (online help is not currently implemented)

The top of the main GS *De Novo* Assembler window contains a status area that displays a command hint, while the bottom panel is a progress box which remains blank until an assembly project has been opened.

5.2.2.2 The Quick Start and Documentation Text Links

There are 3 text links under the **Quick Start** column:

- The **New Assembly Project** text link creates a new assembly project. It performs the same action as the **New** button on the right side of the window.
- The **Open an Assembly Project** text link opens an existing project. It performs the same action as the **Open** button on the right side of the window.
- The **Reopen Recent Project** text link displays a dialog from which you can open a project on which you worked recently.

There are 2 text links under the **Documentation** column:

- The **Read Help** text link shows a help dialog. Online help is not currently implemented; please use this Software Manual, or click on the Support button (see below), or contact your Roche Representative for any information regarding the Genome Sequencer FLX System and its software package. The Read Help text button performs the same action as the **Help** button on the right side of the window.
- The **Support** text link opens the default internet browser and navigates to the Roche support site.

5.2.3 Creating a New Assembly Project

To create a new assembly project, either click on the **New Project** button in the right toolbar, or click on the "New Assembly Project" text button in the **Quick Start** column. This displays a dialog (Figure 5–3) in which you can specify the name and directory location for a new project.

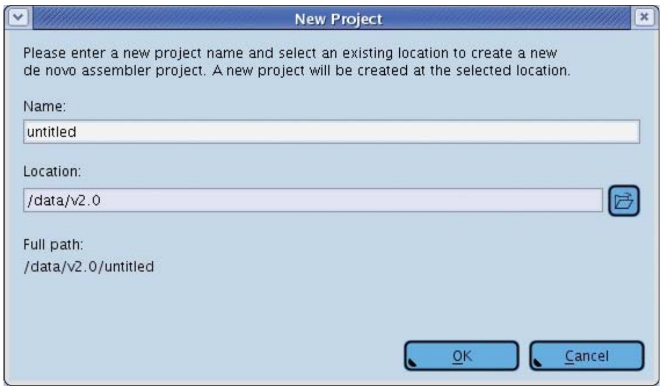

New Project
Please enter a new project name and select an existing location to create a new de novo assembler project. A new project will be created at the selected location.
Name:
untitled
Location:
/data/v2.0
Full path:
/data/v2.0/untitled
OK
Cancel

Figure 5–3: The New Project window of the GS *De Novo* Assembler application

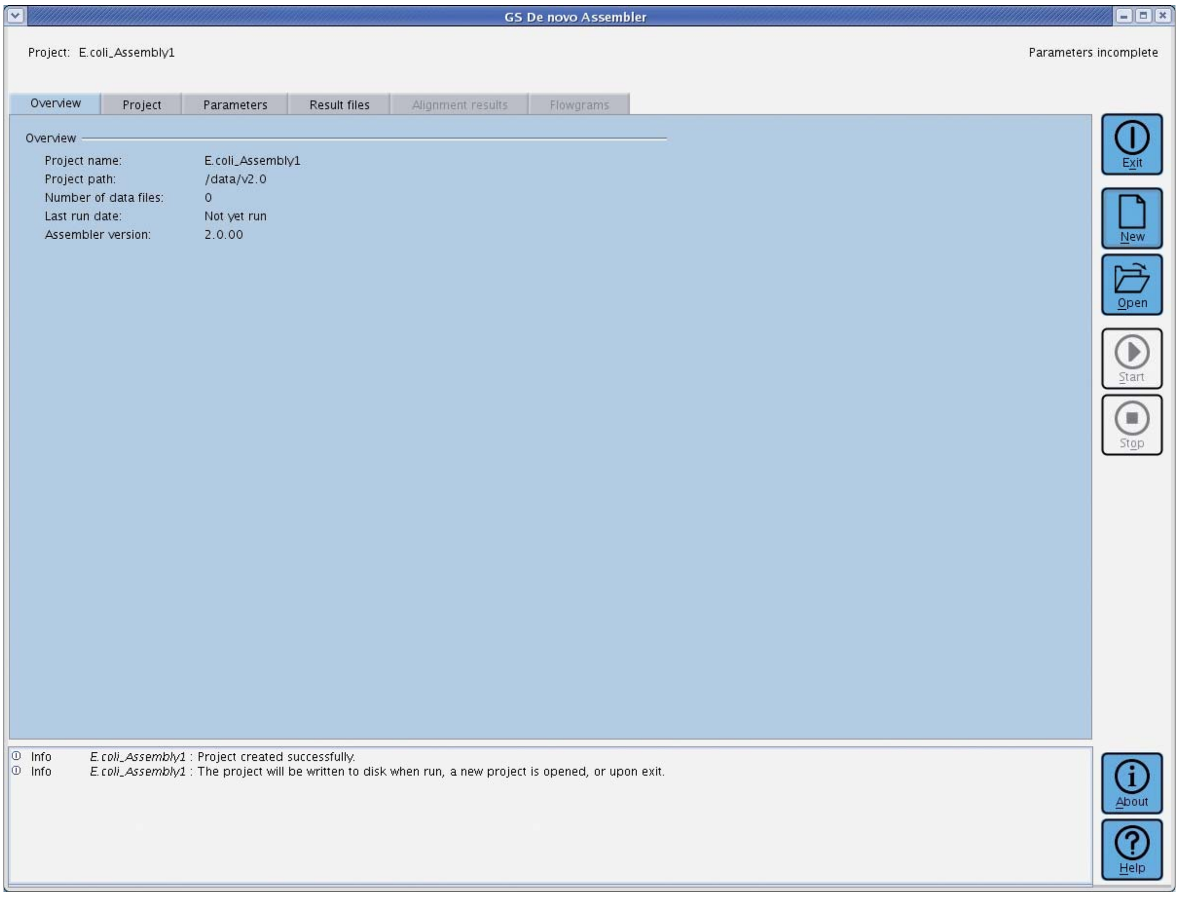
Type a name for the new project in the **Name** text field. To specify the project location, either type the full path in the **Location** text field, or click on the **Open Project** button to the right of the text field and use the "Select Project Location" window (not shown) to navigate to the directory where you want to create the new project. The **Full Path** field changes as you update the **Name** and **Location** fields. When you are satisfied with the location and name for the new project, click on the **OK** button. You can save a project by clicking on the Yes button to the **Save** prompt.

You may save the project to your hard drive either by using the Exit button to exit the GS *De Novo* Assembler application (you will be prompted to save before the application actually exits) or by adding read data and running the project by clicking the Start button (*i.e.*, compute the assembly), in which case the project will automatically be saved prior to computation, as described in section 5.2.5, below.

5.2.4 Setting Assembly Project Parameters

After clicking **OK** in the New Project window, the main application window changes to display the Overview tab for the project (Figure 5–4). This view summarizes various data about the project. The data displayed on this tab is updated as new information becomes available, which occurs when data files are added to the project and when a project computation completes.

The upper right hand corner of the application window indicates “Parameters incomplete” and the Start button is initially disabled because at least one Read Data file must be added to the project before an Assembly computation can be initialized. Other errors in input parameters can also cause this message to appear. Once all such errors have been corrected and at least one Read Data file is added to the project, the message will change to “Ready for analysis” and the Start button will become active.



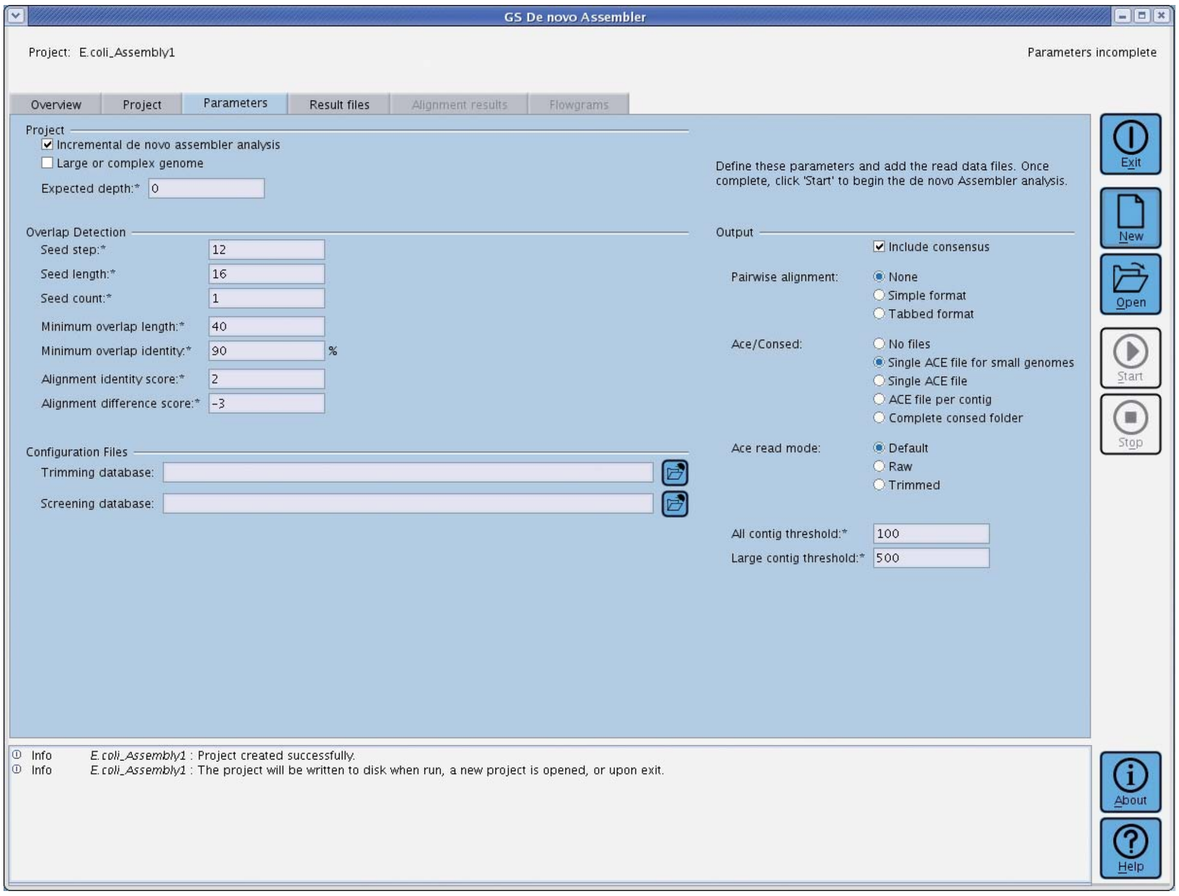
Overview

Project name: E.coli_Assembly1
Project path: /data/v2.0
Number of data files: 0
Last run date: Not yet run
Assembler version: 2.0.00

```
Info    E.coli_Assembly1 : Project created successfully.
Info    E.coli_Assembly1 : The project will be written to disk when run, a new project is opened, or upon exit.
```

Figure 5–4: The Overview tab of a new project in the GS *De Novo* Assembler application. This tab is used to summarize certain characteristics of an open assembly project. The values displayed are updated any time data files are added to the project and when an assembly computation completes.

Click on the Parameters tab of the main application window to display the parameters for the assembly project (Figure 5–5). For a new project, the assembly parameters that take numerical values are initially set at with their default values, as shown. If an invalid value is entered into any of these fields, a small red "X" appears in the lower left corner of the field and activates the "Parameters incomplete" warning in the upper right hand corner of the screen. Pausing the mouse over the red X reveals a tooltip indicating the allowed values for the parameter (Figure 5–6).



Project: E.coli_Assembly1

Parameters incomplete

Project
[✓] Incremental de novo assembler analysis
[ ] Large or complex genome
Expected depth:* 0

Define these parameters and add the read data files. Once complete, click 'Start' to begin the de novo Assembler analysis.

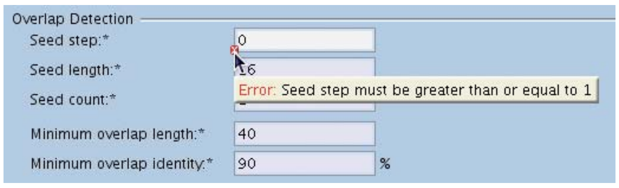
Overlap Detection
Seed step:* 12
Seed length:* 16
Seed count:* 1
Minimum overlap length:* 40
Minimum overlap identity:* 90 %
Alignment identity score:* 2
Alignment difference score:* -3

Output
[✓] Include consensus
Pairwise alignment: [✓ None] Simple format Tabbed format
Ace/Consed: No files [✓ Single ACE file for small genomes] Single ACE file ACE file per contig Complete consed folder
Ace read mode: [✓ Default] Raw Trimmed
All contig threshold:* 100
Large contig threshold:* 500

Configuration Files
Trimming database:
Screening database:

```
Info   E.coli_Assembly1 : Project created successfully.
Info   E.coli_Assembly1 : The project will be written to disk when run, a new project is opened, or upon exit.
```

Figure 5–5: The Parameters tab of a new project in the GS *De Novo* Assembler application. This tab is used to set the parameters and specify Read Data Set(s) for an assembly computation. If horizontal and vertical scrollbars are showing, use them to view the entire set of parameters.



Overlap Detection
Seed step:* 0
Seed length:* 16
Seed count:*
Minimum overlap length:* 40
Minimum overlap identity:* 90 %

Error: Seed step must be greater than or equal to 1

Figure 5–6: Example of a tooltip revealed by pausing the mouse over the red X in the corner of an assembly parameter entry field, after entering an invalid parameter value

The following paragraphs describe the assembly parameters that you can control, how each of the parameters affects subsequent assembly operations, and the default and acceptable range values for the numerical parameters.

- Project Parameter
  - Incremental *de novo* assembler analysis – If this check box is selected, new read data will be assembled into any existing assembly that was created during previous computations in this project. If it is not selected, all the project data will be assembled anew each time an assembly is computed in this project, overwriting any existing assembly results.
    - Default: selected (Note: This value is not saved with a project, and is always reset to "selected" when a project is re-opened)
  - Large or complex genome – If this check box is selected, large or complex datasets (*i.e.* larger fungal or eukaryotic genomes) will be assembled successfully and speedily.
    - Default: not selected
  - Expected depth: filters out random-chance level events at bigger depths that would be significant at a lower depth level.
    - Default: 0
    - Allowed values: 0 or greater, where a value of 0 tells the assembler to not use expected depth information in its computation
- Overlap Detection Parameters
  - Seed step – The number of bases between seed generation locations used in the exact k-mer matching part of the overlap detection
    - Default value: 12
    - Allowed values: 1 or greater
  - Seed length – The number of bases used for each seed in the exact k-mer matching part of the overlap detection (*i.e.* the "k" value of the k-mer matching)
    - Default value: 16
    - Allowed values: 6-16
  - Seed count – The number of seeds required in a window before an extension is made
    - Default value: 1
    - Allowed values: 1 or greater
  - Minimum overlap length – The minimum length of overlaps used by the assembler
    - Default value: 40
    - Allowed values: 1 or greater
  - Minimum overlap identity – The minimum percent identity of overlaps used by the assembler
    - Default value: 90
    - Allowed values: 0-100
  - Alignment identity score – When multiple overlaps are found, the per-overlap-column identity score used to sort the overlaps for use in the progressive alignment
    - Default value: 2
    - Allowed values: 0 or greater
  - Alignment difference score – When multiple overlaps are found, the per-overlap-column difference score used to sort the overlaps for use in the progressive multi-alignment
    - Default value: -3
    - Allowed values: 0 or less

- Configuration Files
  - Trimming database – The path to a FASTA file of sequences to be used to trim the ends of input reads (for cloning vectors, primers, adapters or other end sequences).
  - Screening database – The path to a FASTA file of sequences to be used to screen the input reads for contaminants. A read that completely aligns against the screening database is removed so that it is not used in the computation; if only part of a read aligns with the screening database, no action is taken.
  - Long file paths will be displayed in full by pausing the mouse cursor on the text area
- Output Parameters
  - Include consensus – If this check box is selected, the application will generate all the output files, including the output files related to the generation of contigs and consensus sequence information. If it is not selected, the application will perform the assembly and will output files associated with the creation of the multiple-alignments, but will not output files or metrics involved with contig or consensus information.
    - Default: selected (Note: This value is not saved with a project, and is always reset to "selected" when a project is re-opened)
  - Pairwise Alignment – Determines whether the 454PairAlign.txt file is output; this file contains the overlaps used by the assembler.
    - None – the file is not generated
    - Simple format – the file contains a human-readable view of the alignments
    - Tabbed format – the file contains tab-delimited lines of the overlaps
    - Default is "None". See section 5.4.2.9 for a description of the 454PairAlign.txt file.
  - Ace/Consed – Determines whether or what form of ACE file(s) are output by the application.
    - No files – no ACE file is generated
    - Single ACE file for small genome – a single ACE file is generated if fewer than 4 million reads are input to the assembly
    - Single Ace file – a single ACE file is generated containing the multiple alignments of all the contigs in the assembly, irrespective of the size of the genome
    - ACE file per contig – a separate ACE file is generated for each contig in the assembly
    - Complete consed folder – a "consed" folder is generated, containing all the directories and files necessary to display the data in the consed software
    - Default is "Single ACE file for small genomes". See section 6.4.2.15 for a description of the 454Contigs.ace file and of the ace and consed directories.
  - Ace read mode – when the ACE file is generated, output reads using either the raw, complete basecalled read or using just the trimmed portion of the reads (after low quality, vector and key trimming)
    - Default is set to output trimmed reads
    - Raw is to output the entire sequence of reads
    - Trimmed is to output trimmed read
  - All contig threshold – the minimum number of bases for a contig to be output in the 454AllContigs.fna file
    - Default: 100
  - Large contig threshold – the minimum number of bases for a contig to be output in the 454LargeContigs.fna file
    - Default: 500

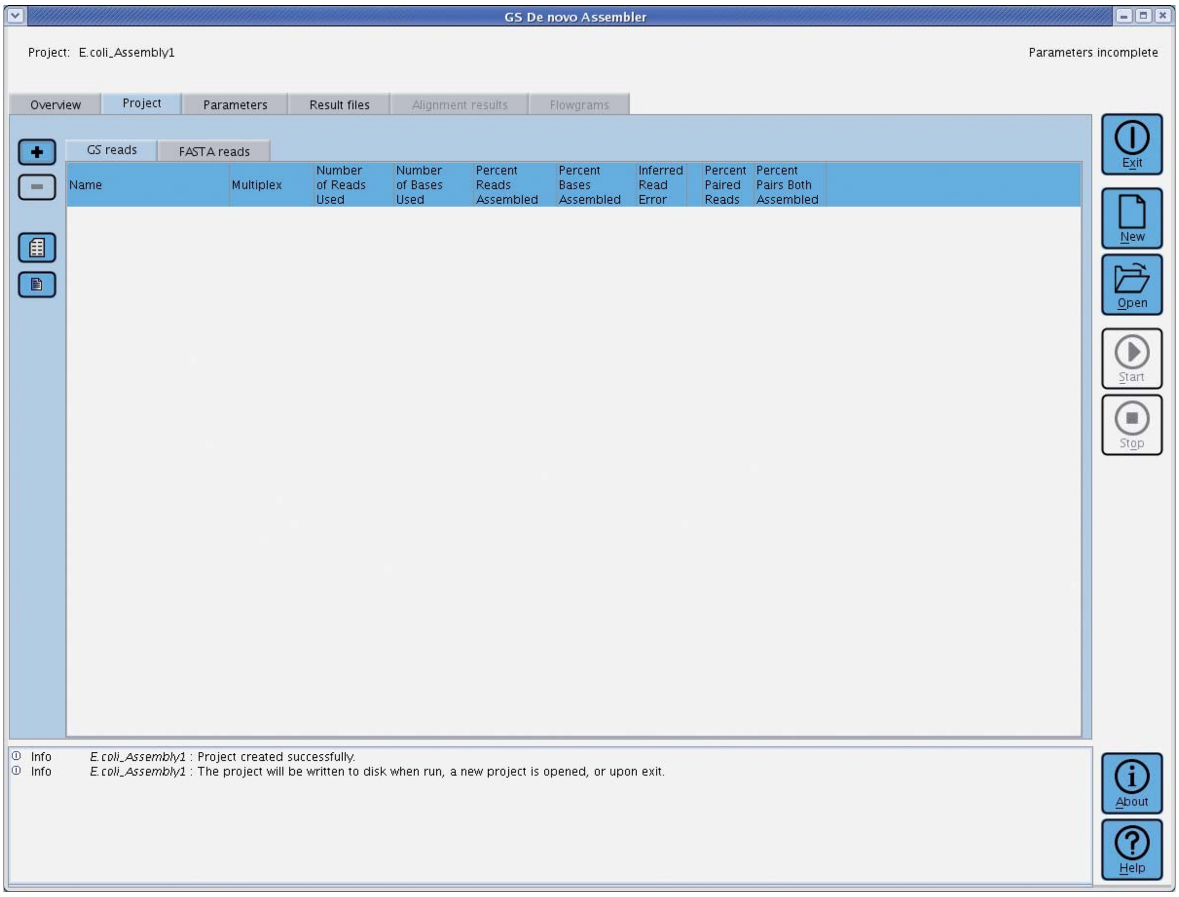
External files: FASTA files are not actually copied to the project directory. Rather, the file path is simply recorded in the project setup. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies to all instances of FASTA files that can be added to an Assembly project, whether done via the GUI or the command line:

- FASTA reads files, including Sanger reads (Project tab/FASTA reads sub-tab)
- Trimming database files (Parameters tab)
- Screening database files (Parameters tab)

Unlike with the GS Read Data files, no symbolic link to the file is created for FASTA files.

5.2.5 Adding Read Data Sets to an Assembly Project

Read Data files are added to a project using the GS Reads and FASTA reads sub-tabs under the Project tab of the main application window (Figure 5–7). The two sub-tabs are used to add either Genome Sequencer reads and/or FASTA reads to an assembly project. The **Start** button in the toolbar, which is used to start an assembly computation, remains disabled, and the "Parameters incomplete" message remains visible, until at least one Read Data file (either GS read or FASTA read) has been added to the project.

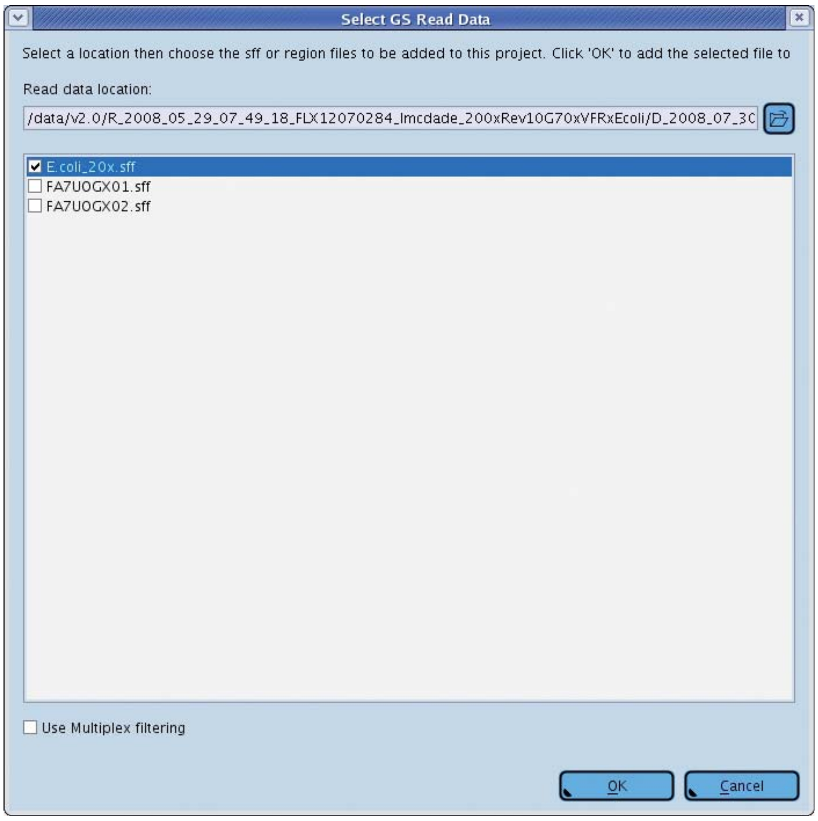


```
Info    E.coli_Assembly1 : Project created successfully.
Info    E.coli_Assembly1 : The project will be written to disk when run, a new project is opened, or upon exit.
```

Figure 5–7: The Project tab window of the GS *De Novo* Assembler application

5.2.5.1 Adding GS Read Data Sets to an Assembly Project

To add GS read data sets to a project, click on the GS reads sub-tabs, then click the **Add** button $\boxed{+}$ to open the "Select GS Read Data" window (Figure 5–8). The top portion of this window lists, for the specified location, the GS Read Data files that can be added to the project; if no files exist at this location, the window displays the message: "There are no GS Read Data files in the selected directory". The bottom area (MID) is used to append Multiplex Identifiers (MID) filtering information to the selected read file(s), if MIDs were used in the design of your experiment and they are relevant to your assembly analysis. By default, the MID section is not visible, but it can be viewed by selecting the "Use MID/Multiplex filtering" checkbox located in the lower left corner of the window (Figure 5–9).


Select GS Read Data

Select a location then choose the sff or region files to be added to this project. Click 'OK' to add the selected file to

Read data location:

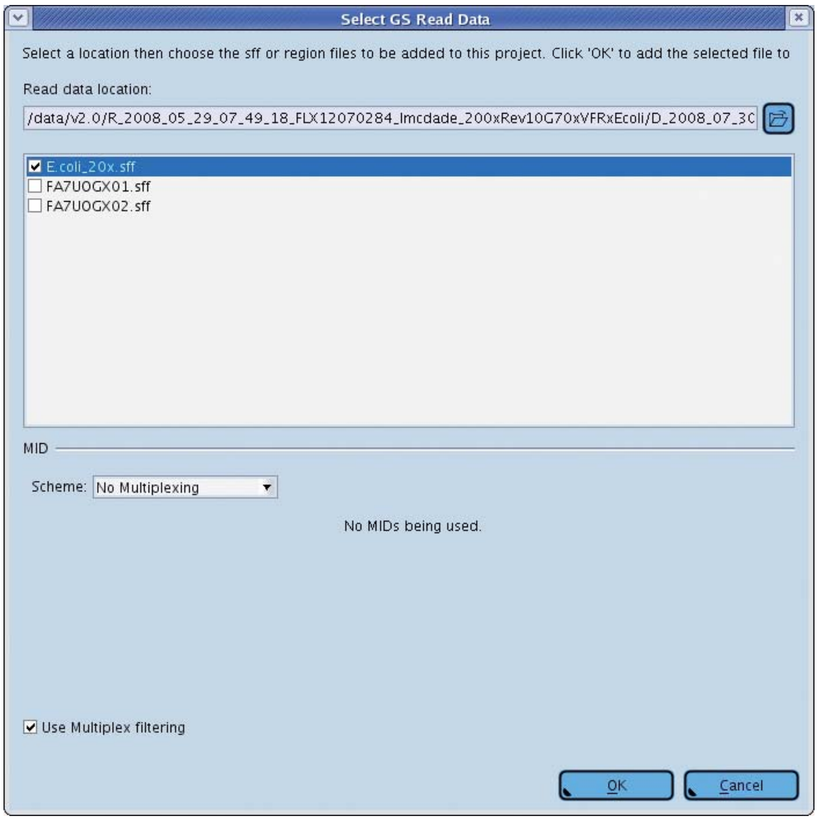
/data/v2.0/R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/D_2008_07_3C

- [x] E.coli_20x.sff
- [ ] FA7U0GX01.sff
- [ ] FA7U0GX02.sff

- [ ] Use Multiplex filtering

OK Cancel

Figure 5–8: The Select GS Read Data window of the GS *De Novo* Assembler application



Select a location then choose the sff or region files to be added to this project. Click 'OK' to add the selected file to

Read data location:

/data/v2.0/R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/D_2008_07_30

- [x] E.coli_20x.sff
- [ ] FA7UOGX01.sff
- [ ] FA7UOGX02.sff

MID

Scheme: No Multiplexing

No MIDs being used.

- [x] Use Multiplex filtering

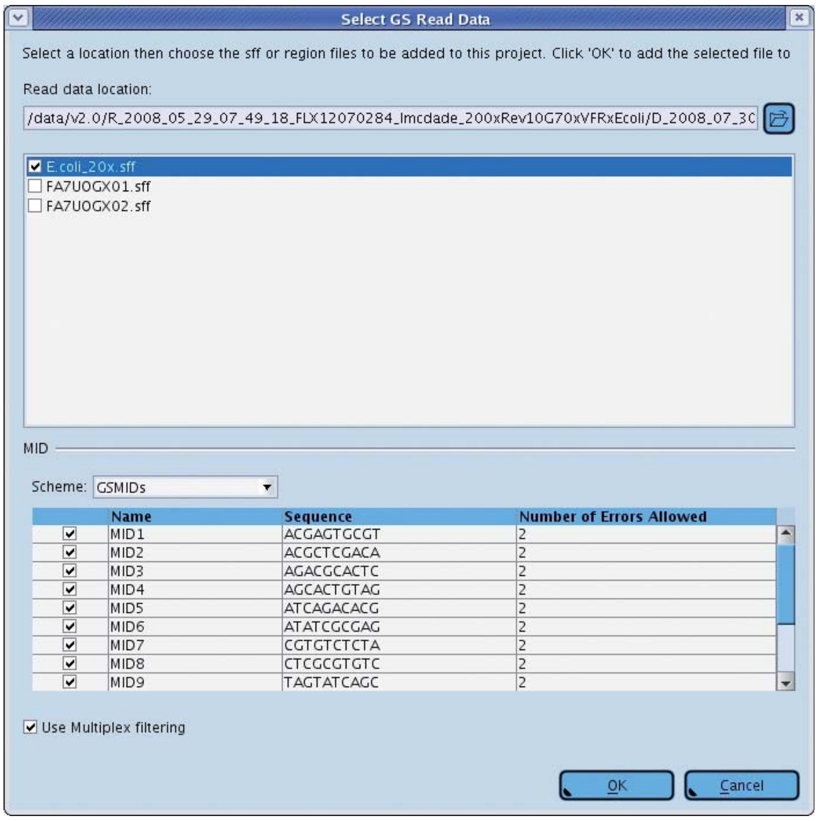
Figure 5–9: The Select GS Read Data window of the GS *De Novo* Assembler application, with the "Use MID/Multiplex filtering" checkbox selected and the MID area showing

To change to a different location that contains Read Data files, click on the **"Change the read data location"** button (with the open-folder icon) to the right of the text field and use the "Select Read Data Directory" window (not shown) to navigate to and select the new location. You may also directly type or paste the desired path into the "File Name:" field of the navigation window, rather than browsing through the file system for the directory of interest. While using this interface, only directories will appear, not the files within those directories. When you return from the Select GS Read Data window, the list of available Read Data files found at the new location will be refreshed and appear in the field below the Read data location text area (as seen on Figure 5–8). Use the check boxes to select the file(s) you want to include in the assembly project. If MID multiplexing is not required for these files, select "No Multiplexing" from the scheme (drop-down) menu (see Figure 5–9) (or do not select the "Use Multiplex filtering" checkbox), and click the **OK** button.

When the Use Multiplex Filtering checkbox is selected, special controls for MID filtering are displayed in the bottom half of the Select GS Read Data window. Multiplex Identifiers (MIDs) allow you to design an experiment whereby multiple libraries are prepared using distinct MID tags and sequenced together, in the same region of a PTP device. (This can greatly improve the workflow and cost effectiveness of your experiment.) In these conditions, the reads of all the pooled libraries are contained in a single Read Data (.sff) file; the MID controls on this window allow you to filter the reads from one (or more) library(ies) in the selected Read Data files, for inclusion in the Assembly project. Note, this filtering operation does not produce a different Assembly for each specified MID. Rather, the reads of the SFF files are scanned for the presence of MIDs, and the reads containing the selected MIDs are made eligible for assembly in the project, as a group. Any reads without the specified MIDs will be effectively ignored. To produce independent Assemblies for different MIDs (or sets of MIDs), you must create independent Assembly projects for each of the desired MID subsets of the data.

The MID filtering information is contained in 'MID schemes', which list a set of MID tags along with their exact sequences and the number of sequencing errors permitted in each MID string that will still allow the tag to be recognized. The Genome Sequencer FLX System software comes with a standard MID scheme (GSMIDs) that contains 14 MIDs (of which MIDs 1 to 12 match the "MID Adaptors" available in kit form), each with two errors allowed. These MIDs are all 10-mers that were carefully chosen such that two errors in any of them can be corrected without confusing one MID with another MID of the set.

By default, no MID/Multiplexing scheme is associated with Read Data files. To use an MID scheme, select it using the Scheme drop down menu. When an MID scheme is selected, a table of the MIDs present in that scheme is displayed, as shown in Figure 5–10. Use the checkboxes on the left to select or deselect the MIDs to be included in the assembly. The names, sequences and error limits of the MIDs selected will be used to filter the reads (from the Read Data sets selected in the top part of the window). The result is that only the reads that contain the selected MIDs, within the number of errors specified in the scheme, will be included in the project.



Select a location then choose the sff or region files to be added to this project. Click 'OK' to add the selected file to

Read data location:

/data/v2.0/R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/D_2008_07_3C

- ☑ E.coli_20x.sff
- ☐ FA7UOGX01.sff
- ☐ FA7UOGX02.sff

MID

Scheme: GSMIDs

	Name	Sequence	Number of Errors Allowed
☑	MID1	ACGAGTGCGT	2
☑	MID2	ACGCTCGACA	2
☑	MID3	AGACGCACTC	2
☑	MID4	AGCACTGTAG	2
☑	MID5	ATCAGACACG	2
☑	MID6	ATATCGCGAG	2
☑	MID7	CGTGTCTCTA	2
☑	MID8	CTCGCGTGTC	2
☑	MID9	TAGTATCAGC	2

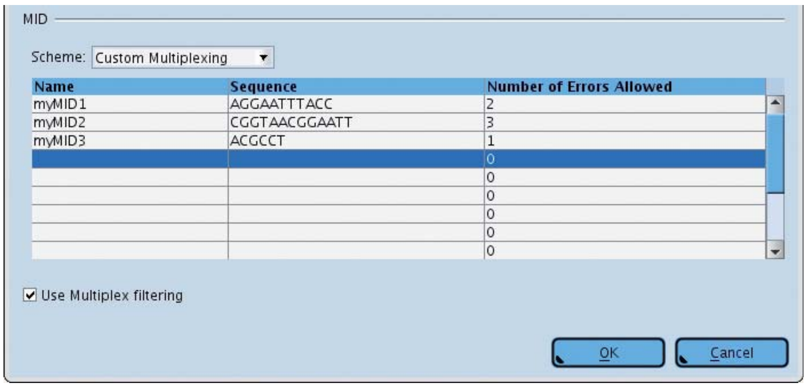
☑ Use Multiplex filtering

Figure 5–10: The Select GS Read Data window of GS *De Novo* Assembler application's main window, showing the table of MIDs present in the MID scheme GDMIDs

The list of MID schemes provided in the Scheme drop down menu is read from the file MIDConfig.parse. For pre-existing MID schemes, the names and sequences of the MIDs, and the number of errors allowed, cannot be changed in the Select GS Read Data window. To modify these values, or to add new user-specified MID schemes, edit the MIDConfig.parse file using any text editor. The standard location of this file is /opt/454/config/MIDConfig.parse; more generally, if the software is installed to the */install/directory* folder, the configuration file will be found at */install/directory*/config/MIDConfig.parse. See section 13.4.15 to view the default MIDConfig.parse file and for a description of how to edit it.

You can also create new MIDs directly in the Select GS Read Data window, by selecting "Custom Multiplexing" from the Scheme drop-down menu. In this case, a table will appear and new MID names, sequences, and error limits can be typed into the table (Figure 5–11). However, these names, sequences and error limits will only apply to the current project.

There is no formal limit to the number or length of custom MIDs that one can define in an Assembly project. However, efficient demultiplexing requires that the MID sequences be divergent enough that the software will be able to differentiate them from one another (within the "number of errors allowed" specified).


MID

Scheme: Custom Multiplexing

Name	Sequence	Number of Errors Allowed
myMID1	AGGAATTTACC	2
myMID2	CGGTAACGGAATT	3
myMID3	ACGCCT	1
		0
		0
		0
		0
		0
		0

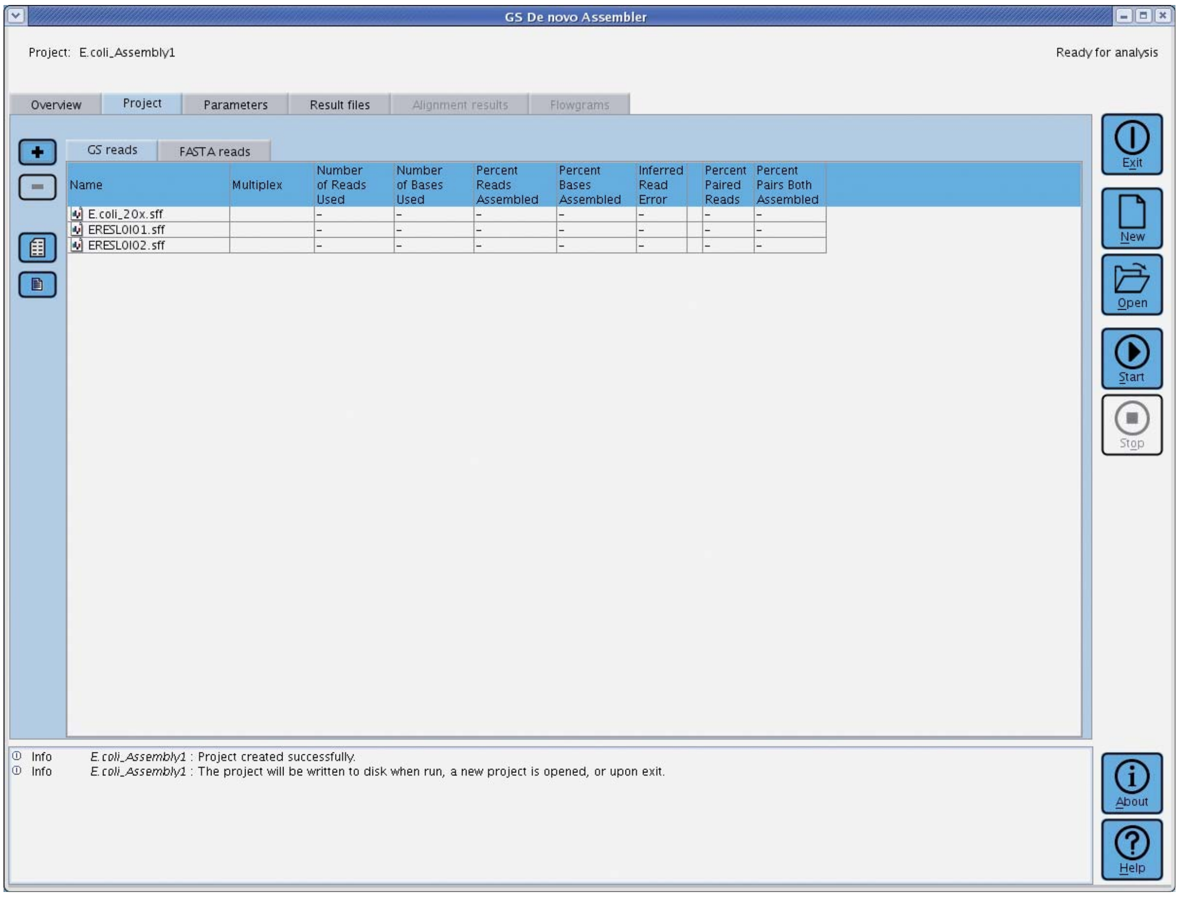
☑ Use Multiplex filtering

Figure 5–11: Creating custom MIDs

Once the MID scheme has been specified, click the **OK** button to add the selected data files (with MID filtration information) to the list of GS read data files. These are displayed in tabular format (Figure 5–12). Except for the Name column and the Multiplex column, which contain comma-delimited lists of the MIDs associated with the file, all columns with run statistics are initially filled with dashes. These data are updated in the table each time the project runs to completion.

You can add any number of Read Data files to a project, both by selecting multiple files present in a given directory (see Figure 5–8), or by using the **Add** button as many times as you need to navigate to all the files you want to include. You can even add multiple files that share the same name into a project, as long as they have different path locations in the file system (*e.g.* "/dir1/path1/reads.sff" and "/dir2/otherpath2/reads.sff"). In such a case, both files will be added to the Read Data list and displayed using the same name. To see the path to a file listed in the Read Data area of the main window (and the file's last modification date), pause the mouse over the filename of interest and a tooltip containing the file's path will be displayed. Files that have failed validation will have a red X left of the file name as an indication of failure. Pausing the mouse cursor over the red X will bring up a tooltip explaining the problem encountered.

External files GS Read Data files (SFF files) are not actually copied to the project directory. Rather, a symbolic link to the file is created and placed in the project's sff directory. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies whether the files are added via the GUI or the command line.



Project: E.coli_Assembly1

Ready for analysis

Name	Multiplex	Number of Reads Used	Number of Bases Used	Percent Reads Assembled	Percent Bases Assembled	Inferred Read Error	Percent Paired Reads	Percent Pairs Both Assembled
E.coli_20x.sff		–	–	–	–	–	–	–
ERESL0I01.sff		–	–	–	–	–	–	–
ERESL0I02.sff		–	–	–	–	–	–	–

```
Info    E.coli_Assembly1 : Project created successfully.
Info    E.coli_Assembly1 : The project will be written to disk when run, a new project is opened, or upon exit.
```

Figure 5–12: The GS Read Data sub-tab of the Project tab, in the GS *De Novo* Assembler application's main window, showing 3 SFF files that were added to the project

In certain circumstances, you may want to apply a different MID selection to different Read Data sets, within the same project. For example, you may have multiple libraries of the same organism, made with different MIDs (or with/without MIDs), that you want to assemble together. In this case, select the Read Data set(s) that need to be filtered with any given MID (or combination of MIDs), and click **OK** to add the MID-filtered data file(s) to the project. Back on the Project tab of the GS *De Novo* Assembler window (with the GS Reads sub-tab active) (Figure 5–12), click the **Add** button (+) again, and select other Read Data sets, to be filtered with different MIDs.

You can also remove currently included Read Data files from a project, by selecting them from the list in the Read Data area of the Assembler application's Parameters tab, and then clicking the **Remove** button ⊟.

The **Remove** button removes the selected Read Data files from the list, but does not immediately remove the file(s) from the project's sff sub-directory or from any computed assembly results. Actual removal of the files from the project (and of the reads they contain from an existing assembly computed therefrom), occurs only when the assembly is re-computed (see section 5.2.6).

5.2.5.2 Adding FASTA Read Data Sets to an Assembly Project

The procedure for adding FASTA data files to an Assembly project is nearly identical to the one for adding GS Read data (section 5.2.5.1). The only difference is that when a directory containing FASTA files is selected, the software examines the files in that directory to determine which files are FASTA files. This can take some time if there are many files in the directory, so a progress bar is displayed to show the progress of the search.

External files: FASTA files are not actually copied to the project directory. Rather, the file path is simply recorded in the project setup. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies to all instances of FASTA files that can be added to an Assembly project, whether done via the GUI or the command line:

- FASTA reads files, including Sanger reads (Project tab/FASTA reads sub-tab)
- Trimming database files (Parameters tab)
- Screening database files (Parameters tab)

Unlike with the GS Read Data files, no symbolic link to the file is created for FASTA files.

5.2.6 Computing an Assembly

5.2.6.1 Computing an Assembly for the First Time

When at least one Read Data file has been added to the project and all parameters are within acceptable limits, the **Start** button becomes enabled and the "Ready for analysis" message appears in the upper right corner of the application window (*e.g.* see Figure 5–12, above). Press the **Start** button to carry out the assembly computation of the current project, *i.e.* as defined in the Project and Parameters tabs. As the computation progresses, the **Start** button is disabled, periodic progress messages appear in the progress box, at the bottom of the application's main window, and the current stage of the assembly computation along with a progress bar are displayed in the upper right corner of the window. When the assembly computation is complete, the **Start** button is re-enabled and the message "Ready for analysis" appears again in the upper right corner of the application window.

5.2.6.2 Re-Computing an Assembly

After an assembly has been performed on a project, the assembly computation can be repeated on that project – with the same Read Data or after addition or removal of one or more files, – with the same assembly parameter settings or with different ones. If the "Incremental *de novo* assembler analysis" checkbox on the Parameters tab is checked, this re-analysis will take much less time than the initial analysis, since only some data will be re-computed (see section 5.2.4).

5.2.7 Stopping an Assembly Project Computation in Progress

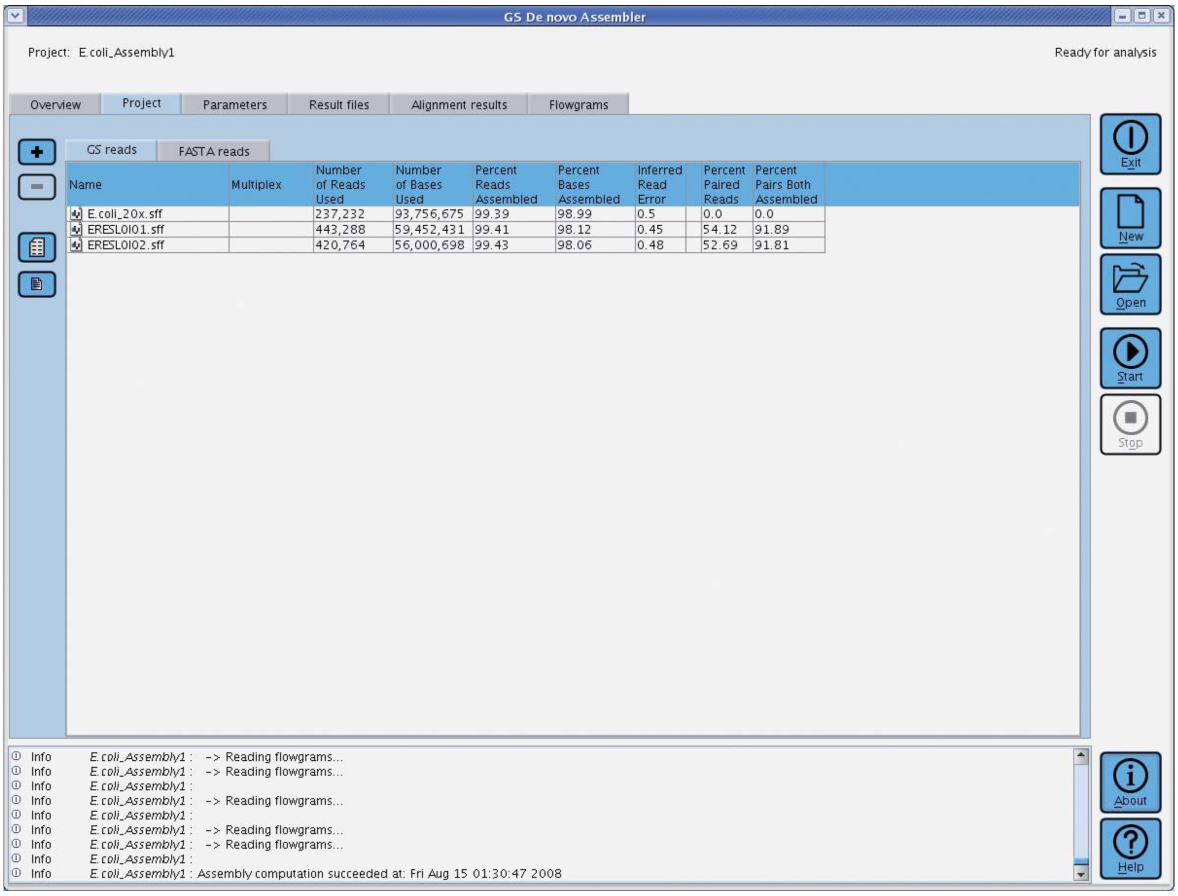
Clicking on the **Stop** button while an assembly project computation is in progress halts the computation. The effect may not be instantaneous, however, with any delay being a function of the complexity of the assembly being performed and of the stage of the computation at the time it was interrupted; an informational message appears in the progress box at the bottom of the application's main window, containing the text "Warning: stopRun called for this project. Stopping….". When the assembly computation has been halted, the **Stop** button is disabled and the **Start** button re-enabled.

5.2.8 Viewing the Results of an Assembly Computation

After an assembly project has been computed, the results of the assembly can be viewed on the Overview, Project, Results Files, Alignment Results, and Flowgram tabs of the GS *De Novo* Assembly application's main window.

5.2.8.1 Viewing the GS read and FASTA read metadata: the Project Tab

After a successful assembly, the data for the various columns of the reads in the GS reads and FASTA reads tables will be updated (Figure 5–13). Placing the mouse pointer over any cell in the table brings up a tool tip with additional information about that item. A successful assembly also updates data in the Overview tab.



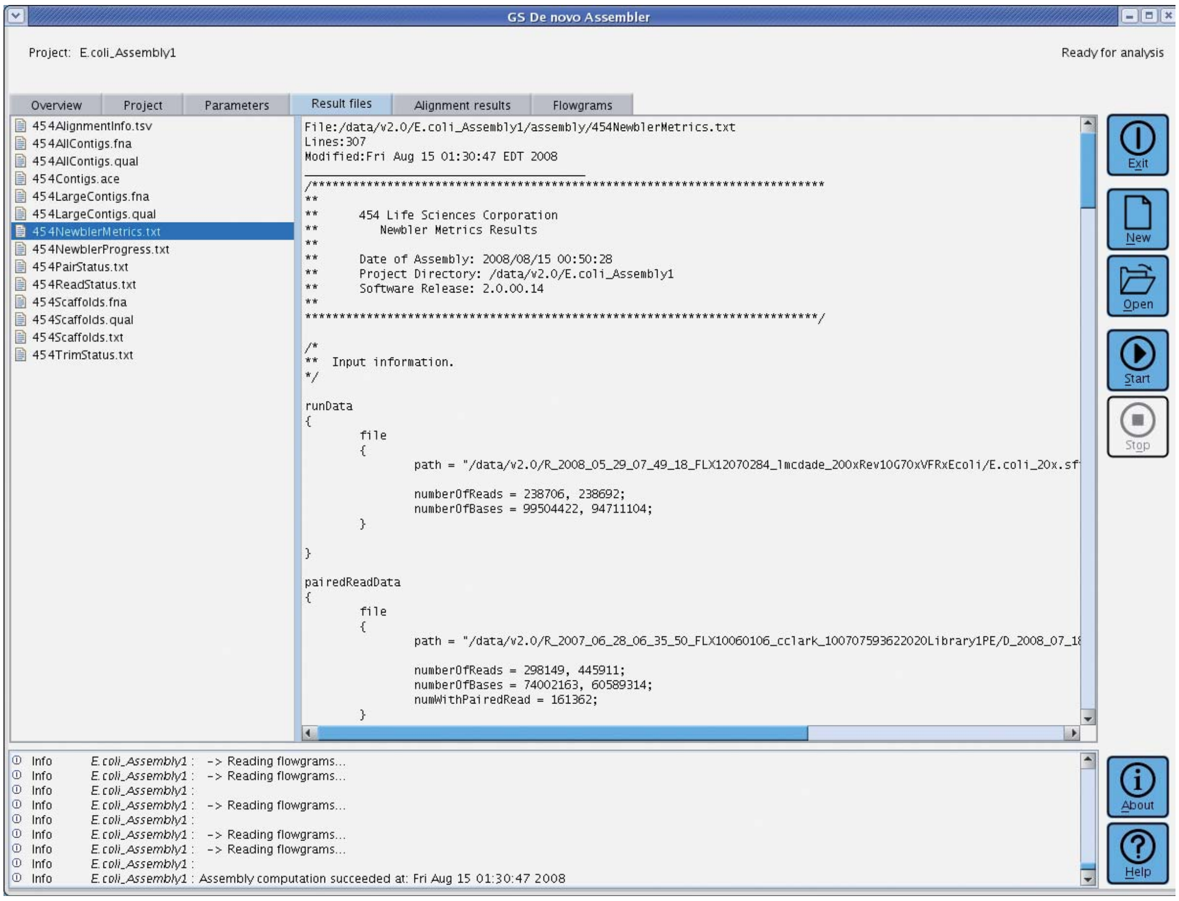
Name	Multiplex	Number of Reads Used	Number of Bases Used	Percent Reads Assembled	Percent Bases Assembled	Inferred Read Error	Percent Paired Reads	Percent Pairs Both Assembled
E.coli_20x.sff		237,232	93,756,675	99.39	98.99	0.5	0.0	0.0
ERESL0I01.sff		443,288	59,452,431	99.41	98.12	0.45	54.12	91.89
ERESL0I02.sff		420,764	56,000,698	99.43	98.06	0.48	52.69	91.81

```
Info    E.coli_Assembly1 :   -> Reading flowgrams...
Info    E.coli_Assembly1 :   -> Reading flowgrams...
Info    E.coli_Assembly1 :
Info    E.coli_Assembly1 :   -> Reading flowgrams...
Info    E.coli_Assembly1 :
Info    E.coli_Assembly1 :   -> Reading flowgrams...
Info    E.coli_Assembly1 :   -> Reading flowgrams...
Info    E.coli_Assembly1 :
Info    E.coli_Assembly1 : Assembly computation succeeded at: Fri Aug 15 01:30:47 2008
```

Figure 5–13: An example of the GS *De Novo* Assembler application main window (Project tab, GS reads sub-tab), after a successful assembly

5.2.8.2 Viewing the Assembly Metrics Files: the Result Files Tab

The Result Files tab allows you to view the various metrics files generated by the GS *De Novo* Assembly software (described in detail in Section 5.4). Using the list in the left-hand panel of the tab, click on the name of the output file you want to examine; its content will be displayed on the right-hand panel of the tab (Figure 5–14). For very long files, such as the sequences of all contigs of a large assembled genome, this view displays the file truncated to 50,000 lines. When this occurs, a note to that effect specifies that it is so both at the beginning and at the end of the display. The file itself remains intact, however.



Project: E.coli_Assembly1

Ready for analysis

```
454AlignmentInfo.tsv
454AllContigs.fna
454AllContigs.qual
454Contigs.ace
454LargeContigs.fna
454LargeContigs.qual
454NewblerMetrics.txt
454NewblerProgress.txt
454PairStatus.txt
454ReadStatus.txt
454Scaffolds.fna
454Scaffolds.qual
454Scaffolds.txt
454TrimStatus.txt
```

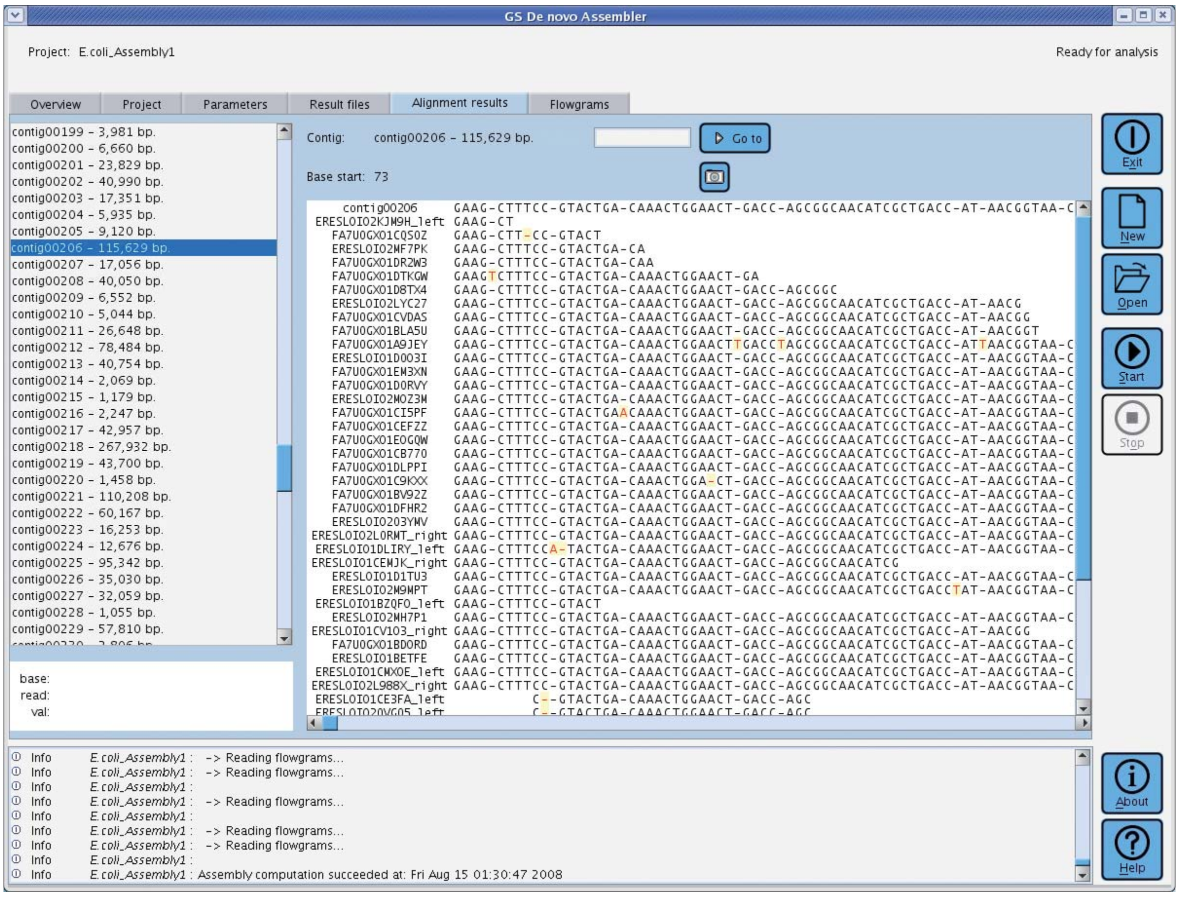
```
File:/data/v2.0/E.coli_Assembly1/assembly/454NewblerMetrics.txt
Lines:307
Modified:Fri Aug 15 01:30:47 EDT 2008
______________________________________
/****************************************************************************
**
**      454 Life Sciences Corporation
**         Newbler Metrics Results
**
**      Date of Assembly: 2008/08/15 00:50:28
**      Project Directory: /data/v2.0/E.coli_Assembly1
**      Software Release: 2.0.00.14
**
****************************************************************************/

/*
**  Input information.
*/

runData
{
        file
        {
                path = "/data/v2.0/R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/E.coli_20x.sf

                numberOfReads = 238706, 238692;
                numberOfBases = 99504422, 94711104;
        }

}

pairedReadData
{
        file
        {
                path = "/data/v2.0/R_2007_06_28_06_35_50_FLX10060106_cclark_100707593622020Library1PE/D_2008_07_1

                numberOfReads = 298149, 445911;
                numberOfBases = 74002163, 60589314;
                numWithPairedRead = 161362;
        }
```

```
Info    E.coli_Assembly1 :  -> Reading flowgrams...
Info    E.coli_Assembly1 :  -> Reading flowgrams...
Info    E.coli_Assembly1 :
Info    E.coli_Assembly1 :  -> Reading flowgrams...
Info    E.coli_Assembly1 :
Info    E.coli_Assembly1 :  -> Reading flowgrams...
Info    E.coli_Assembly1 :  -> Reading flowgrams...
Info    E.coli_Assembly1 :
Info    E.coli_Assembly1 : Assembly computation succeeded at: Fri Aug 15 01:30:47 2008
```

Figure 5–14: An example of the GS *De Novo* Assembler application main window (Result Files tab) after a successful assembly computation. Clicking on a file from the list in the left column causes its contents to be displayed in the right panel. Files containing more than 50,000 lines will truncated, and flagged as such.

5.2.8.3 Viewing the Alignment Data: the Alignment Results Tab

Click on the Alignment Results tab to view the alignment data from the assembly computation (Figure 5–15). The left panel contains a list of all the contigs produced by the assembly. To view the alignments to a contig, click on that contig on the list, and the multi-alignment of all the reads that aligned to that contig will be displayed on the right panel. By default, the first contig on the list will be automatically selected.



GS De novo Assembler

Project: E.coli_Assembly1

Ready for analysis

```
contig00199 - 3,981 bp.
contig00200 - 6,660 bp.
contig00201 - 23,829 bp.
contig00202 - 40,990 bp.
contig00203 - 17,351 bp.
contig00204 - 5,935 bp.
contig00205 - 9,120 bp.
contig00206 - 115,629 bp.
contig00207 - 17,056 bp.
contig00208 - 40,050 bp.
contig00209 - 6,552 bp.
contig00210 - 5,044 bp.
contig00211 - 26,648 bp.
contig00212 - 78,484 bp.
contig00213 - 40,754 bp.
contig00214 - 2,069 bp.
contig00215 - 1,179 bp.
contig00216 - 2,247 bp.
contig00217 - 42,957 bp.
contig00218 - 267,932 bp.
contig00219 - 43,700 bp.
contig00220 - 1,458 bp.
contig00221 - 110,208 bp.
contig00222 - 60,167 bp.
contig00223 - 16,253 bp.
contig00224 - 12,676 bp.
contig00225 - 95,342 bp.
contig00226 - 35,030 bp.
contig00227 - 32,059 bp.
contig00228 - 1,055 bp.
contig00229 - 57,810 bp.
contig00230 - 2,806 bp.
```

```
base:
read:
 val:
```

Contig: contig00206 - 115,629 bp.

Base start: 73

```
     contig00206      GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
 ERESLOIO2KJM9H_left  GAAG-CT
    FA7U0GX01CQS0Z    GAAG-CTT-CC-GTACT
    ERESLOIO2MF7PK    GAAG-CTTTCC-GTACTGA-CA
    FA7U0GX01DR2W3    GAAG-CTTTCC-GTACTGA-CAA
    FA7U0GX01DTKGW    GAAGTCTTTCC-GTACTGA-CAAACTGGAACT-GA
    FA7U0GX01D8TX4    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGC
    ERESLOIO2LYC27    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACG
    FA7U0GX01CVDAS    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGG
    FA7U0GX01BLA5U    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGT
    FA7U0GX01A9JEY    GAAG-CTTTCC-GTACTGA-CAAACTGGAACTTGACCTAGCGGCAACATCGCTGACC-ATTAACGGTAA-C
    ERESLOIO1D003I    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
    FA7U0GX01EM3XN    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
    FA7U0GX01D0RVY    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
    ERESLOIO2MOZ3M    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
    FA7U0GX01CI5PF    GAAG-CTTTCC-GTACTGAACAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
    FA7U0GX01CEFZZ    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
    FA7U0GX01E0GQW    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
    FA7U0GX01CB770    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
    FA7U0GX01DLPPI    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
    FA7U0GX01C9KXX    GAAG-CTTTCC-GTACTGA-CAAACTGGA-CT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
    FA7U0GX01BV92Z    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
    FA7U0GX01DFHR2    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
    ERESLOIO203YMV    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
ERESLOIO2L0RMT_right  GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
 ERESLOIO1DLIRY_left  GAAG-CTTTCCA-TACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
ERESLOIO1CEMJK_right  GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCG
    ERESLOIO1D1TU3    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
    ERESLOIO2M9MPT    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACCTAT-AACGGTAA-C
 ERESLOIO1BZQFO_left  GAAG-CTTTCC-GTACT
    ERESLOIO2MH7P1    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
ERESLOIO1CV1O3_right  GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGG
    FA7U0GX01BDORD    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
    ERESLOIO1BETFE    GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
 ERESLOIO1CMXOE_left  GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
ERESLOIO2L988X_right  GAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCTGACC-AT-AACGGTAA-C
 ERESLOIO1CE3FA_left             C--GTACTGA-CAAACTGGAACT-GACC-AGC
 ERESLOIO2OVG05_left             C--GTACTGA-CAAACTGGAACT-GACC-AGC
```

```
Info    E.coli_Assembly1 :   -> Reading flowgrams...
Info    E.coli_Assembly1 :   -> Reading flowgrams...
Info    E.coli_Assembly1 :
Info    E.coli_Assembly1 :   -> Reading flowgrams...
Info    E.coli_Assembly1 :
Info    E.coli_Assembly1 :   -> Reading flowgrams...
Info    E.coli_Assembly1 :   -> Reading flowgrams...
Info    E.coli_Assembly1 :
Info    E.coli_Assembly1 : Assembly computation succeeded at: Fri Aug 15 01:30:47 2008
```

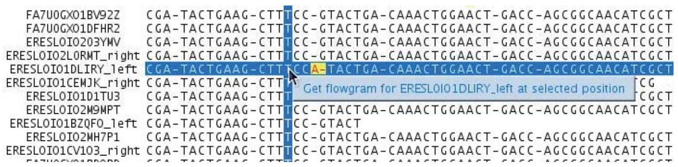
Figure 5–15: The read alignments of a selected contig

The multi-alignment panel has the following features:

- The larger area, on white background, contains the multi-alignment proper:
  - The consensus sequence of the selected contig is shown on top of the panel, gapped for insertions in the aligned reads
  - All the aligned reads included in the contig appear underneath, gapped and showing bases that diverge from the consensus as red dashes on yellow background (some reads near the edges of the contigs may appear very short; these are actually instances of reads that align to two contigs, but the other one was determined to be the boundary of a DNA repeat region and could therefore not be integrated into the contig).
  - The names of all the reads in the multi-alignment appear in a column, on the left side of the multi-alignment area
  - Scroll bars allow you to navigate the alignment manually:
    - Clicking on the arrowheads on either side of a scroll bar scrolls the alignment by one base (horizontal) or one read (vertical) in the direction of the arrowhead
    - Clicking on the light-colored area on either side of the horizontal scroll bar "handle" scrolls the alignment by 40 bases is the corresponding direction
    - Clicking and dragging the "handle" of a scroll bar allows you to move larger distances rapidly
  - Right clicking on a GS read will produce a "Get flowgram for … at selected position" menu item which, if selected, will activate the Flowgrams tab and display the flowgram data for the read; indicating the flow corresponding to the base on which the user clicked to activate the option. This capability is not active for FASTA reads or the contig/consensus sequences themselves, for which no flowgrams are available.
- Above that area are some informational and navigational controls:
  - **Contig**: The name of the contig and its length
  - **Go To**: A data entry field and a **Go To** button allow you to navigate directly to a position of interest: enter a number not larger than the length of the contig in the data entry field, and click the **Go To** button.
  - **Base start**: indicates the first base (relative to the contig consensus sequence) that is currently displayed in leftmost column of the multi-alignment (*i.e.* per scrolling)
  - **Camera icon**: saves an image of the part of the multi-alignment table currently visible on screen, to a file in .png format.
- The Alignment Results tab also features a "Mouse Tracker" area, in the left panel, below the list of contigs. If you pause the mouse pointer over a base in the multi-alignment, the following information is provided about that base:
  - **Base**: the position of that base relative to the contig consensus (all contigs start their numbering at base #1); gaps in the contig consensus are displayed as the following base of the contig
  - **Read**: the name of the read to which this base belongs
  - **Val**: the "value" of the base under the mouse pointer, in that read, *i.e.* A, G, T, or C.

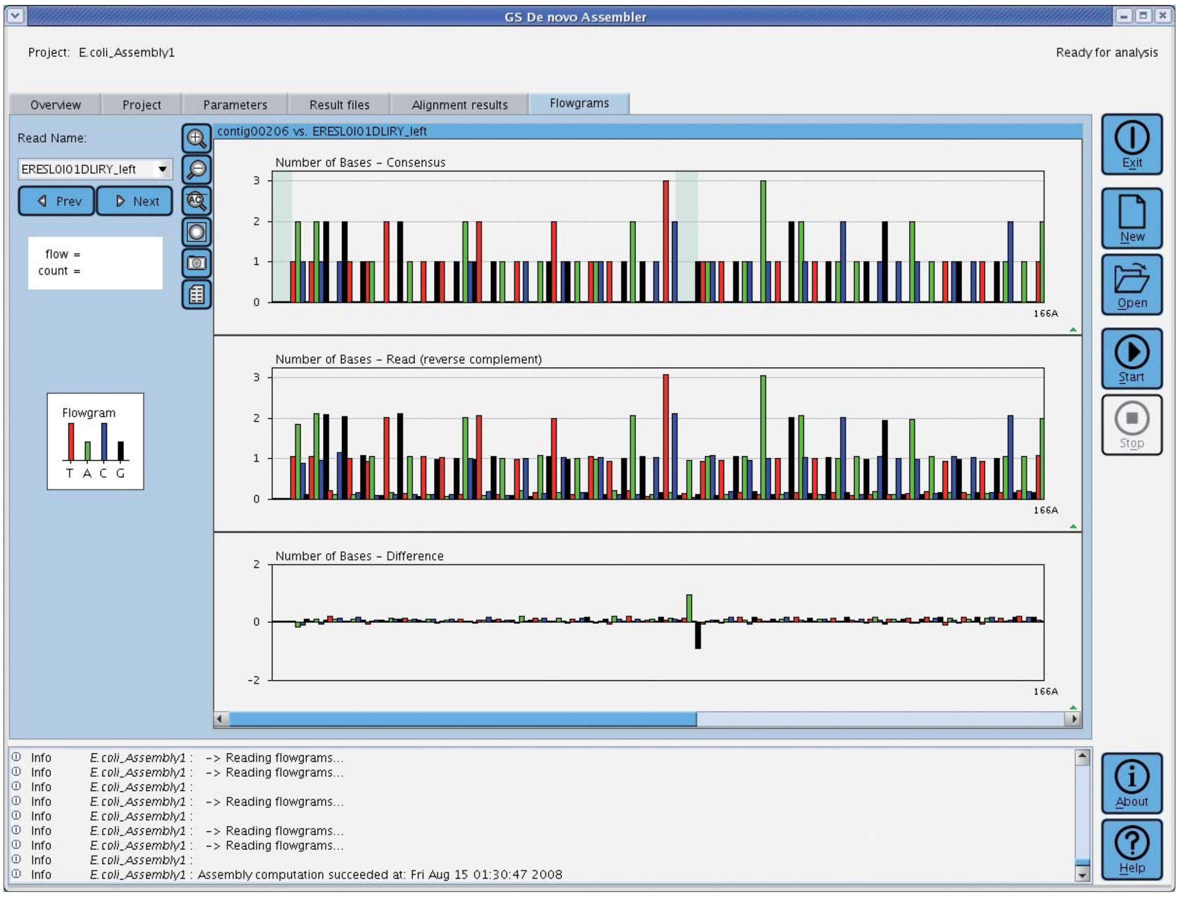
5.2.8.4 Viewing the Flowgram Data: the Flowgram Tab

To view the flowgram from any read in a contig, first select that contig in the Alignment Results tab, and then right-click on any base in the read of interest to open a contextual menu that contains a single item (starting with "Get flowgram:"; Figure 5–16). Selecting that item will display the flowgram for this read, in the Flowgram tab (Figure 5–17). The term "flowgram" is defined in the Glossary (see the *Genome Sequencer FLX Operator's Manual*), and a full description of flowgram views is given in the section on the Amplicon Variant Analysis (section 9.8).



```
     FA7U0GX01BV92Z    CGA-TACTGAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCT
     FA7U0GX01DFHR2    CGA-TACTGAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCT
     ERESLOIO203YMV    CGA-TACTGAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCT
ERESLOIO2L0RMT_right   CGA-TACTGAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCT
  ERESLOIO1DLIRY_left  CGA-TACTGAAG-CTTTCCA-TACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCT
ERESLOIO1CEMJK_right   CGA-TACTGAAG-CTTT Get flowgram for ERESLOIO1DLIRY_left at selected position  CG
     ERESLOIO1D1TU3    CGA-TACTGAAG-CTTT                                                          CGCT
     ERESLOIO2M9MPT    CGA-TACTGAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCT
  ERESLOIO1BZQFO_left  CGA-TACTGAAG-CTTTCC-GTACT
     ERESLOIO2MH7P1    CGA-TACTGAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCT
ERESLOIO1CV103_right   CGA-TACTGAAG-CTTTCC-GTACTGA-CAAACTGGAACT-GACC-AGCGGCAACATCGCT
```

Figure 5–16: Selecting a read for flowgram display with the right mouse button



```
Info   E.coli_Assembly1 :  -> Reading flowgrams...
Info   E.coli_Assembly1 :  -> Reading flowgrams...
Info   E.coli_Assembly1 :
Info   E.coli_Assembly1 :  -> Reading flowgrams...
Info   E.coli_Assembly1 :
Info   E.coli_Assembly1 :  -> Reading flowgrams...
Info   E.coli_Assembly1 :  -> Reading flowgrams...
Info   E.coli_Assembly1 :
Info   E.coli_Assembly1 : Assembly computation succeeded at: Fri Aug 15 01:30:47 2008
```

Figure 5–17: The Flowgram tab of the GS *De Novo* Assembler application

The flowgram view as displayed on the Flowgram tab of the GS *De Novo* Assembler is most similar to the tri-flowgram view described in detail in section 9.8.2 (for the AVA) in that it shows 3 plots:

- The top plot is an idealized flowgram for the contig of the read selected, constructed from the corresponding portion of the contig's consensus sequence, including the insertions of any flow cycle shifts that may be needed to maintain the alignment to the selected read's actual flowgram.
- The center plot is the aligned flowgram for the selected read, including the insertions of any flow cycle shifts that may be needed to maintain the alignment to the consensus idealized flowgram.
- The lower plot is the difference between the read and contig's consensus flowgram.

A small green triangle will indicate the flow corresponding to the base of the read on which you right-clicked when launching the Flowgrams tab view.

The Flowgram tab provides various navigation options and other features:

- The top-left corner of the tab provides two controls that allow the user to change the display to a different read:
  - A pull-down menu allows you to select from a list of all the reads in the contig to which the read currently displayed read belongs (Figure 5–18), and which span the base position of the alignment that was used when initially launching the Flowgrams tab. That base position may be quickly updated by navigating back to the Alignment results tab and selecting a new alignment column by simply left-clicking in that column. Then switch back to the Flowgrams tab (by clicking on the tab). The drop down menu will be updated to include only those reads that have flowgrams and also which intersect with the new column of interest. The read previously displayed in the flowgram view will remain displayed (unless the new column of interest is not spanned by that read), but the green triangle will be updated to point to the flow corresponding to the base in the new column of interest. If the previously displayed read does not span the new column of interest, then the display will automatically go to the first read, topmost in the displayed alignment, that does span the column.
  - A pair of buttons ("Prev" and "Next"; see Figure 5–17, above) allows you to change the display to the previous or the next read of the contig's multiple alignment (in the order shown in the Alignment Results tab).
  - You can also return to the Alignment Results tab and select a different read, from the alignment to the same or to any other contig.

- Various zooming, saving, and mousing functions are available on the Flowgram tab, some via a column of buttons located to the upper-left of the graph area (Figure 5–17). These buttons and functions operate in ways very similar to their counterparts on the AVA (see section 9.8.2 for a full description), and include:
  - Various zooming buttons to zoom in, zoom out, zoom to labels, and zoom to fit.
  - A Snapshot and a Spreadsheet button, to save the data in image (.png) or spreadsheet (.tsv) form, respectively.
  - A mouse tracker area that shows the values for the nucleotide flow under the mouse pointer, when you pause it over the graph area:
    - position of the nucleotide in the alignment,
    - name of the nucleotide, and
    - the "y" value on the plot
      - number of normalized signal from the bases extended during that flow (idealized per the consensus or observed in the read), or
      - the value of the "difference", where bases in the read that are absent in the consensus are positive differences.
  - There is also a free hand zoom function that allows you to zoom in to any area of a graph by drawing a box with the left mouse button. This function has the following unique features (not available in the corresponding GS Run Browser freehand zoom in functions) to facilitate the comparison between the three plots:
    - zooming on the consensus or on the read plots will apply to the Y axis of both those plots, and the X-axis of all three plots (including the difference plot) (Figure 5–19).
    - zooming on the difference plot zooms its Y axis alone, but zooms the X axis of all three plots (not shown).

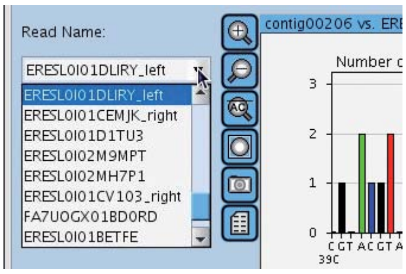
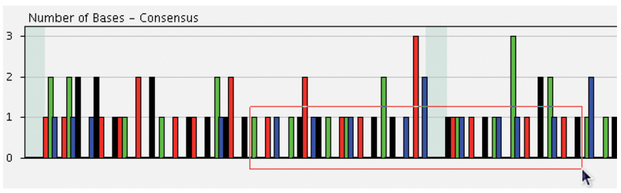
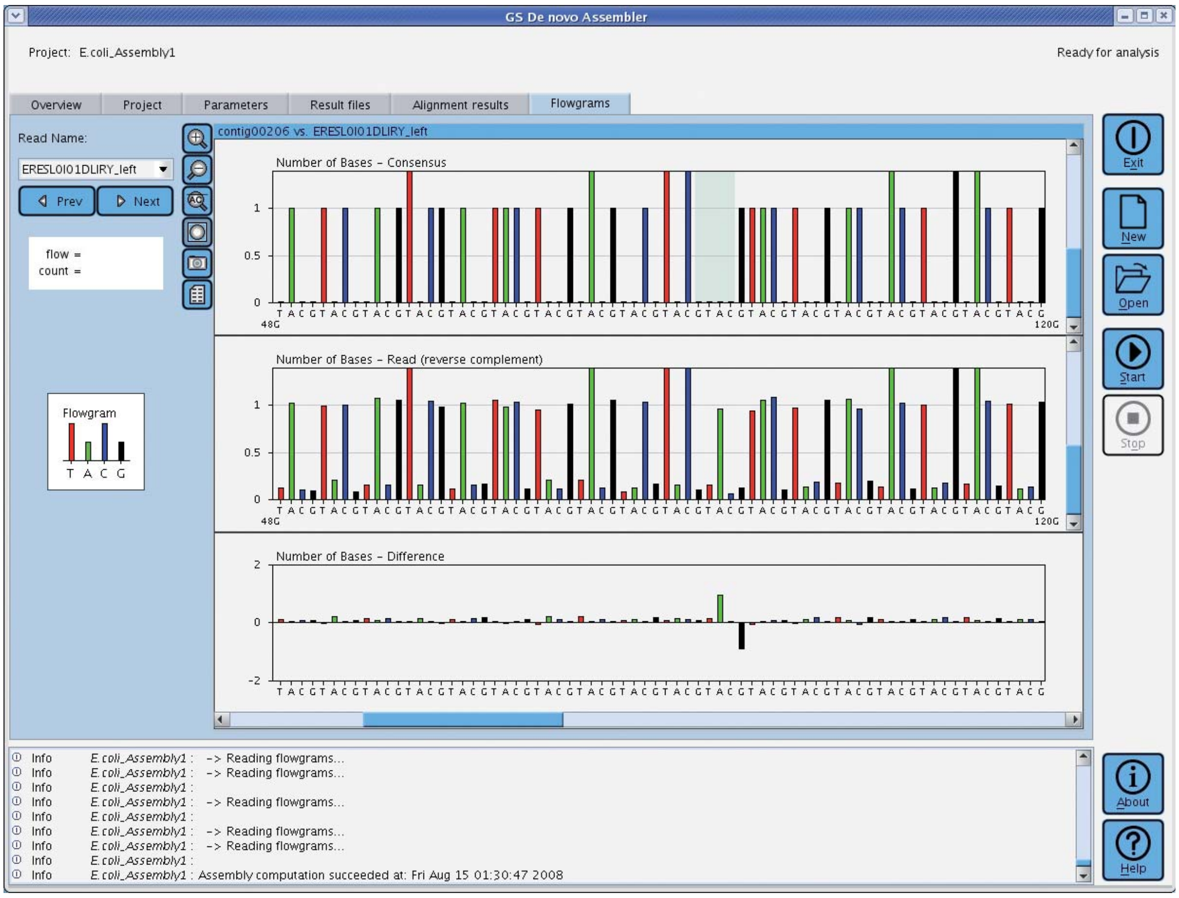


Figure 5–18: Selecting a read from the drop down menu in the Flowgram tab

A



B



```
Info    E.coli_Assembly1 :  -> Reading flowgrams...
Info    E.coli_Assembly1 :  -> Reading flowgrams...
Info    E.coli_Assembly1 :
Info    E.coli_Assembly1 :  -> Reading flowgrams...
Info    E.coli_Assembly1 :
Info    E.coli_Assembly1 :  -> Reading flowgrams...
Info    E.coli_Assembly1 :  -> Reading flowgrams...
Info    E.coli_Assembly1 :
Info    E.coli_Assembly1 : Assembly computation succeeded at: Fri Aug 15 01:30:47 2008
```

Figure 5–19: Using the mouse to zoom into a sub-region of a flowgram

5.3 Using the GS *De Novo* Assembler Via the Command Line Interface

Assemblies can be performed either using the Graphical User Interface described above, or using the Linux command line, described in this section.

5.3.1 One-Step Assembly: the runAssembly Command

If all the reads to be assembled are available at once, the GS *De Novo* Assembler application can process them with a single command, which has the following command line structure:

```
runAssembly [options] [MIDList@]filedesc…
```

…where:

- "[options]" are zero or more of the command line options (listed below),
- each "filedesc" is one of the following:
  - `sfffilename or`
  - `[regionlist:]datadir or`
  - `readfastafile`
- and each "MIDList" is a multiplexing information string used to filter the set of file reads to be used in the assembly (see the third Note below for the format of the MIDList information). If MIDs were used in the generation of the datafile, then an MIDList string must be specified in order for the assembler to properly handle the file's reads.

External files:

- GS Read Data files (SFF files) are not actually copied to the project directory. Rather, a symbolic link to the file is created and placed in the project's sff directory. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies whether the files are added via the GUI or the command line.
- FASTA files are not actually copied to the project directory. Rather, the file path is simply recorded in the project setup. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies to all instances of FASTA files that can be added to an Assembly project, whether done via the GUI or the command line:
  - FASTA reads files, including Sanger reads
  - Trimming database files
  - Screening database files

Unlike with the GS Read Data files, no symbolic link to the file is created for FASTA files.

Command	Description
runAssembly	Complete set of contig assembly algorithms, beginning with the read flowgrams of one or more sequencing Runs (which may include one or more Paired End Runs), and ending with the final consensus sequence of the sample DNA library, assembled *de novo* into one or more contigs (scaffolded with Paired End).

There are a large number of command line options to this command, only a subset of which are commonly used or important. The following are the most common command line arguments:

Option	Description
[-nrm]	An option to allow the assembly results to be used for further project-based assembly, by "not removing" the project files and folders at the end of the assembly processing (as described in section 5.3.2; see also the first Note below)
[-o dir]	An option to specify the directory where all the output files should be written. If this option is omitted, the program will generate the 'P_' directory in the current working directory at the time the command is launched.
[-no]	With the "no output" option, the program will perform all the alignments for the assembly, but will not generate consensus/contig output. This is useful to avoid wasting computing time, *e.g.* when the user knows that more Runs will be added to the project. (However, the project and alignment metrics are still output.)
[-large]	An option to tell the assembler that it is working with a large dataset of reads from a large or complex genome (fungal or eukaryotic). This option short-circuits some of the computationally expensive algorithms, ensuring that the computation finishes in a reasonable amount of time.
[-noace]	With this option, the ACE file is not generated.
[-ace]	With this option, the command generates a single ACE file containing all the contigs produced by the assembly.
[-acedir]	With this option, the command generates an "ace" sub-directory containing a separate ACE file for each contig in the assembly (see Figure 1–4).
[-consed]	With this option, the command generates a "consed" sub-directory containing all the directories and files needed to run the consed application on the results of the assembly (see Figure 1–4).
[-pair]	This option outputs the pairwise overlaps used in the assembly, in simple text format. By default, these alignments are not output.
[-pairt]	This option outputs the pairwise overlaps used in the assembly, in tab-delimited format. By default, these alignments are not output.
[-p (sfffilename or [regionlist:]datadir)]	An option to specify that the SFF file or Data Processing folder whose path immediately follows contains the results of a Paired End sequencing Run, to be used to order and orient the contigs of the Shotgun sequencing Run(s) (if present). Multiple Paired End Runs may be included in the Analysis. The –p option forces the software to consider a data set as Paired End; all Runs whose reference is not preceded by "–p" will be determined as being Paired End or Shotgun sequencing Runs, by virtue of the presence or absence of the Paired End linker sequence in the read data they contain: if at least 25% of the first 500 reads in the file (or 25% of all the reads if there are fewer than 500 reads in the file) contain the linker, the file is recognized as a Paired End file. The details for Shotgun Run datasets (below) also apply to Paired End Run datasets.
[--help]	Displays a usage line and short description of the command.
[--version]	Displays the current software version of the assembler.

Argument	Description
[*MIDlist@*] (sfffilename or [*regionlist:*]datadir or readfastafile)	Path to an SFF file or to a Data Processing folder containing the results of a Shotgun sequencing Run, created by the GS Run Processor application (either on a Genome Sequencer FLX Instrument or on a DataRig or cluster), and containing the SFF files for the regions of the sequencing Run (in the "sff" sub-folder; see Figure 1–3). There can be any number of SFF files and/or Data Processing folders given as argument (and thus included in the assembly), an optional list of regions may be prepended to each data directory argument (see the second Note below), and a multiplexing information string can be prepended to each file/data-directory argument (see the third Note below). [If SFF files are not present in a Data Processing folder (*e.g.* for a Run whose data has been processed with a version of the Genome Sequencer System software anterior to 1.0.52), the signal processing step of the GS Run Processor application must be rerun on the Data Processing folder.] Alternatively, path to a FASTA file containing Sanger reads to be included in the assembly. These reads (or contigs) must be shorter than 2000 bases per read (longer sequences are ignored) and longer than 50 bases (shorter sequences are ignored). See section 13.2 for input FASTA file format requirements.

Input read size constraints: These reads (or contigs) input for assembly computation must be shorter than 2000 bases per read (longer sequences are ignored) and longer than 50 bases (shorter sequences are ignored).

A second set of arguments are less commonly used, but are useful to configure the output generated by the command or to adjust the execution of the computation:

Option	Description
[-a #]	This option sets the minimum length for a contig to appear in the 454AllContigs.fna file. The default is 100 bases. Setting this parameter to zero is another way to tell the application to output the 454ContigGraph.txt file (besides the –g option, described below).
[-ar]	An option telling the assembler to output the full "raw" read sequence when generating an ACE file or consed folder.
[-at]	An option telling the assembler to output only the "trimmed" sequences for each read when generating an ACE file or consed folder.
[-e #]	An option telling the assembler that the "expected depth" of the data is at a certain level. The assembler has been optimized for datasets in the 10–50× oversampling size, and this option helps the assembler with datasets that have a higher oversampling level. A value of 0 resets the assembler computation to use its default algorithms.
[-g]	This option outputs the "contig graph" giving the branching structure of the contiguous alignments in the assembler. By default, the contig graph information is not output. This option also resets the "AllContigs" length to 0 (a requirement for the contig graph file).
[-l #]	This option sets the minimum length for a contig to appear in the 454LargeContigs.fna file. The default is 500 bases.
[-mcf *filename*]	This option specifies a different MID configuration file to be used by the assembly for decoding the multiplex information appearing on the command line.
[-nobig]	This option turns off the generation of the "big" output files, namely the ACE/consed files, the 454PairAlign.txt file and the 454AlignmentInfo.tsv file.
[-notrim]	This option turns off the default quality and primer trimming that the GS *De Novo* Assembler performs on all input reads (454 Sequencing System reads and Sanger reads).
[-v fastafile] or [-vt fastafile]	This option specifies a "vector trimming database", or FASTA file of sequences to be used to trim the ends of input reads (for cloning vectors, primers, adapters or other end sequences).
[-vs fastafile]	This option specifies a "vector screening database", or FASTA file of sequences to be used to screen the input reads for contaminants. Reads that completely align against the screening database are trimmed completely (so that it is not used in the computation); if only part of a read aligns with the screening database, no action is taken.

A final set of options exist, that can be used to adjust the assembly algorithm parameters (equivalent to the Parameter tab settings in the GUI), or minor options that are more rarely used. See section 5.2.4 for a more complete description of the Parameter tab settings. The assembly computation has been optimized to give its best results using the default parameters, but these are provided for users who wish to experiment with adjusting parameters for their data sets.

Option	Description
[-ais]	This option sets the "Alignment identity score" parameter.
[-ads]	This option sets the "Alignment difference score" parameter.
[-m]	An option telling the GS *De Novo* Assembler to keep all sequence data "in memory" throughout the computation. This option can speed up the execution of the assembler, but requires the use of more computer memory.
[-mi]	This option sets the "Minimum overlap identity" parameter.
[-ml]	This option sets the "Minimum overlap length" parameter.
[-nor]	This option turns off the automatic "rescore" function that generates new quality scores for each SFF read using the new quality score algorithms.
[-sc]	This option sets the "Seed count" parameter.
[-sl]	This option sets the "Seed length" parameter.
[-ss]	This option sets the "Seed step" parameter.
[-ud]	An option telling the GS *De Novo* Assembler to treat each read as a separate read, and not group them into duplicates for assembly or consensus calling.

- The runAssembly command is actually a wrapper program around the newAssembly and related commands used in project-based assembly (see section 5.3.2). After the last project assembly command is completed, runAssembly then transforms the project files and folders into this one-step form (which more closely matches the output structure of older versions of the "Assembly" application). The actions taken are:
  - All the assembly output files are moved up from the assembly sub-directory into the main folder
  - All internal data files in the assembly sub-directory are deleted
  - The 454AssemblyProject.xml and 454Project.xml files are deleted
  - The assembly sub-directory is deleted

  If the –nrm option is given, runAssembly does not perform this transformation, and the resulting folder can be used for further incremental assembly (the structure of the files/folders will match that of an assembly project).

- The path given for any of the data directories may be prepended with an optional list of regions, separated from the path by a colon. For example:

  1-3,4,6-8:D_2005_01_01_01_01_01_testuser_runImagePipe

  The list of regions must have the following format:
  - It must be a comma-separated list, ending with a colon.
  - Each element of the list can either be a single region number or a dash-separated range of region numbers.
  - Duplicate regions may be specified, and the regions may be specified in any order, but duplicates will be removed and the regions will be processed and reported in numerical order.
  - No spaces or other characters are allowed in the string.

If the region list is not present for a data directory path, all regions of the Run for that directory will be used in the assembly. Also, any combination of explicit SFF files and data directory paths (with optional region lists) may be specified on the command line. For each data directory path given, the runAssembly command will read the existing SFF files in the "sff" sub-directory of the data directory. If SFF files are not present in a data directory (*e.g.* for a Run whose data has been processed with a version of the Genome Sequencer System software anterior to 1.0.52), the signal processing step of the GS Run Processor application must be rerun on the Data Processing folder.

The path for any filename or data directory name can be prepended with a multiplex information string, separated from the filename/directory-string by an "@" sign. Multiplex information strings should be used when multiple samples are prepared using different initial "tag" sequences (such as those provided by the GS Multiplex Identifiers (MID) Kits), then mixed together for sequencing. These sequences are used to segragate the reads from each sample, by matching the initial bases of the reads to the differentiating tag sequences used in the preparation of the libraries. If a multiplex string is given, the GS *De Novo* Assembler will automatically match the 5' bases of each read against the multiplex sequences, and will only use the reads that match the specified sequences (and reset the trim point of those reads so that the multiplex sequences are not used in the assembly itself).

Three examples of multiplex information strings are the following:

mid2@myreads.sff
GSMIDs:mid1,mid4,mid8@/home/xxx/morereads.sff
aattctc/1@1-3,4,6-8:D_2005_01_01_01_01_01_testuser_runImagePipe

The first example gives "myreads.sff" as the input SFF file and tells the assembler to only use reads whose initial 5'-sequence matches the "mid2" MID (as listed in the MID configuration file). The second example tells the GS *De Novo* Assembler to use the file "/ home/xxx/morereads.sff" and filter that file, using only the reads starting with the mid1, mid4 and mid8 sequences, as defined in the "GSMIDs" MID set. The third example tells the GS *De Novo* Assembler to read regions 1, 2, 3, 4, 6, 7 and 8 of the "D_2005_..." sequencing Run, but only use the reads whose 5'-end matches the DNA sequence "AATTCTC" with 1 error or less.

The format of the multiplex information string is the following:

```
[setname:]mid[/#](,mid[/#]…)@
```

…where

- the "setname" is an optional name of an MID set found in the MID configuration file and can be given in uppercase or lowercase letters (the matching is case-insensitive);
- each "mid" is either an MID name found in the configuration file or a DNA sequence; and
- each "#" is an optional allowed number of errors.

In other words, the format consists of:

- An optional MID set name, followed by a colon
  - This name must occur in the MID configuration file
- One or more MID names or DNA sequences, separated by commas. Each name/sequence can be followed by an optional slash and number of allowed errors.
  - The MID names must occur in the MID configuration file, and if the same MID name occurs in multiple MID sets in the file, then the MID set name must be given. (If the MID name is unique over all the names in the file, then no MID set name is required.)
- A "@" sign to end the multiplex information string.

No spaces are allowed in the multiplex information string, and no colons, slashes or "@" signs are allowed in the MID set names or MID names.

The off-instrument installation contains a default MID configuration file, found by default at /usr/local/rig/config/MIDConfig.parse. This file is read by the GS *De Novo* Assembler and used to match MID set names and MID names with their multiplexing information. Users can edit this file to add their own MID sets (following the format and syntax described in the file), or can copy this file to create their own separate MID configuration file (and then use the "-mcf" option to specify that as the MID configuration file to be used). See section 13.4.15 for the contents of the MIDConfig.parse file.

Some of the runAssembly command options listed above are mutually exclusive, for obvious functional reasons. Specifically:

- Don't use -no or -nobig together with any of the following: -ace, -acedir, consed, -pair, or -pairt
- Use no more than one of each of the following sets:
  - -noace, -ace, -acedir, or -consed
  - -pair (-p) or -pairt (-pt)
  - -ar or -at
  - -a # or -g
  - -vs or -novs
  - -vt (-v) or -novt (-nov)

5.3.2 Project-Based Assembly: the newAssembly and Related Commands

This section describes the main commands of the GS *De Novo* Assembler application, to be used when one or more Runs are to be assembled as part of an assembly project. These commands allow you to add and remove Runs over time and incrementally update the assembly. With these commands, the execution of the assembly algorithms and generation of the results can be controlled by command line options and configuration parameters (paralleling the equivalent controls in the GUI application). Such incremental assembly can be useful when, for example, you want to see intermediate results on existing sequencing Runs to determine if you need to carry out further Runs to reach a desired depth of coverage (or just to monitor the project).

Four commands are used in an assembly project: newAssembly, addRun, removeRun and runProject. These are described in the subsections below.

5.3.2.1 The newAssembly Command

The newAssembly command is used to initiate an assembly project and set an assembly project folder to contain the project data (see Figure 1–6). Its command line structure is:

```
newAssembly [dir]
```

Command	Description
newAssembly	Creates an assembly "project" folder where the project-based assembly is performed and the results are stored.

Argument	Description
[dir]	An optional argument to set the directory where all the project input, status and output files should be written. If a directory is specified, that directory is used (created if it does not already exist) as the project directory. If the command is executed with no argument, it creates a new "P_..._runAssembly" directory as the project directory, in the current working directory at the time the command is launched. Note: The directory name may not begin with a dash ('-').

5.3.2.2 The addRun Command

The addRun command is used to add Read Data sets to existing assembly projects. Its command line structure is:

```
addRun [options] [MIDList@]filedesc…
```

…where:

- "[options]" are zero or more of the command line options (listed below),
- each "filedesc" is one of the following:
  - `sfffilename` or
  - `[regionlist:]datadir` or
  - `readfastafile`
- and each "MIDList" is a list of multiplexing information used to filter the set of file reads used in the assembly (see the Note in section 5.3.1 for the format of the MIDList information). If MIDs were used in the generation of the datafile, then an MIDList string must be specified in order for the GS *De Novo* Assembler to properly handle the file's reads.

External files:

- GS Read Data files (SFF files) are not actually copied to the project directory. Rather, a symbolic link to the file is created and placed in the project's sff directory. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies whether the files are added via the GUI or the command line.
- FASTA files are not actually copied to the project directory. Rather, the file path is simply recorded in the project setup. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies to all instances of FASTA files that can be added to an Assembly project, whether done via the GUI or the command line:
  - FASTA reads files, including Sanger reads
  - Trimming database files
  - Screening database files

Unlike with the GS Read Data files, no symbolic link to the file is created for FASTA files.

Command	Description
addRun	Adds the reads from one or more Read Data sets to an existing assembly project.

Option	Description
[-p]	An option to specify that the Read Data given on the rest of the command line should be treated as the results of Paired End sequencing Runs. (Note: Unlike the runAssembly command (see section 5.3.1), this –p option only needs to be specified once, and applies to all Read Data sets on this execution of addRun.) If the "-p" option is not given, each Read Data set given in argument will be determined as being Paired End or Shotgun sequencing Runs, by virtue of the presence or absence of the Paired End linker sequence in the read data they contain: if at least 25% of the first 500 reads in the file (or 25% of all the reads if there are fewer than 500 reads in the file) contain the linker, the file is recognized as a Paired End file. The details for Shotgun Run datasets (below) also apply to Paired End Run datasets.
[-lib libname]	This option specifies that the default Paired End library for each of these files should be the string "libname". It can be used to group together the Paired End reads from different files, so that they are all used together in calculating a mean distance between the halves of each Paired End pair of reads. (By default, each SFF file is treated as a separate library, and a separate mean Paired End distances is calculated.)
[-mcf *filename*]	This option specifies an MID configuration file (other than the default file) to be used by the assembly for decoding the multiplex information appearing on the command line.

Argument	Description
[dir]	An optional argument (must be first in the list of arguments) to identify the project folder for the command. If the command is executed with no "directory" argument, it verifies whether the current working directory is either an assembly project folder or the "assembly" sub-folder inside a assembly project folder, and uses that if so. This way, users can create a assembly project and then simply change directory ("cd") into it, and all the commands will work on the project they are in.
[MIDList@] (sfffile or [*regionlist*:] datadir or readfastafile)	Path to an SFF file or to a Data Processing folder containing the results of a Shotgun sequencing Run or Paired End sequencing Run, created by the GS Run Processor application (either on a Genome Sequencer FLX Instrument or on a DataRig or cluster), and containing the SFF files for the regions of the sequencing Run (in the "sff" sub-folder; see Figure 1–3). There can be any number of SFF files and/or Data Processing folders given as argument (and thus included in the assembly), an optional list of regions may be prepended to each data directory argument (see Note in section 5.3.1), and a multiplexing information string can be prepended to each argument (see Note in section 5.3.1). [If SFF files are not present in a Data Processing folder (*e.g.* for a Run whose data has been processed with a version of the Genome Sequencer System software anterior to 1.0.52), the signal processing step of the GS Run Processor application must be rerun on the Data Processing folder.] Alternatively, path to a FASTA file containing Sanger reads to be included in the assembly. These reads (or contigs) must be shorter than 2000 bases per read (longer sequences are ignored) and longer than 50 bases (shorter sequences are ignored). See section 13.2 for input FASTA file format requirements.

- The addRun command can be executed multiple times for an assembly project (in any combination with the removeRun and runProject commands). When addRun is executed, it adds (not "resets") the given Runs/regions or SFF files to the list of sequencing data used in the project. (It does not reset the list of sequence data). The reads used in the assembly (runProject, section 5.3.2.4, below) are the union of the data from all executions of addRun for the project.
- The commands addRun, removeRun, and runProject can be executed in any combination and in any order, in an assembly project.

Input read size constraints: These reads (or contigs) input for assembly computation must be shorter than 2000 bases per read (longer sequences are ignored) and longer than 50 bases (shorter sequences are ignored).

5.3.2.3 The removeRun Command

The removeRun command is used to remove Read Data sets from existing assembly projects. Its command line structure is:

```
removeRun [dir] (sffname or readfastafilepath)…
```

Command	Description
removeRun	Removes the reads of one or more Read Data sets from an existing Assembly project.

Argument	Description
[dir]	An optional argument (must be first in the list of arguments) to identify the project folder for the command. If the command is executed with no “directory” argument, it verifies whether the current working directory is either an assembly project folder or the “assembly” sub-folder inside an assembly project folder, and uses that if so. This way, users can create an assembly project and then simply change directory (“cd”) into it, and all the commands will work on the project they are in.
sffname or readfastafilepath	The name of an SFF file currently included in the project, as given in the sff sub-directory of the project folder; or the path to an input read FASTA file, as given in the 454AssemblyProject.xml file for the project.

- This command is more conveniently carried out from the GUI application. When called from the command line, it requires the SFF file name(s) *given in the project sff sub-directory* or the FASTA file name(s) *given in the project configuration file 454AssemblyProject.xml*, which may not match the original names of the corresponding files or may not be known to the user, especially if the software had to rename any of them to ensure name uniqueness (see section 5.4.2.13 for details on this).

- The execution of this command does not physically remove the file(s) from the project sff sub-directory or from any existing assembly. The file(s) and the reads they contain are only marked for removal by the removeRun command, and are actually removed only the next time the project is computed (via the runProject command).

- The commands addRun, removeRun, and runProject can be executed in any combination and in any order, in an assembly project.

5.3.2.4 The runProject Command

The runProject command performs the actual assembly computation for a project and generates the results of the assembly. Its command line structure is:

```
runProject [options] [dir]
```

Command	Description
runProject	Executes the GS *De Novo* Assembler application algorithms, assembling any new Runs present in the project directory to the existing assembly and generating the combined results for all the Runs currently included in the project.

There are a large number of command line options to this command, only a subset of which are commonly used or important. The following are the most common command line arguments:

Option	Description
[-r]	This option restarts the assembly computation anew, removing any incremental assembly results that may exist in the project directory.
[-no]	With the "no output" option, the program will perform all the alignments for the assembly, but will not generate consensus/contig output. This is useful to avoid wasting computing time, *e.g.* when the user knows that more Runs will be added to the project. (However, the project and alignment metrics are still output.)
[-large]	An option to tell the assembler that it is working with a large dataset of reads from a large or complex genome (fungal or eukaryotic). This option short-circuits some of the computationally expensive algorithms, ensuring that the computation finishes in a reasonable amount of time.
[-noace]	With this option, the ACE file is not generated.
[-ace]	With this option, the command generates a single ACE file containing all the contigs produced by the assembly.
[-acedir]	With this option, the command generates an "ace" sub-directory containing a separate ACE file for each contig in the assembly (see Figure 1–6).
[-consed]	With this option, the command generates a "consed" sub-directory containing all the directories and files needed to run the consed application on the results of the assembly (see Figure 1–6).
[-p] or [-pair]	This option outputs the pairwise overlaps used in the assembly, in simple text format. By default, these alignments are not output.
[-pt] or [-pairt]	This option outputs the pairwise overlaps used in the assembly, in tab-delimited format. By default, these alignments are not output.
[--help]	Displays a usage line and short description of the command.
[--version]	Displays the current software version of the assembler.

Argument	Description
[dir]	An optional argument (follows any options) to identify the project folder for the command. If the command is executed with no "directory" argument, it verifies whether the current working directory is either an assembly project folder or the "assembly" sub-folder inside an assembly project folder, and uses that if so. This way, users can create an assembly project and then simply cd into it, and the commands will all work on the project they are in.

External files: FASTA files are not actually copied to the project directory. Rather, the file path is simply recorded in the project setup. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies to all instances of FASTA files that can be added to an Assembly project, whether done via the GUI or the command line:

- FASTA reads files, including Sanger reads
- Trimming database files
- Screening database files

Unlike with the GS Read Data files, no symbolic link to the file is created for FASTA files.

A second set of arguments are less commonly used, but are useful to configure the output generated by the command or to adjust the execution of the computation:

Option	Description
[-a #]	This option sets the minimum length for a contig to appear in the 454AllContigs.fna file. The default is 100 bases. Setting this parameter to zero is another way to tell the application to output the 454ContigGraph.txt file (besides the –g option, described below).
[-ar]	An option telling the GS *De Novo* Assembler to output the full "raw" read sequence when generating an ACE file or consed folder.
[-at]	An option telling the GS *De Novo* Assembler to output only the "trimmed" sequences for each read when generating an ACE file or consed folder.
[-ad]	An option telling the GS *De Novo* Assembler to use the default method of outputting reads when generating an ACE file or consed folder (which, for assembly, is to use the trimmed sequences).
[-e #]	An option telling the GS *De Novo* Assembler that the "expected depth" of the data is at a certain level. The assembler has been optimized for datasets in the 10–50× oversampling size, and this option helps the assembler with datasets that have a higher oversampling level. A value of 0 resets the GS *De Novo* Assembler computation to use its default algorithms.
[-g]	This option outputs the "contig graph" giving the branching structure of the contiguous alignments in the assembler. By default, the contig graph information is not output. This option also resets the "AllContigs" length to 0 (a requirement for the contig graph file).
[-l #]	This option sets the minimum length for a contig to appear in the 454LargeContigs.fna file. The default is 500 bases.
[-nobig]	This option turns off the generation of the "big" output files, namely the ACE/consed files, the 454PairAlign.txt file and the 454AlignmentInfo.tsv file.
[-notrim]	This option turns off the default quality and primer trimming that the GS *De Novo* Assembler performs on all input reads (454 Sequencing System reads and Sanger reads).
[-trim]	This option turns on the default quality and primer trimming that the GS *De Novo* Assembler performs on all input reads (454 Sequencing System reads and Sanger reads). This is the default for assembly.
[-v fastafile] or [-vt fastafile]	This option specifies a "vector trimming database", or FASTA file of sequences to be used to trim the ends of input reads (for cloning vectors, primers, adapters or other end sequences).
[-nov] or [-novt]	This option specifies no trimming database should be used, removing a vector database that had been specified in an earlier execution of runProject.
[-vs fastafile]	This option specifies a "vector screening database", or FASTA file of sequences to be used to screen the input reads for contaminants. Reads that completely align against the screening database are trimmed completely (so that they are not used in the computation); if only part of a read aligns with the screening database, no action is taken.
[-novs]	This option specifies no screening database should be used, removing a screening database that had been specified in an earlier execution of runProject.

A final set of options exist, that can be used to adjust the assembly algorithm parameters (equivalent to the Parameter tab settings in the GUI), or minor options that are rarely used. See section 5.2.4 for a more complete description of the parameter tab settings. The assembly computation has been optimized to give its best results using the default parameters, but these are provided for users who wish to experiment with adjusting parameters for their data sets.

Option	Description
[-ais]	This option sets the "Alignment identity score" parameter.
[-ads]	This option sets the "Alignment difference score" parameter.
[-m]	An option telling the assembler to keep all sequence data "in memory" throughout the computation. This option can speed up the execution of the GS *De Novo* Assembler, but requires the use of more computer memory.
[-mi]	This option sets the "Minimum overlap identity" parameter.
[-ml]	This option sets the "Minimum overlap length" parameter.
[-nor]	This option turns off the automatic "rescore" function that generates new quality scores for each SFF read using the new quality score algorithms.
[-sc]	This option sets the "Seed count" parameter.
[-sl]	This option sets the "Seed length" parameter.
[-ss]	This option sets the "Seed step" parameter.
[-ud]	An option telling the GS *De Novo* Assembler to treat each read as a separate read, and not group them into duplicates for assembly or consensus calling.

- The commands addRun, removeRun, and runProject can be executed in any combination and in any order, in an assembly project.

- Some of the runProject (for Assembly) options listed above are mutually exclusive, for obvious functional reasons. Specifically:
  - Don't use -no or -nobig together with any of the following: -ace, -acedir, consed, -pair, or –pairt
  - Use no more than one of each of the following sets:
    - -noace, -ace, -acedir, or –consed
    - -pair (-p) or -pairt (-pt)
    - -ar, -at, or –ad
    - -a # or –g
    - -trim or –notrim
    - -vs or –novs
    - -vt (-v) or -novt (-nov)

5.4 GS *De Novo* Assembler Application Output

The GS *De Novo* Assembler application uses a folder on the file system to hold the assembly project information (whether the assembly of the reads, the computation, is carried out in project-based mode through the GUI application or through the newAssembly and related commands) and to hold the output files generated during the assembly computation. The contents of this folder and the names of the files generated by the application are the same for any mapping project or output folder.

5.4.1 Project/Output Folder Structure

Since the assembly computation is often performed on a pool of sequencing Runs (or Read Data files) rather than on any single Run, the result files it generates are not deposited in a Run folder. Two general cases exist.

- If the assembly is performed using the "one-step" command runAssembly, a folder with a 'P_' prefix (for 'P'ost-Run Analysis) is created in the user's current working directory on the DataRig at the time the application is launched, or written to a directory specified by the user on the command line (or its GUI equivalent), to contain these files (see Figure 1–4). The name structure for this folder is as follows (see also section 5.4.2):

  "P_yyyy_mm_dd_hh_min_sec_runAssembly"

  The "-o" option can be specified on the command line to change the directory where the output files should be written (see section 5.3.1). If the specified directory exists, the output files will be written to that directory. If the specified directory does not exist, the program will create it, if possible.
- For "project-based" assembly, using the GUI application or using the newAssembly and related commands, the output is placed in a "project" folder (see Figure 1–6 and section 5.3.2). You can specify any name for your project folder; it will be recognized as a project folder by virtue of the "454Project.xml" file that will be automatically created within it. If you do not specify a directory name on the newAssembly command line, the software will use the same default name as the runAssembly command:

  "P_yyyy_mm_dd_hh_min_sec_runAssembly"

As shown in Figure 1–6, the project-based assembly folder contains additional folders and files that mark the folder as a project folder and that store configuration information and internal data for the GS *De Novo* Assembler application. A project folder comprises two sub-folders: an "assembly" sub-folder which contains the project state and output files; and an "sff" sub-folder containing the copies and/or symbolic links for the SFF files used as input to the assembly project. An additional file, "454Project.xml", is also placed there to identify the folder as a 454 project folder (and as a place holder for later integration, where mapping, assembly and/or amplicon "projects" can all reside in a single project folder and interact with the same data set(s), in an integrated fashion).

5.4.2 GS *De Novo* Assembler Output Files

The GS *De Novo* Assembler application creates the files listed in Table 5–1. Each time an assembly computation is carried out (no matter how many sequencing Runs or analysis regions are included), the files produced will have the exact fixed names shown in Table 5–1, with the exception of the *.ace files (if individual, per contig; see section 5.4.2.12) and files in the "sff" sub-directory (see section 5.4.2.13, below). You can view the output files from the sample data sets provided in the accompanying DVD, using the GS Run Browser application.

File Name Structure	Definition
454AllContigs.fna	FASTA file of all the consensus basecalled contigs longer than 100 bases.
454AllContigs.qual	Corresponding Phred-equivalent quality scores for each base in the consensus contigs in 454AllContigs.fna.
454LargeContigs.fna	FASTA file of all the "large" consensus basecalled contigs contained in 454AllContigs.fna. What constitutes a large contig is currently set to 500 bp.
454LargeContigs.qual	Corresponding Phred-equivalent quality scores for each base in the "large" consensus contigs in 454LargeContigs.fna.
454NewblerMetrics.txt	File providing various assembly metrics, including the number of input Runs and reads, the number and size of the large consensus contigs and the number of all consensus contigs.
454NewblerProgress.txt	A text log of the messages sent to standard output during the assembly computation.
454AlignmentInfo.tsv	Tab-delimited file giving position-by-position consensus base and flow signal information.
454ContigGraph.txt	A text file giving the "contig graph" that describes the branching structure that relates the contigs produced by the assembler (only produced when using the –g option or when using the –a option with a value of 0).
454PairAlign.txt	A text file giving the pairwise alignment(s) of the overlaps used in the assembly computation (only produced when using the –p option [or –pt option for the tab-delimited version of the file]).
454ReadStatus.txt	Tab-delimited text file providing a per-read report of the status of each read in the assembly.
454TrimStatus.txt	Tab-delimited text file providing a per-read report of the original and revised trimpoints used in the assembly.
454Contigs.ace or ace/*ContigName*.ace or consed/…	ACE format file that can be loaded by third-party viewer programs that understand the ACE format, or consed sub-directory that can be loaded by the consed software. This can be a single file for the entire project or a folder containing individual files for each contig in the assembly (see Figure 1–6).
sff/*sfffilename*	The SFF file(s) used as input by the assembly computation.
454Scaffolds.fna[1]	FASTA file of the concatenated contig sequences that were scaffolded as a result of Paired End analysis. The contigs are separated by a number of 'N' corresponding to the estimated size of the gap between them (but with a minimum of 20 N's to ensure the separation of the contigs).
454Scaffolds.qual[1]	Corresponding Phred-equivalent quality scores for each nucleotide in the scaffolded consensus contigs in 454Scaffolds.fna.

▶▶▶

File Name Structure	Definition
454Scaffolds.txt[1]	An AGP file (NCBI's format for describing scaffolds of contigs) containing the scaffold layout.
454PairStatus.txt[1]	Tab-delimited text file providing a per-pair report of the location and status of how each Paired End pair of reads were used in the assembly.
454TagPairAlign.txt[1]	A text file giving the pairwise alignments used in the assembly computation for Paired End reads shorter than 50 bases (which are not part of the overlap computation, but are mapped to the consensi in a later computation step).

Table 5–1: Files generated by the GS *De Novo* Assembler application. Words in *italics* are generic; real file names would have the actual name of the contigs or the full name of the SFF files. [1]Files listed under the heavy line are produced only when the Paired End option is used.

5.4.2.1 454AllContigs.fna

This file contains the nucleotide sequences of all the contigs with a default minimum length of 100 nucleotides, produced by the GS *De Novo* Assembler application. You can change the minimum length by using the [-a #] option. For a general description of the ".fna" (FASTA) file format, see section 13.3.6. For a sample of a Consensus Basecalled Contig file such as the 454AllContigs.fna file, see section 13.4.5.

5.4.2.2 454AllContigs.qual

This file contains the nucleotide Quality Scores (Phred-equivalent) for all the contigs with a default minimum length of 100 nucleotides, produced by the GS *De Novo* Assembler application. You can change the minimum length by using the [-a #] option. For a description of the process used to compute individual base Quality Scores, see section 3.2.2.2. It should be remembe red that Quality Scores indicate the probability that an individual called base is correct in the sequence, but provide no information on the possibility that a given homopolymer stretch might be longer than called. For a general description of the ".qual" file format, see section 13.3.6. For a sample of a quality score file for Consensus Basecalled Contigs, such as the 454AllContigs.qual file, see section 13.4.5.

5.4.2.3 454LargeContigs.fna

This file contains the same type of information as the 454AllContigs.fna file, but restricted to contigs longer than 500 called bases. The default value, set at 500 bases, can be modified by using the [-1 #] option.

5.4.2.4 454LargeContigs.qual

This file contains the same type of information as the 454AllContigs.qual file, but restricted to the long contigs listed in the 454LargeContigs.fna file (longer than 500 bases, by default. Use the [-1 #] option to modify the length).

5.4.2.5 454NewblerMetrics.txt

The 454NewblerMetrics.txt file is a 454 parser file that reports the key input, algorithmic and output metrics for the GS *De Novo* Assembler application. Below is a description of all the sections present in this file, with their named groups and keywords. Note that the GS *De Novo* Assembler application uses the same basic algorithms as the GS Reference Mapper application, and both generate a 454NewblerMetrics.txt file; however, each application tailors its output to its specific purpose, so the content of the file is different. For a full description of the 454NewblerMetrics.txt file output by the GS Reference Mapper application, see section 6.4.2.7. For a general description of the parser file format, see section 13.3.2. For a sample 454NewblerMetrics.txt file for the GS *De Novo* Assembler application, see section 13.4.6.

1. runData group – contains information about the inputs
   a. run group – information about each Run included in the assembly computation (for data directories only; not for explicit SFF files)
      I. path – the path to the data directory for this Run
      II. region group – information about each of the regions of this Run that are included in the assembly computation
         01. name – the name of the region
         02. sffFile – the name of the SFF file in the sff sub-directory of the GS *De Novo* Assembler application's output directory
         03. numberOfReads – the number of reads that occur in the SFF file (*i.e.* that passed filtering), and then the number of reads used after primer trimming, quality trimming and vector screening
         04. numberOfBases – the total number of bases in the trimmed reads in the SFF file, and then the number of bases after primer trimming, quality trimming and vector screening
   b. file group – information about each SFF or FASTA file included in the assembly computation (for each explicit SFF or FASTA file on the command line; not for data directories)
      I. sffFile – the name of the SFF file in the sff sub-directory of the GS *De Novo* Assembler application's output directory (SFF files only)
      II. path – the original path to the file given on the command line
      III. numberOfReads – the number of reads that occur in the file (*i.e.* that passed filtering, for SFF files), and then the number of reads used after primer trimming, quality trimming and vector screening
      IV. numberOfBases – the total number of bases in the trimmed reads in the file, and then the number of bases after primer trimming, quality trimming and vector screening
      V. numWithPairedReads – the number of reads in the file which were annotated as Paired-End reads (for FASTA files only)

▶▶▶

2 pairedReadData group – contains information about the Paired End input data [Paired End only; 454 Sequencing reads only (not Sanger reads)]

a. run group – information about each Run included in the assembly computation (for data directories only; not for explicit SFF files)
   - I. path – the path to the data directory for this Run
   - II. region group – information about each region of this Run that are included in the assembly computation
     - *01.* name – the name of the region
     - *02.* sffFile – the name of the SFF file in the sff sub-directory of the GS *De Novo* Assembler application's output directory
     - *03.* numberOfReads – the number of reads that occur in the SFF file (*i.e.* that passed filtering), and then the number of individual sequence reads used after dividing each SFF file read into its left and right half sequence reads, and after primer trimming, quality trimming and vector screening
     - *04.* numberOfBases – the number of bases in the trimmed reads in the SFF file, and then the number of bases used after dividing into left and right half sequence reads, primer trimming, quality trimming and vector screening
     - *05.* numberWithPairedRead – the number of reads found to have a valid paired read sequence

b. file group – information about each SFF file included in the assembly computation (for each explicit SFF file on the command line; not for data directories)
   - I. sffFile – the name of the SFF file in the sff sub-directory of the GS *De Novo* Assembler application's output directory
   - II. path – the original path to the SFF file given on the command line
   - III. numberOfReads – the number of reads that occur in the SFF file (*i.e.* that passed filtering), and then the number of individual sequence reads used after dividing each SFF file read into its left and right half sequence reads, and after primer trimming, quality trimming and vector screening
   - IV. numberOfBases – the number of bases in the trimmed reads in the SFF file, and then the number of bases used after dividing into left and right half sequence reads, primer trimming, quality trimming and vector screening
   - V. numberWithPairedRead – the number of reads found to have a valid paired read sequence

3 runMetrics group – contains information about the assembly computation

a. totalNumberOfReads – the number of reads used in the assembly computation (equal to the sum of the numberOfReads for all the Shotgun Runs/regions or SFF/FASTA files)

b. totalNumberOfBases – the number of read's bases used in the assembly computation (equal to the sum of the numberOfBases for all the Runs/regions or SFF/FASTA files in runData group).

c. numberSearches – the number of overlap searches performed during the assembly computation

d. seedHitsFound – internal 454 metric giving the total number and average per search of the initial seed hits found during the exact k-flow matching phase of the overlapper (similar to the initial exact k-mer matching used by blastn and other assemblers)

e. overlapsFound – internal 454 metric giving the total number, average per search and percentage of seedHitsFound of the overlaps that passed the initial matching criteria

f. overlapsReported – internal 454 metric giving the total number, the average per search and the percentage of overlapsFound that were used after a filtering step that looks at multiple overlaps found between specific pairs of reads

g. overlapsUsed – internal 454 metric giving the total number, the average per search and the percentage of overlapsReported that were used after a filtering step performed during overlap detection for a read that looks at all the overlaps found for that read (to all other reads)

►►►

4 readAlignmentResults group – contains information about the alignments per each input file (SFF, FASTA, or Run regions from wells file)

a. run group – information about each Run included in the assembly computation (for data directories only; not for explicit SFF files)
   - I. path – the path to the data directory for this Run
   - II. region group – information about each region of this Run that are included in the assembly computation
     - *01.* name – the name of the region
     - *02.* sffFile – the name of the SFF file in the sff sub-directory of the GS *De Novo* Assembler application's output directory
     - *03.* numAlignedReads – the number of reads from this input file that aligned with other reads in the input data (percentage represents numAligne-dReads over numberOfReads from the runData group)
     - *04.* numAlignedBases – the number of bases from this input file that aligned with other reads in the input data (percentage represents numAligned-Bases over numberOfBases from the runData group)
     - *05.* inferredReadError – percentage of total number of differences such as "indels" found in aligned reads over numAlignedBases (the second number represents the number of differences). (Important Note: This number is calculated by comparing the read alignments to a preliminary consensus of the contigs, so the inferred read error reported for an assembly tends to be ~0.25% higher than the inferred read error found when mapping the reads to the finished genome, using the GS Reference Mapper application (see section 6.4.2.7). The main use of this statistic is to evaluate the Run for poor sequencing data, as compared to the other input files in the input data.

b. file group – information about each SFF or FASTA file included in the assembly computation (for each explicit SFF or FASTA file on the command line; not for data directories)
   - I. sffFile – the name of the SFF file in the sff sub-directory of the GS *De Novo* Assembler application's output directory (SFF files only)
   - II. path – the original path to the file given on the command line
   - III. numAlignedReads – the number of reads from this input file that aligned with other reads in the input data (percentage represents numAlignedReads over numberOfReads from the runData group)
   - IV. numAlignedBases – the number of bases from this input file that aligned with other reads in the input data (percentage represents numAlignedBases over numberOfBases from the runData group)
   - V. inferredReadError – percentage of total number of differences such as "indels" found in aligned reads over numAlignedBases (the second number represents the number of differences). (Important Note: This number is calculated by comparing the read alignments to a preliminary consensus of the contigs, so the inferred read error reported for an assembly tends to be ~0.25% higher than the inferred read error found when mapping the reads to the finished genome, using the GS Reference Mapper application (see section 6.4.2.7). The main use of this statistic is to identify poor sequencing data when compared to other Runs or files, rather than as an absolute measure of sequencing error (which, for accurate results, requires a finished reference sequence and the use of the mapping software).

▶▶▶

5 pairedReadResults group – contains information about the Paired End input data [Paired End only; 454 Sequencing reads only (not Sanger reads)]

a. run group – information about each Run included in the assembly computation (for data directories only; not for explicit SFF files)
   - I. path – the path to the data directory for this Run
   - II. region group – information about each region of this Run that are included in the assembly computation
     - *01.* name – the name of the region
     - *02.* sffFile – the name of the SFF file in the sff sub-directory of the GS *De Novo* Assembler application's output directory
     - *03.* numAlignedReads – the number of reads from this input file that aligned with other reads in the input data (the percentage represents numAlignedReads over numberOfReads from the runData group)
     - *04.* numAlignedBases – the number of bases from this input file that aligned with other reads in the input data (the percentage represents numAlignedBases over numberOfBases from the runData group)
     - *05.* inferredReadError – percentage of total number of differences such as "indels" found in aligned reads over numAlignedBases (the second number represents the number of differences). (Important Note: This number is calculated by comparing the read alignments to a preliminary consensus of the contigs, so the inferred read error reported for an assembly tends to be ~0.25% higher than the inferred read error found when mapping the reads to the finished genome, using the GS Reference Mapper application (see section 6.4.2.7). The main use of this statistic is to evaluate the Run for poor sequencing data, as compared to the other input files in the input data.
     - *06.* numberWithBothMapped – the number of reads where both halves align uniquely into the contigs; the subset of these where both halves aligned to *different* contigs is used to generate scaffolds (within the distance constraint)
     - *07.* numberWithOneUnmapped – the number of reads where one half failed to align to any of the contigs
     - *08.* numberMultiplyMapped – the number of reads where one or both halves aligned equally well to multiple places in the set of contigs
     - *09.* numberWithBothUnmapped – the number of reads where neither half aligned to any of the contigs

▶▶▶

5

b. file group – information about each SFF or FASTA file included in the assembly computation (for each explicit SFF or FASTA file on the command line; not for data directories)
   - I. path – the original path to the file given on the command line
   - II. numAlignedReads – the number of reads from this input file that aligned with other reads in the input data (the percentage represents numAlignedReads over numberOfReads from the runData group)
   - III. numAlignedBases – the number of bases from this input file that aligned with other reads in the input data (the percentage represents numAlignedBases over numberOfBases from the runData group)
   - IV. inferredReadError – percentage of total number of differences such as "indels" found in aligned reads over numAlignedBases (the second number represents the number of differences). (Important Note: This number is calculated by comparing the read alignments to a preliminary consensus of the contigs, so the inferred read error reported for an assembly tends to be ~0.25% higher than the inferred read error found when mapping the reads to the finished genome, using the GS Reference Mapper application (see section 6.4.2.7). The main use of this statistic is to identify poor sequencing data when compared to other Runs or files, rather than as an absolute measure of sequencing error (which, for accurate results, requires a finished reference sequence and the use of the mapping software).
   - V. numberWithBothMapped – the number of reads where both halves align uniquely into the contigs; the subset of these where both halves aligned to *different* contigs is used to generate scaffolds (within the distance constraint)
   - VI. numberWithOneUnmapped – the number of reads where one half failed to align to any of the contigs
   - VII. numberMultiplyMapped – the number of reads where one or both halves aligned equally well to multiple places in the set of contigs
   - VIII. numberWithBothUnmapped – the number of reads where neither half aligned to any of the contigs

6

consensusDistribution group – contains information about the consensus signals and basecalling thresholds

a. fullDistribution group – a histogram of the signal intensities of the consensus signal averages

b. distributionPeaks group – the identified peaks used as the mean signal for a n-mer homopolymer stretch

c. thresholdsUsed group – the thresholds used to perform basecalling of the signals, along with an interpolation amount used when the signal is above the detectable peaks

▶▶▶

7 results group – contains information about the results

a. readStatus group – contains information about the fates of the reads
   I. numAlignedReads – the number of reads that aligned with other reads in the input data (the percentage represents the sum of numAlignedReads over the sum of numberOfReads across the input files)
   II. numAlignedBases – the number of bases that aligned with other reads in the input data (the percentage represents the sum of numAlignedBases over the sum of numberOfBases across the input files)
   III. inferredReadError – percentage of total number of differences such as “indels” found in aligned reads over numAlignedBases (the second number represents the number of differences). (Important Note: This number is calculated by comparing the read alignments to a preliminary consensus of the contigs, so the inferred read error reported for an assembly tends to be ~0.25% higher than the inferred read error found when mapping the reads to the finished genome, using the GS Reference Mapper application (see section 6.4.2.7). The main use of this statistic is to identify poor sequencing data when compared to other Runs or files, rather than as an absolute measure of sequencing error (which, for accurate results, requires a finished reference sequence and the use of the mapping software).
   IV. numberAssembled – the number of reads fully assembled into the contigs
   V. numberPartial – the number of reads partially assembled into the contigs
   VI. numberSingleton – the number of reads that did not overlap with other reads
   VII. numberRepeat – the number of reads deemed to be from repeat regions
   VIII. numberOutlier – the number of outlier sequences identified by the assembler
   IX. numberTooShort – the number of reads whose trimmed sequence was too short to be used

b. pairedReadStatus – contains information about the fates of the paired reads [Paired End only (454 Sequencing reads or Sanger reads)]
   I. numberWithBothMapped – the number of reads where both halves align uniquely into the contigs; the subset of these where both halves aligned to *different* contigs is used to generate scaffolds (within the distance constraint)
   II. numberWithOneUnmapped – the number of reads where one half failed to align to any of the contigs
   III. numberMultiplyMapped – the number of reads where one or both halves aligned equally well to multiple places in the set of contigs
   IV. numberWithBothUnmapped – the number of reads where neither half aligned to any of the contigs
   V. library – information for each Paired End library in the input data. Sanger reads explicitly identify the library they came from; each 454 Paired End region or SFF file is treated by default as a separate library
      01. libraryName – the name of the Paired End library
      02. pairDistanceAvg – the average distance between both halves, when both halves align on the same contig
      03. pairDistanceDev – the standard deviation of the pairDistanceAvg distribution

c. scaffoldMetrics – contains information about the scaffolds [Paired End only (454 Sequencing reads or Sanger reads)]
   I. numberOfScaffolds – the number of scaffolds identified
   II. numberOfBases – the total number of bases in the scaffolds
   III. avgScaffoldSize – the average scaffold size
   IV. N50ScaffoldSize – the N50 scaffold size
   V. largestScaffoldSize – the size of the largest scaffold

▶▶▶

7 d. largeContigMetrics group – contains information about the large contigs
- I. numberOfContigs – the number of large contigs identified
- II. numberOfBases – the total number of bases in the large contigs
- III. avgContigSize – the average contig size
- IV. N50ContigSize – the N50 contig size
- V. largestContigSize – the size of the largest contig
- VI. Q40PlusBases – the number and percentage of bases called that have a quality score of 40 or above
- VII. Q39MinusBases – the number and percentage of bases called that have a quality score of 39 or below

e. allContigMetrics group – contains information about all the contigs
- I. numberOfContigs – the number of contigs identified
- II. numberOfBases – the total number of bases in the contigs

5.4.2.6 454NewblerProgress.txt

This file represents the text log of the messages sent to standard output by the runProject command (showing the progress of the execution of the assembly computation). If Runs are added incrementally and multiple executions of runProject occur, the output messages are appended to this file. If the "Incremental *de novo* assembly analysis" checkbox option is not selected in the GUI application, or the "-r" option is given on the runProject command line, the GS *De Novo* Assembler deletes intermediate data and "restarts" the assembly computation, and this file is deleted and restarted as well. For a sample 454NewblerProgress.txt file, see section 13.4.10.

5.4.2.7 454AlignmentInfo.tsv

The 454AlignmentInfo.tsv file contains position-by-position summary information about the consensus sequence for the contigs generated by the GS *De Novo* Assembler application, listed one nucleotide per line (in a tab-delimited format). For a sample 454AlignmentInfo.tsv file, see section 13.4.11.

The columns of each line contain the following information:

1	Position – the position in the contig
2	Consensus – the consensus nucleotide for that position in the contig
3	Quality Score – the quality score of the consensus base
4	Depth – the number of reads that align at that position in the alignment
5	Signal – the average signal of the read flowgrams, for the flows that correspond to that position in the alignment
6	StdDevation – the standard deviation of the read flowgram signals at the corresponding flows

Prior to each region of lines for each contig, a header line beginning with a '>' displays the contig name.

5.4.2.8 454ContigGraph.txt

The 454ContigGraph.txt file is output only when specifically called for, and is available only when the application is run from the command line: this is done by using the –g option; or by setting the minimum length for a contig to appear in the 454AllContigs.fna file to zero, using the –a option (see sections 5.3.1 and 5.3.2.4). The file contains a graph-based description of the branching structure of the assembly's contigs, where nodes represent the contigs and edges between contigs give the branching structure. When Paired End reads are included in the assembly, the file also shows scaffold edges, describing how the contigs have been linked together into scaffolds.

The file comprises two or three sections (the third section is for scaffold information, with Paired End reads), with each line of a section describing a single node or edge of the graph. The first section describes the nodes (contigs) of the graph, and contain the following columns, separated by tab characters:

1. The number of the contig (*i.e.*, for contig "contig00002", this column would contain the number "2")
2. The full name of the contig (*e.g.*, "contig00002")
3. The length of the consensus for this contig
4. The average depth of the contig's multiple alignment

The second section describes the edges of the graph, describing how the alignment of reads branch off in different directions (or converge together) from the end of a contig. Edges connect not contigs, but the ends of contigs (each contig has a 5' and 3' end, corresponding to the beginning (5') and end (3') of the contig multiple alignment). Different contigs are not oriented relative to each other, so an edge can connect the 5' end of contig00001 and the 5' end of contig00002 (or the 3' ends of two contigs). In this situation, reverse complementing one of the contigs, so that the edge is a 5' to 3' edge, presents the two contigs in a consistent orientation with each other. The columns of the second section are the following (separated by tab characters):

1. 'C' – the letter 'C' to mark this line as a "contig edge"
2. The contig number on one side of the edge
3. Either "5'" or "3'" to denote which end of the contig
4. The contig number on the other side of the edge
5. Either "5'" or "3'" to denote which end of the contig
6. The depth of the multiple alignment that spans between the two contig ends (*i.e.*, the number of reads whose alignments crosses the two contigs ends).

For example, the line

```
C    1    5'    2    3'    21
```

…describes an edge between the 5' end of contig00001 and the 3' end of contig00002, where the alignments of 21 reads occur partially at the 5' end of contig00001 and partially at the 3' end of contig00002. All edges are undirected edges.

The third section of the file describes the "scaffold edges" found when performing scaffolding (with Paired End reads only). Only edges used in the scaffolding itself are listed (not edges representing all paired end data). The columns of the third section are the following (separated by tab characters):

1	'S' – the letter 'S' to mark this line as a "scaffold edge"
2	The contig number on one side of the edge
3	Either "5'" or "3'" to denote which end of the contig
4	The contig number on the other side of the edge
5	Either "5'" or "3'" to denote which end of the contig

For example, the line

```
S       6       3'       8       5'
```

…describes an edge between the 3' end of contig00006 and the 5' end of contig00008. All edges are undirected edges.

5.4.2.9 454PairAlign.txt

This file contains the pairwise alignments of the overlaps that were found during the assembly computation. By default, this file is not generated, but if the "-p" or "-pt" options are given on the runProject command line, this file will be generated either in a human-readable text format ("-p") or in tab-delimited format ("-pt").

Each of the displayed alignments contains the following information (these are the columns in the tab-delimited format):

1	QueryAccno – accession number of the read used in the overlap detection search (the "query sequence")
2	QueryStart – starting position of the alignment in query sequence
3	QueryEnd – ending position of the alignment in query sequence
4	QueryLength – length of the query sequence
5	SubjAccno – accession number of the other read (the "subject sequence")
6	SubjStart – starting position of the alignment in subject sequence
7	SubjEnd – ending position of the alignment in subject sequence
8	SubjLength – length of the subject sequence
9	NumIdent – number of identities in the pairwise alignment, *i.e.* where query and subject characters match
10	AlignLength – the length of the pairwise alignment
11	QueryAlign – query alignment sequence
12	SubjAlign – subject alignment sequence

5.4.2.10 454ReadStatus.txt

The 454ReadStatus.txt file contains the status identifiers for all the reads used in the assembly computation, listed one per line, in tab-delimited format. The status string describes the read's fate in the assembly, and can be one of the following five values:

- Assembled – the read is fully incorporated into the assembly
- PartiallyAssembled – only part of the read was included in the assembly, the rest was deemed to have diverged sufficiently to not be included
- Singleton – the read did not overlap with any other reads in the input
- Repeat – the read was identified by the assembler as likely coming from a repeat region, and so was excluded from the final contigs
- Outlier – the read was identified by the GS *De Novo* Assembler as problematic, and was excluded from the final contigs (one explanation of these outliers are chimeric sequences, but sequences may be identified as outliers simply as an assembler artifact)
- TooShort – the trimmed read was too short to be used in the computation (*i.e.*, shorter than 50 bases, unless 454 Paired End Reads are included in the dataset, in which case, shorter than 15 bases).

For a sample 454ReadStatus.txt file resulting from an assembly computation, see section 13.4.7. Note that as the possible fates of reads in an assembly computation are different from possible fates in a mapping computation, the file of the same name, 454ReadStatus.txt, produced by the GS Reference Mapper (see sections 6.4.2.12 and 13.4.12) is completely different from that produced by the GS *De Novo* Assembler.

5.4.2.11 454TrimStatus.txt

This file contains a per-read report of the original and revised trimpoints used in the assembly, where the revised trimpoints include the effects of primer, vector and/or quality trimming of the original input reads. Each line contains the following information (these are the columns in the tab-delimited format):

1. Accno – accession number of the input read
2. Trimpoints Used – the final trimpoints used in the assembly, in #-# format
3. Trimmed Length – the final trimmed length of the read
4. Orig. Trimpoints – the original trimpoints of the read, found in the SFF or FASTA file
5. Orig. Trimmed Length – the original trimmed length of the read
6. Raw Length – the length of the raw read (without any trimming)

5.4.2.12 454Contigs.ace or ace/ContigName.ace or consed/…

This viewer-ready genome file shows all the contigs contained in 454AllContigs. fna and allows the display of how the individual reads aligned to those contigs, in an ACE format file suitable for use in various third-party sequence finishing programs. (The freeware "clview" application can be downloaded from: http://compbio.dfci.harvard.edu/tgi/software/; a full description of the .ace file format can be found at: http://bozeman.mbt.washington.edu/consed/consed.html.) It should be noted, however, that such third-party viewing software will not be able to make full use of the flowspace assembly information available with Genome Sequencer reads, and that conversely some of the third-party program's functions (*e.g.* involving sequence chromatogram input) are not usable with Genome Sequencer datasets. Nonetheless, these programs may be useful to view and assess read characteristics and coverage depth in regions of interest.

The number of ACE files generated during as assembly computation is determined by which ACE option is checked, as described in sections 5.2.4 (for project-based assemblies using the GUI) and 5.3.2.4 (for assemblies carried out from the command line).

The GS *De Novo* Assembler application can also output the assembly results in a complete directory structure suitable for use as input to the consed software (see the above link for a description of this structure and its files). When the appropriate option is selected, a "consed" sub-directory is produced in the output (or project) directory for the assembly. This directory contains the sub-directories, the ACE file and the PHD file, so that the functions of consed (viewing traces, editing reads, autofinishing) can be performed on the assembly. In order to integrate the 454 Sequencing reads (and their SFF files) into the consed data, an extra "sff_dir" directory is created in the consed sub-directory, and a number of consed options are automatically specified in this directory (see the "consed/ edit_dir/.consedrc" file for the options specified by the generated structure). One option tells consed to use the "sff2scf" command to access any trace information requested by a user. See Section 7.3 for a description of the sff2scf command, and how it can generate synthetic traces from SFF data, as well as provide a pass-through access to SCF data for Sanger reads.

The read information included in the ACE file produced by the GS *De Novo* Assembler application differs from traditional assemblies in that a single read may appear in multiple contigs of the assembly. This occurs because the objective of the Genome Sequencer FLX assembler software is to first identify and partition the repeat and non-repeat regions of the genome, and then output the consensus sequences for those regions. Therefore, if a single read spans the boundary between two contigs only one of which consists of a long repeat region, it will be displayed on both the repeat contig and the specific contig on the other side of the boundary.

To ensure compatibility with ACE file viewers, as well as to assist in the post-assembly analysis of the assembly, the identifiers for the reads that appear in multiple places, and for paired-end reads, are output with an additional suffix. Several suffixes may be added to the original read identifier:

- For 454 Paired End reads, the two halves of the 454 read that constitute the sequences at the two ends of the original clone (from which the paired-end read was generated) are marked by "_left" and "_right" suffixes.
- If only part of the trimmed read aligns in this contig (either because the read is only Partially Assembled, or the read is aligned with different sections in different contigs, because it spans a repeat region boundary, for example), the base position range of the region aligned in this contig is added, as in ".1-60" if the read has bases 1-60 aligned in the contig.
- If a read has sections aligning to different contigs, then a ".fm" or ".to" suffix (or both) is added to list the contig the read's alignment is "coming from" or "going to". The ".fm" suffix is added when the read's alignment continues off the 5' end of the contig, and the ".to" suffix is added for reads that contig off the 3' end of the contig. For example, if read "C3U5GNB01C40I3" appears in contigs 2 and 45 (with bases 1-60 appearing in contig 2 and bases 61-248 appearing in contig 45), the entry in contig 2 will use the identifier "C3U5GNB01C40I3.1-60.to45" and the entry in contig 45 will be "C3U5GNB01C40I3.61-248.fm2".
- Paired End reads where both halves of the pair align into contigs will have a ".pr" suffix to list where the other half aligns in the assembly. For example, if reads "C3U5GNB01R8GSX_left" and "C3U5GNB01R8GSX_right" align in contigs 36 and 87, respectively, then the accessions will appear as "C3U5GNB01R8GSX_left.pr87" and "C3U5GNB01R8GSX_right.pr36" (assuming that their complete trimmed sequences aligned inside the two contigs, and did not span across multiple contigs).

5.4.2.13 sff/sfffilename

The files in the "sff" sub-directory of the GS *De Novo* Assembler application's output directory are the files used as input for the assembly computation. They are constructed as the application processes the command line arguments (or their equivalent GUI parameters). The conversion of each command line argument into SFF files in this directory occurs in one of two ways:

- if the command line argument gives the path to an SFF file, that is the file to use
- if the command line argument specifies a data directory in which the SFF files exist, then the files in the "sff" sub-directory of the data directory (possibly restricted to the regions given in the optional region list) are the files to use

This "sff" sub-directory in the assembly output (inside the "P_" directory; see Figure 1–4) provides a consistent place for users to access the input used by the GS *De Novo* Assembler application, independent of which method is used to specify the files.

The filenames of the SFF files in this sub-directory are assigned by the GS *De Novo* Assembler application to ensure uniqueness for all the files, while trying to preserve the original names when possible.

- For SFF files explicitly specified on the command line, the software uses the original filename of the file except when there are duplicate filenames, in which case a numerical suffix is added to names found to be duplicates (*e.g.* if the command line is "runAssembly dir1/454Reads.sff dir2/454Reads.sff dir3/454Reads.sff", the assembler will name the first "454Reads.sff", the second "454Reads.sff.1" and the third "454Reads.sff.2").
- For data directory arguments on the command line, the software relies on the universal accession naming convention to ensure uniqueness of the filenames. In fact, it reports an error if duplicate filenames are found, because the most common cause of that is when two different Signal Processing analyses from the same sequencing Run are given as input to the GS *De Novo* Assembler application, which assumes that each read is an independent sequence.

5.4.2.14 454Scaffolds.fna

This file contains the same type of information as the 454AllContigs.fna file (see section 5.4.2.1), except that the sequences consist of sets of concatenated contigs, separated by regions of "N" characters. For a general description of the ".fna" (FASTA) file format, see section 13.3.6. For a sample of a Consensus Basecalled Contig file such as the 454AllContigs.fna file, see section 13.4.5.

5.4.2.15 454Scaffolds.qual

This file contains the same type of information as the 454AllContigs.qual file (see section 5.4.2.2), but with quality scores corresponding to the bases in the 454Scaffolds.fna file (where "N" characters are given a quality score of 0). For a general description of the ".qual" file format, see section 13.3.6. For a sample of a quality score file for Consensus Basecalled Contigs, such as the 454AllContigs.qual file, see section 13.4.5.

5.4.2.16 454Scaffolds.txt

This file contains AGP-formatted information describing how the contigs included in the 454LargeContigs.fna file (see section 5.4.2.3) are scaffolded into the sequence scaffolds of 454Scaffolds.fna (section 5.4.2.14). A description of the AGP format can be found on NCBI's web site (http://www.ncbi.nlm.nih.gov/genome/guide/Assembly/AGP_Specification.html).

5.4.2.17 454PairStatus.txt

This file contains the per-pair report of the location and status of how each Paired End pair of reads were used in the assembly. Each line contains the following information (these are the columns in the tab-delimited format):

1. Template – template string for the pair (this will be the original 454 accession for 454 Paired End reads, and the "template" string for Sanger reads)
2. Status – the status of the pair in the assembly, with the following possible values
   a. BothUnmapped – both halves of the pair were unmapped
   b. OneUnmapped – one of the reads in the pair were unmapped
   c. MultiplyMapped – one or both of the reads in the pair were marked as Repeat
   d. SameContig – both halves of the pair were assembled into the same contig, and are within the expected distance of each other
   e. Link – the halves were assembled to different contigs, and are near enough to the ends of those contigs that they could be used as a link in a scaffold
   f. FalsePair – the halves were assembled, but the directions of the alignment is inconsistent with a Paired End pair or the distance between the halves is outside the expected distance
3. Distance – for "SameContig" or "Link" pairs, the distance between the halves (where the "Link" distance is simply the sum of the distances from each half to the end of its respective contig).
4. Left Contig – the contig where the left half was assembled, or "-" if the read was Unmapped or Repeat. (If the read aligns across multiple contigs, the location of the first base in the read that is aligned is used, to denote where the end of the corresponding clone occurs.)
5. Left Pos – the position in the contig where the left half was assembled
6. Left Dir – the direction ('+' for the forward strand of the contig and '-' for reverse strand) in which the left half was assembled
7. Right Contig – the contig where the right half was assembled, or "-" if the read was Unmapped or Repeat. (If the read aligns across multiple contigs, the location of the first base of the read that is aligned is used, to denote where the end of the corresponding clone occurs.)
8. Right Pos – the position in the contig where the right half was assembled
9. Right Dir – the direction ('+' for the forward strand of the contig and '-' for reverse strand) in which the right half was assembled
10. Left Distance – the distance from the Left Pos to the respective end of the contig (for forward matches, this is the distance to the 3' end of the contig, for reverse matches, to the 5' end)
11. Right Distance – the distance from the Right Pos to the respective end of the contig (for forward matches, this is the distance to the 3' end of the contig, for reverse matches, to the 5' end).

5.4.2.18 454TagPairAlign.txt

This file contains the pairwise alignments of "short" reads that were found during the assembly computation. When 454 Paired End reads are included in the dataset, reads that are between 15 and 50 bases long are not included in the overlap computation, but are aligned to the contig consensi in a later step of the computation (using different alignment detection settings). This file reports the alignments of the 15–50 bp reads against the contig consensi. By default, this file is not generated, but if the "-p" or "-pt" options are given on the runProject command line, this file will be generated either in a human-readable text format ("-p") or in tab-delimited format ("-pt"). The format of this file is identical to the 454PairAlign.txt file described in section 5.4.2.9.

5.5 Assembling with Reads Obtained Using the Sanger Sequencing Method

Introducing reads obtained using the Sanger sequencing method ("Sanger reads") into an assembly can be more complicated than including 454 Sequencing reads, because paired end information, trimming information and vector (and other non-insert) information is often not directly associated with the read sequences. To utilize the full capabilities of the GS *De Novo* Assembler, the FASTA files of input Sanger reads must be prepared so that they provide the software with the paired end, trimming and non-insert information. This section describes how the GS *De Novo* Assembler accesses that information from the FASTA files it gets as input; section 7.4 describes the "fnafile" command, which can assist in preparing the files.

External files: FASTA files are not actually copied to the project directory. Rather, the file path is simply recorded in the project setup. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies to all instances of FASTA files that can be added to an Assembly project, whether done via the GUI or the command line:

- FASTA reads files, including Sanger reads
- Trimming database files
- Screening database files

Unlike with the GS Read Data files, no symbolic link to the file is created for FASTA files.

5.5.1 Direct Incorporation into the Assembly Process

In the simplest case, the GS *De Novo* Assembler can take FASTA files and assemble the sequences they contain directly, without any preparation. In this case, it treats all the reads present in the FASTA files as single ended, shotgun reads, and assumes that the sequence has been trimmed for quality and for vector/primer/adapter/linker/… sequences (all non-insert sequence). The assembler will use all the sequence data present in the file, and produce contigs based on those sequences.

5.5.2 Sanger Reads with available Quality Scores

If there are quality score files associated with the FASTA files, the GS *De Novo* Assembler will automatically read the quality scores for the reads, and use them in the assembly and consensus calling. The assembler looks for a file that begins with the same file name or file prefix as the FASTA file but ending with ".qual". So, for a FASTA file whose name is "myreads.fna", the assembler will test both "myreads.fna.qual" and "myreads.qual" (replacing any suffix with ".qual") in its search for a quality score file. If such a file exists, it must contain sequence entries that match (in order) the sequences of the FASTA file, and the number of quality scores for each sequence must match the number of bases of each corresponding sequence in the FASTA file.

5.5.3 Sanger Read Annotations

Additional information about the Sanger reads can be specified using "*name=value*" annotation strings on the description line of each sequence in the FASTA file. The GS *De Novo* Assembler looks for and uses are the following annotation strings:

1. **template** – the paired end template string for this read
(paired end reads are matched by having the same template string)
2. **dir** – values "F", "R", "fwd" or "rev" giving the direction of the paired end read
3. **library** – the name of the library that generated this paired end read (all paired end reads are grouped by library name for the determination of expected pair distance)
4. **trim** – the trimmed region of the sequence, given as "#-#" (see below for more details)
5. **scf** – the path or "command string" to use to access the SCF file for the read
(see below)
6. **phd** – the path or "command string" to use to access the PHD file for the read

For example, the description line:

```
>DJS045A03F template=DJS054A03 dir=F library=DJS045
trim=12-543
```

…tells the GS *De Novo* Assembler that this read (accession number "DJS045A03F") is a paired end read whose template is "DJS045A03", is a forward read from library "DJS045", and bases 12-543 should be used in the assembly.

The GS *De Novo* Assembler looks for the six "*name=*" strings on the description line, and then takes the text from the "=" to the next whitespace character as the "*value*" of the annotation string. So, other text besides annotation strings can appear on the description line, and no whitespace may appear in the value of an annotation string.

5.5.3.1 The "template", "dir", and " library" Annotations

The first three annotation strings, template, dir and library, are used to specify the paired end information for a read; only reads with template, dir and library annotations will be treated as paired end reads. The GS *De Novo* Assembler performs exact string matching of the template and library annotations to determine which reads come from the same template or library. Multiple reads may have the same template, dir and library information: they all will be assembled but only the best aligned sequence (by aligned length or number of differences from the consensus) will be used for scaffolding. There is no requirement that the accession number for the read encode the paired end information, and the assembler will not try to "parse" the accession number for any information. However, if you use a naming convention for encoding the paired end information in the accession numbers for reads, the fnafile command can be very useful for translating that encoded information into the annotation strings that the GS *De Novo* Assembler can then read.

5.5.3.2 The "trim" Annotation

The trim annotation describes the region of the sequence that the GS *De Novo* Assembler should use for assembly (with the assumption that the rest of the sequence is non-insert or low quality). For a read 800 bases in length and a trim annotation string "trim=12-543", for example, the assembler will ignore bases 1-11 and 544-800, and only use bases 12-543 in the assembly. If either number in the trim annotation string is 0, the beginning or end of the read will be treated as the trimming point (*i.e.*, for the same example read as above, "trim=12-0" would tell the assembler to use bases 12-800 of the read in the assembly).

This trim annotation should combine the results of all of the sources of trimming that may occur (low quality, vector, primer, adapter, linker, *etc.*), so that the GS *De Novo* Assembler is given just the sequence region that represents the bases to be included in the assembly. Although the GS *De Novo* Assembler software does have some mechanisms to handle such sequences, their widespread appearance in the reads of an assembly project will slow down the assembler processes or may cause contig breaks. The "–v" option can be used with the runAssembly and runProject commands to specify a FASTA file containing sequences to be trimmed: each read will then be screened against this database, and the ends of reads that match the sequences included will be trimmed off (and, if the whole sequence matches, that read will not be used in the assembly).

5.5.3.3 The "scf" and "phd" Annotations

The scf and phd annotations provide a link back to the "trace" information for the read, and are only used when generating the full consed directory structure (when the –consed option is used). The value string for scf or phd annotations can take one of two forms, either as a simple path to the SCF or PHD file, or as a "command string" to be executed to access the SCF or PHD file contents. If the value string does not begin with "cmd:", it is treated as a path. If the value string begins with "cmd:", it should have the format "cmd:*commandline*", where any whitespace in the *commandline* portion of the command should be replaced with colons.

For example, if the command to access a PHD file when run from the command line is "/usr/local/bin/getphd –v *accno*" (to get the PHD file for "*accno*"), then the phd annotation string should be given as "phd=cmd:/usr/local/bin/getphd:-v" (the accno will be appended as the final argument when the command is executed). Be aware that this command will likely be executed from a different location from where it is added to the FASTA file, so the string should either contain an absolute path to the executable, or the executable should be placed in a directory in the user's PATH environment variable's list of directories.

Any command specified in the scf or phd annotation strings must, when executed, write the PHD or SCF contents to standard output. The command's standard output is read, and those bytes are written or processed as needed, by the assembler and/or the sff2scf command (the two programs in the current software that require access to the PHD or SCF contents).

5.5.4 Recommended Method for including Sanger Reads

The best method for including Sanger reads into an assembly is to first run phred with vector screening (or an equivalent basecaller that generates a FASTA file with vector sequence marked as 'X' or 'N', plus a quality score file). The ".fasta.screen" and "fasta. screen.qual" files produced by phred contain the screening and trimming information needed by the GS *De Novo* Assembler (and it is setup to properly process those reads and the marking and quality information).

There are two ways to include the pairing information for the reads, each of which requires some Perl scripting, because the paired-end information is typically encoded in the accession numbers of the reads. The simplest way is to write a Perl script like the the one below. This script is designed to:

- handle a paired-read naming convention where there is only one paired-end library;
- where the forward and reverse reads have suffixes ".f1" or ".r1"; and
- where the paired-end reads can be distinguished from follow-up finishing reads by having only alphanumeric characters before the ".f1" or ".r1"):

```
#!/usr/bin/perl
if ($ARGV[0] =~ /^(\w*)\.([rf])1$/) {
   print “template=$1 dir=$2 library=pairlib\n“;
}
```

…where "$ARGV[0]" will be the accession number for the read, when used in the procedure below.

The key to this script is the regular expression "^(\w*)\.([rf])1$", which matches a word of any length, "\w*", followed by a period, "\.", followed by either an 'r' or an 'f', followed by a '1'. The parentheses are there to then extract the characters that matched the word and matched the 'r' or 'f' and place them into the "$1" and "$2" parameters (so that they can be used in the print statement on the next line). For your naming convention, just customize the perl regular expression to extract out the template, library and forward/ reverse direction strings from the accession, and output them appropriately (see section 5.5.3 for a description of this output).

If we call this script "accnoPair", then the following procedure can be used to get Sanger reads from your runs into the assembler:

1. Run phred with vector screening to produce the "fasta.screen" file (plus fasta.screen.qual).
2. Run the command "fnafile -o sanger.fna -ac accnoPair fasta.screen" to attach the paired-end information to each read, where fnafile calls the accnoPair program for each read, then adds the output of that command to the read's description line.
3. Give sanger.fna to the assembler, where it will read and use the pairing information.

The one drawback to this method is that the execution of the perl script for each read causes the fnafile program to take a couple of minutes to convert 15,000–20,000 reads (the operating system just takes that long to run the perl script 15,000–20,000 times). However, a slightly more complicated script and slightly different procedure can be used to run quickly.

If the following perl script is generated (call this “detPairs”, where one will note that lines 7 and 8 match the two lines of the accnoPair script):

```
#!/usr/bin/perl
die “Usage: detPairs screenfile\n“ unless
defined($ARGV[0]); open(FILE, $ARGV[0]) or die “Error:
Unable to open file: $ARGV[0]\n“;
while (<FILE>) {
    if (/^\>(\S*)/) {
        my $accno = $1;
        if ($accno =~ /^(\w*)\.([rf])1$/) {
            print “$accno template=$1 dir=$2
            library=pairlib\n“;
        }
    }
}
```

…then the following procedure can be used to get the Sanger reads into the assembler:

1. Run phred with vector screening to get the fasta.screen file.
2. Run the command “detPairs fasta.screen > sanger.acc” to get a file with the pairing information.
3. Run the command “fnafile -o sanger.fna -af sanger.acc fasta.screen”.
4. Give the sanger.fna file to the assembler.

For both of these scripts, see section 7.4 for a description of the command line structure and arguments of fnafile.

6. GS Reference Mapper Application

6.1 Introduction to the GS Reference Mapper

The GS Reference Mapper software is an interactive application that allows the user to create mapping projects, add and remove reads from the project, set and replace the reference sequence for the project, run the mapping algorithms on the project data, and view the output produced by the mapping computations. The Mapper operates via a Graphical User Interface (GUI; section 6.2) or from the command line interface (section 6.3).

Mapping projects allow the user to align the reads from one or more sequencing Runs to a reference sequence, reference database of sequences or GoldenPath genome, using as input information in the SFF files from some or all the regions of the Run(s) of interest, and generate a consensus sequence of the sample DNA library (in one or more contigs). As described in sections 3.2.1 and 3.2.2, the read information contained in the SFF files contains the high quality data to be used for mapping. The mapping software also allows the inclusion of one or more FASTA files of reads, such as reads obtained using the Sanger sequencing method ("Sanger reads"), in the analysis; the information in these files can then be incorporated into the mapping process, evaluated for accuracy, and/or evaluated for coverage.

In the mapping process, the software performs the following operations (see also Table 1–2):

- ▶ For each read, search for a suitable alignment, or alignments, of the read to the reference sequence(s) (a read may align to multiple positions in the reference sequence); this is done in "nucleotide" space
- ▶ Construct contigs and compute a consensus basecall sequence from the signals of the aligned reads (performed in "flowspace")
- ▶ Identify the positions in the aligned reads (consensus) that differ from the reference sequence(s); alternatively, identify subsets of the aligned reads that are identical within each subset but differ between subsets (these are the "putative differences")
- ▶ Evaluate the list of putative differences to identify High-Confidence differences
- ▶ Output the following information: contig consensus sequence(s) and associated quality values; alignments of the reads and contigs to the reference, position-by-position metrics of the depth and consensus accuracy (quality values) for each position in the aligned reference; and the positions and alignments of identified differences

While the initial read to reference alignments are made in "nucleotide" space, the consensus basecalling and quality value determination for contigs are performed in "flowspace". Work in flowspace allows the averaging of processed flow signals (a continuous variable) at each nucleotide flow of the sequencing Run(s) rather than called bases (a discontinuous variable) at each nucleotide position along the alignment. It also allows the use of information from the "negative flows", *i.e.* flows where no nucleotide incorporation is detected, which would obviously not be possible in nucleotide space. Alignment in flowspace thus results in an improved accuracy for the final basecalls.

The GS Reference Mapper application is not available on the Genome Sequencer FLX Instrument and must be run on a DataRig.

6.2 Using the GS Reference Mapper Via the Graphical User Interface

Mapping sequencing reads to one or more reference sequences can be performed either using the Graphical User Interface described in this section, or using the Linux command line, as described in section 6.3. The application includes graphic interfaces to:

- create a new mapping project
- open an existing mapping project
- carry out a mapping computation on an open project
- view the results of a completed mapping computation
- view progress and logging information of a completed mapping computation

The GS Reference Mapper GUI application is written in Java, and requires Java version 1.5. Java 1.5 is included as part of the off-instrument software package to be installed on your DataRig(s). In the course of its operation, the GS Reference Mapper GUI application makes use of C and C++ components, all of which can be used to perform command line mapping operations separately from the graphic application, as described in sections 6.3 and 7).

6.2.1 Launching the Graphical User Interface Application

The GS Reference Mapper GUI application is launched via a single command, whose command line structure is the following:

```
gsMapper
```

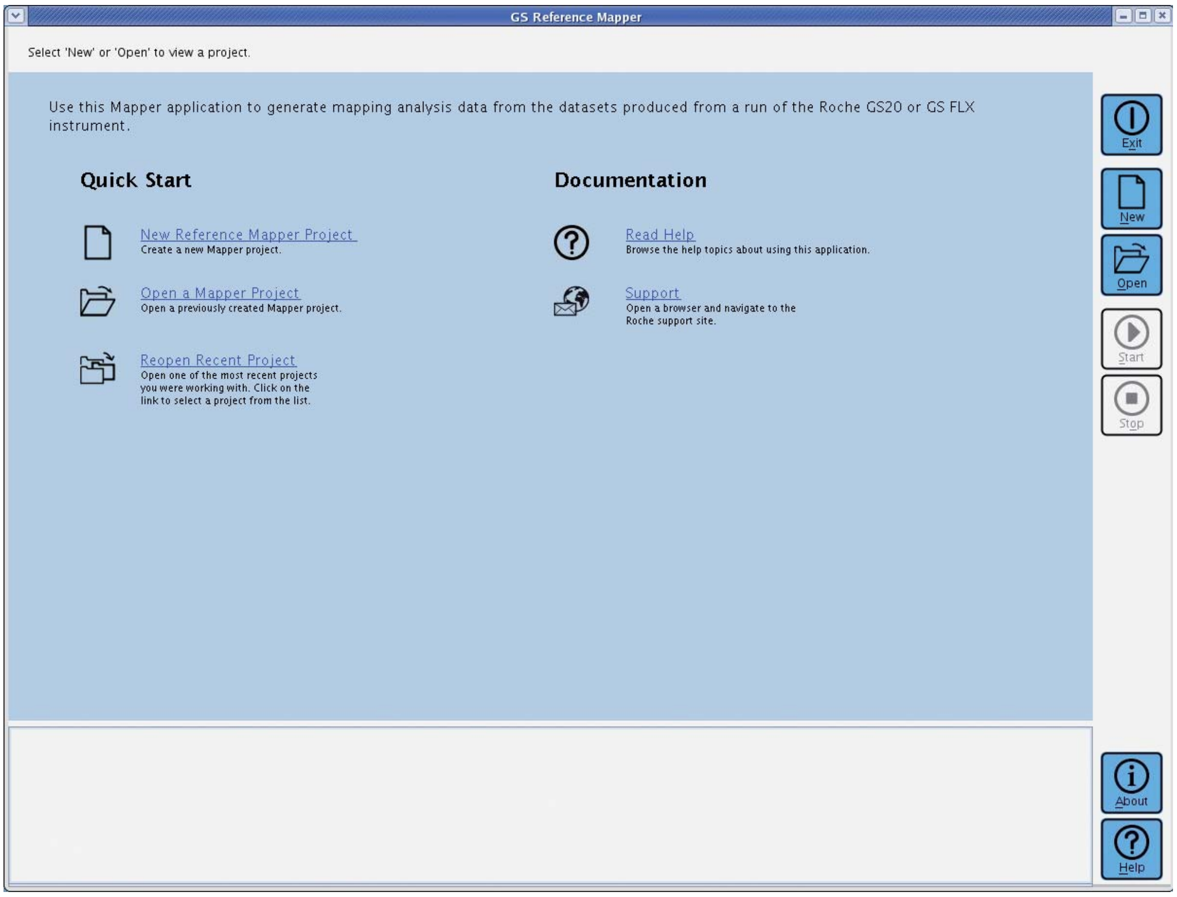
This will open a splash screen (Figure 6–1) and the main application windows (Figure 6–2). The splash screen can be dismissed by clicking anywhere on it, but it will also disappear on its own after approximately 6 seconds.

GS Reference Mapper
Version 2.0.00
Roche
www.roche-applied-science.com
454
SEQUENCING
(c) 454 Life Sciences 2006-2008. All rights reserved.

Figure 6–1: The GS Reference Mapper application splash screen

6.2.2 GS Reference Mapper Interface Overview

The GS Reference Mapper entry window (Figure 6–2) comprises a vertical toolbar of main function buttons; a menu area with three "Text Links" for creating and opening mapping projects, and two "Text Links" for help and support; as well as a status area along its top edge and a progress box at the bottom.



Select 'New' or 'Open' to view a project.

Use this Mapper application to generate mapping analysis data from the datasets produced from a run of the Roche GS20 or GS FLX instrument.

Quick Start

New Reference Mapper Project
Create a new Mapper project.

Open a Mapper Project
Open a previously created Mapper project.

Reopen Recent Project
Open one of the most recent projects you were working with. Click on the link to select a project from the list.

Documentation

Read Help
Browse the help topics about using this application.

Support
Open a browser and navigate to the Roche support site.

Figure 6–2: The GS Reference Mapper application's main window, just after the application is launched

The progress information field at the bottom of the window can be resized by dragging its top frame separator. If the field is not visible, it is probably simply collapsed; you can view it by dragging up the edge of the collapsed frame.

6.2.2.1 The Main Buttons, Status Area, and Progress Box

Seven main buttons are always visible on the toolbar, along the right-hand side of the GS Reference Mapper's main window:

- The **Exit** button closes the GS Reference Mapper application
- The **New** button allows you to create a new mapping project
- The **Open** button allows you to open an existing mapping project
- The **Start** button begins the mapping (computation) of an open mapping project
- The **Stop** button halts the execution of an ongoing mapping computation
- The **About** button shows the GS Reference Mapper splash screen
- The **Help** button shows a help dialog (online help is not currently implemented)

The top of the main GS Reference Mapper window contains a status area that displays a command hint, while the bottom panel is a progress box which remains blank until a mapping project has been opened.

6.2.2.2 The Quick Start and Documentation Text Links

There are 3 text links under the **Quick Start** column:

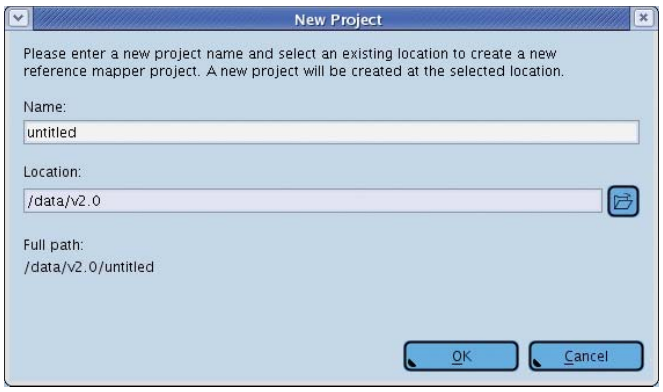
- The **New Reference Mapper Project** text link creates a new mapping project. It performs the same action as the **New** button on the right side of the window.
- The **Open a Mapper Project** text link opens an existing project. It performs the same action as the **Open** button on the right side of the window.
- The **Reopen Recent Project** text link displays a dialog from which you can open a project on which you worked recently.

There are 2 text links under the **Documentation** column:

- The **Read Help** text link shows a help dialog. Online help is not currently implemented; please use this Software Manual, or click on the Support button (see below), or contact your Roche Representative for any information regarding the Genome Sequencer FLX System and its software package. The Read Help text button performs the same action as the **Help** button on the right side of the window.
- The **Support** text link opens the default internet browser and navigates to the Roche support site.

6.2.3 Creating a New Mapper Project

To create a new mapping project, either click on the **New Project** button in the right toolbar, or click on the "New Reference Mapper Project" text button in the **Quick Start** column. This displays a dialog (Figure 6–3) in which you can specify the name and directory location for a new project.



Please enter a new project name and select an existing location to create a new reference mapper project. A new project will be created at the selected location.

Name:
untitled

Location:
/data/v2.0

Full path:
/data/v2.0/untitled

Figure 6–3: The New Project window of the GS Reference Mapper application

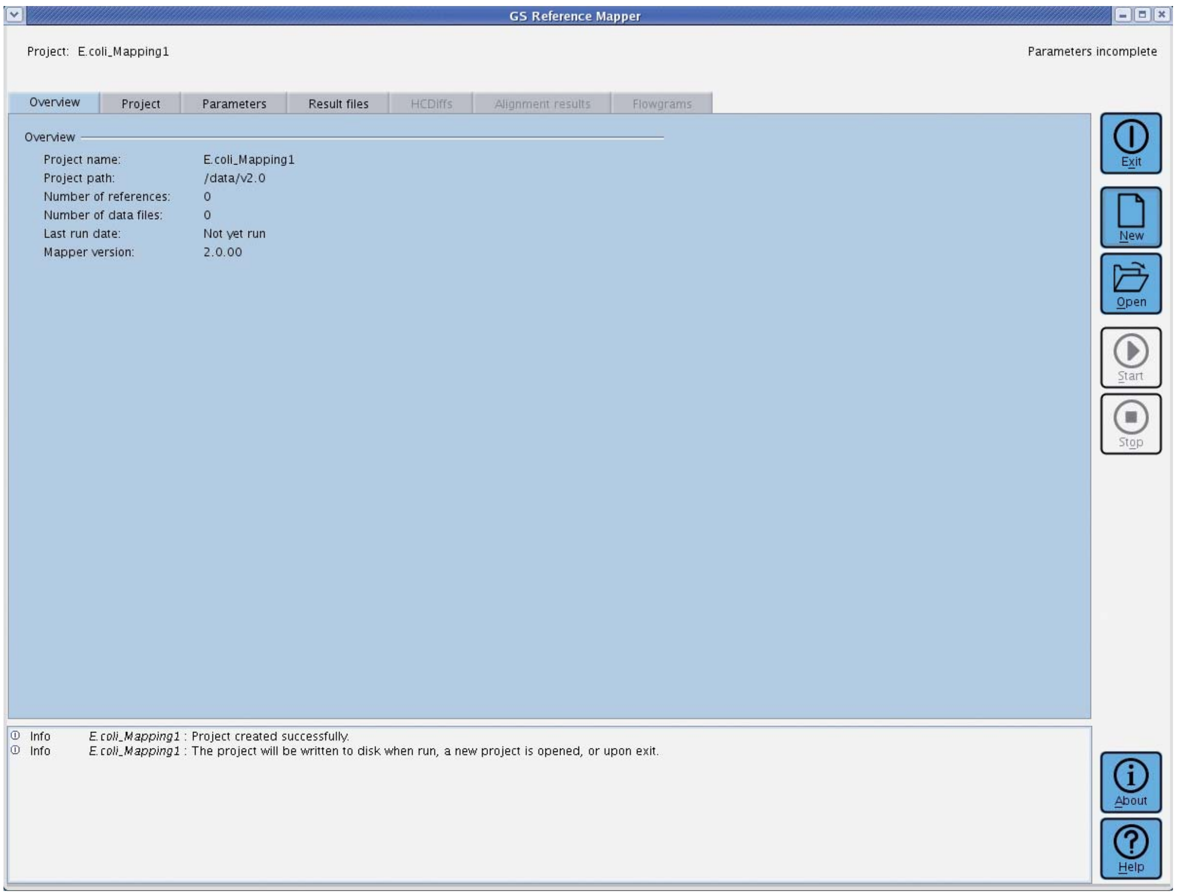
Type a name for the new project in the **Name** text field. To specify the project location, either type the full path in the **Location** text field, or click on the **Open Project** button to the right of the text field and use the "Select Project Location" window (not shown) to navigate to the directory where you want to create the new project. The **Full Path** field changes as you update the **Name** and **Location** fields. When you are satisfied with the location and name for the new project, click on the **OK** button. You can save a project by clicking on the **Yes** button of the **Save** prompt.

You may save the project to your hard drive, either by using the Exit button to exit the GS Reference Mapper application (you will be prompted to save before the application actually exists), or by adding read data and running the project by clicking the Start button (*i.e.* compute the mapping), in which case the project will automatically be saved prior to computation, as described in section 6.2.5, below.

6.2.4 Setting Reference Mapper Project Parameters

After clicking **OK** in the New Project window, the main application window changes to display the Overview tab for the project (Figure 6–4). This view summarizes various data about the project. The data displayed on this tab is updated as new information becomes available, which occurs when data files are added to the project and when a project computation completes.

The upper right hand corner of the application window indicates "Parameters incomplete" and the Start button is initially disabled because at least one Read Data file and one Reference File must be added to the project before a Mapping computation can be initialized. Other errors in input parameters can also cause this message to appear. Once all such errors have been corrected and at least one Read Data and Reference files are added to the project, the message will change to "Ready for analysis" and the Start button will become active.



Project: E.coli_Mapping1

Parameters incomplete

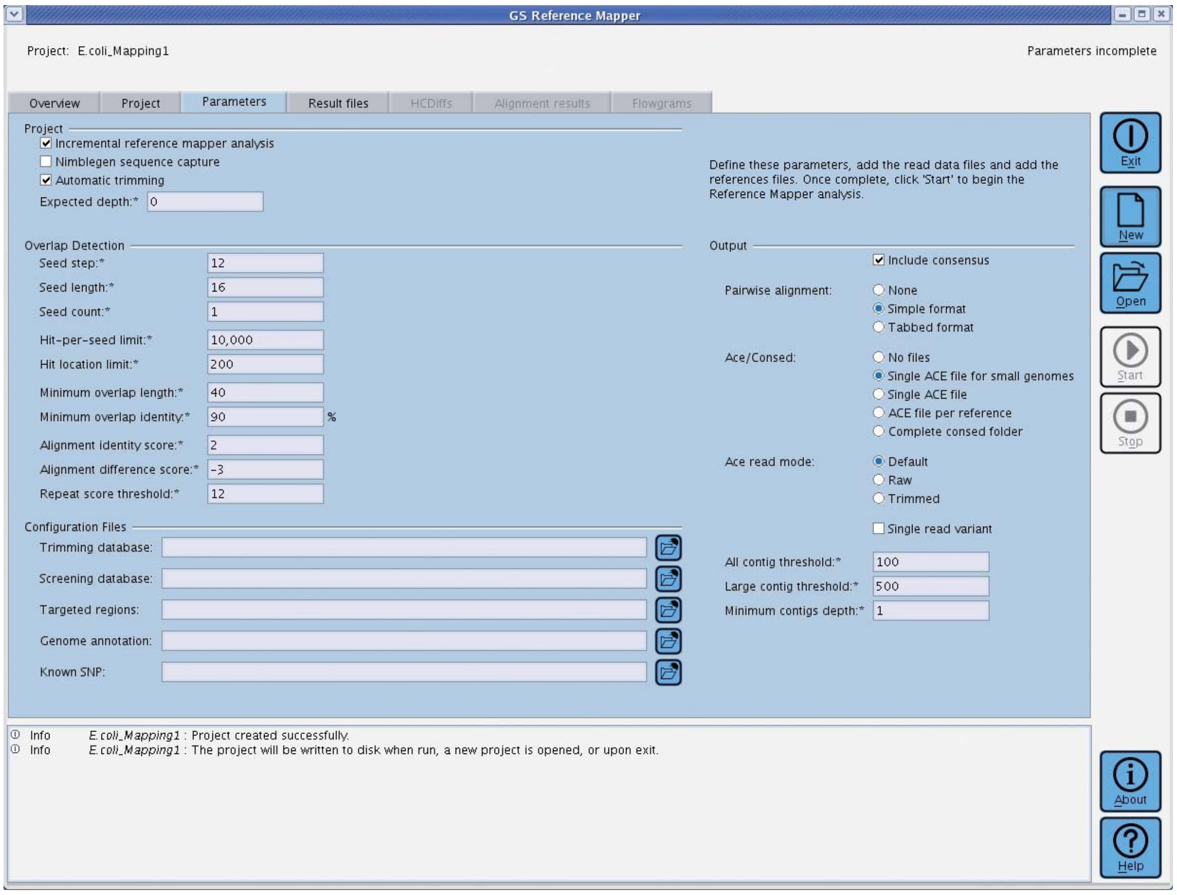
Overview

Project name:	E.coli_Mapping1
Project path:	/data/v2.0
Number of references:	0
Number of data files:	0
Last run date:	Not yet run
Mapper version:	2.0.00

```
Info    E.coli_Mapping1 : Project created successfully.
Info    E.coli_Mapping1 : The project will be written to disk when run, a new project is opened, or upon exit.
```

Figure 6–4: The Overview tab of a new project in the GS Reference Mapper application. This tab is used to summarize certain characteristics of an open mapping project. The values displayed are updated any time data files are added to the project and when an mapping computation completes.

Click on the Parameters tab of the main application window to display the parameters for the mapping project (Figure 6–5). For a new project, the mapping parameters that take numerical values are initially set at their default values, as shown. If an invalid value is entered into any of these fields, a small red “X” appears in the lower left corner of the field, and activates the “Parameters incomplete” warning in the upper right hand corner of the screen. Pausing the mouse over the red X reveals a tooltip indicating the allowed values for the parameter (Figure 6–6).



Project: E.coli_Mapping1

Parameters incomplete

Project
- [✓] Incremental reference mapper analysis
- [ ] Nimblegen sequence capture
- [✓] Automatic trimming

Expected depth:* 0

Define these parameters, add the read data files and add the references files. Once complete, click 'Start' to begin the Reference Mapper analysis.

Overlap Detection

Seed step:*	12
Seed length:*	16
Seed count:*	1
Hit-per-seed limit:*	10,000
Hit location limit:*	200
Minimum overlap length:*	40
Minimum overlap identity:*	90 %
Alignment identity score:*	2
Alignment difference score:*	-3
Repeat score threshold:*	12

Output
- [✓] Include consensus

Pairwise alignment: None [✓ Simple format] Tabbed format

Ace/Consed: No files [✓ Single ACE file for small genomes] Single ACE file; ACE file per reference; Complete consed folder

Ace read mode: [✓ Default] Raw; Trimmed

- [ ] Single read variant

All contig threshold:*	100
Large contig threshold:*	500
Minimum contigs depth:*	1

Configuration Files

Trimming database:

Screening database:

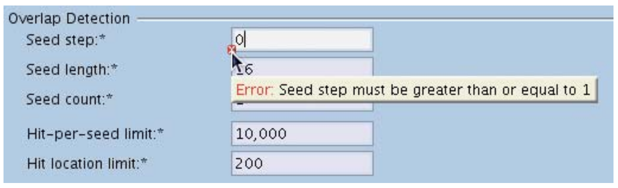
Targeted regions:

Genome annotation:

Known SNP:

```
Info    E.coli_Mapping1 : Project created successfully.
Info    E.coli_Mapping1 : The project will be written to disk when run, a new project is opened, or upon exit.
```

Figure 6–5: The Parameters tab of a new project in the GS Reference Mapper application. This tab is used to set the parameters and specify configuration files for a mapping computation. If horizontal and vertical scrollbars are showing, use them to view the entire set of parameters.



Overlap Detection

Seed step:*	0
Seed length:*	16
Seed count:*	
Hit-per-seed limit:*	10,000
Hit location limit:*	200

Error: Seed step must be greater than or equal to 1

Figure 6–6: Example of a tooltip revealed by pausing the mouse over the red X in the corner of a mapping parameter entry field, after entering an invalid parameter value

The following paragraphs describe the mapping parameters that you can control, how each of the parameters affects subsequent mapping operations, and the default and acceptable range values for the numerical parameters.

- Project Parameter
  - Incremental reference mapping analysis – If this check box is selected, new read data will be mapped into existing mapping alignments that were created during previous mapping computations in this project. If it is not selected, all the project data will be mapped to the reference anew each time a mapping is computed in this project, overwriting any existing mapping results.
    - Default: selected (Note: This value is not saved with a project, and is always reset to "selected" when a project is re-opened)
  - NimbleGen sequence capture – if this check box is selected, the read data will be treated as coming from a NimbleGen Sequence Capture experiment. Currently, this simply enables the automatic trimming of the Sequence Capture primers found on each read, but future software versions may enable additional functionality associated with this option.
    - Default value: not selected
  - Automatic trimming – if this check box is selected, by default, the ends of new reads are trimmed.
    - Default value: selected
  - Expected depth: typical number of reads expected to uniquely map to a single position in a multiple alignment
    - Default value: 0
    - Allowed values: 0 or greater, where a value of 0 tells the GS Reference Mapper to not use expected depth information in its computation
- Overlap Detection Parameters
  - Seed step – The number of bases between seed generation locations used in the exact k-mer matching part of the alignment of each read to the reference
    - Default value: 12
    - Allowed values: 1 or greater
  - Seed length – The number of bases used for each seed in the exact k-mer matching part of the alignment of each read to the reference (*i.e.* the "k" value of the k-mer matching)
    - Default value: 16
    - Allowed values: 6–16
  - Seed count – The number of seeds required in a window before an extension is made
    - Default value: 1
    - Allowed values: 1 or greater
  - Hit-per-seed limit – A filtering limit on the seeds found during the exact k-mer matching (if more than this number of seed matches are found at a lookup position in the query sequence, those seed matches are not tested as candidate alignments)
    - Default value: 10,000
    - Allowed values: 0 or greater
  - Hit location limit – If the sum total of all used seed matches identify more than this limit of different regions of the reference sequence as candidate alignment locations, the query read is marked as a repeat read (without performing the alignments at those locations)
    - Default value: 200
    - Allowed values: 0 or greater
  - Minimum overlap length – The minimum length of an alignment to use when aligning the reads to the reference
    - Default value: 40
    - Allowed values: 1 or greater

- Minimum overlap identity – The minimum percent identity of an alignment to use when aligning the reads to the reference
  - Default value: 90
  - Allowed values: 0–100
- Alignment identity score – The per-alignment-column identity score to use to sort the alignments of a read to the reference (for determining whether the read uniquely or multiply maps to the reference)
  - Default value: 2
  - Allowed values: 0 or greater
- Alignment difference score – The per-alignment-column difference score to use to sort the alignments of a read to the reference (for determining whether the read uniquely or multiply maps to the reference)
  - Default value: -3
  - Allowed values: 0 or less
- Repeat score threshold – The threshold used to declare alignments "unique" or "multiple". If an input read aligns to more than one position in the reference sequence(s), an alignment score is calculated for each alignment by summing the identity or difference score for each alignment column; if the best scoring alignment is larger than any other by at least this threshold value, it is marked as "uniquely mapped". Otherwise, it is "multiply mapped".
  - Default value: 12
  - Allowed values: 0 or greater

- Configuration Files
  - Trimming database – The path to a FASTA file of sequences to be used to trim the ends of input reads (for cloning vectors, primers, adapters or other end sequences).
  - Screening database –The path to a FASTA file of sequences to be used to screen the input reads for contaminants. A Read that completely aligns against the screening database is removed so that it is not used in the computation; if only part of a read aligns with the screening database, no action is taken.
  - Targeted regions – A string argument telling the GS Reference Mapper to only output specific regions of the reference, and the reads aligned to those regions.

    The reads are mapped against the complete reference, but the output files will only report on the specified regions. The string value can be
    - a "accno:#-#" string, *e.g.* "chr9:10549-30594" specifies bases 10549 to 30594 of the chr9 reference sequence;
    - the path to a file containing lines of "accno:#-#" strings (one per line); or
    - the path to a GFF DAR file created during a NimbleGen Sequence Capture experiment, describing the "primary_target_region" regions of the experiment.
  - Genome annotation – The path to an annotation file describing the gene/coding-region annotations for the reference sequences, so that the variation detection (HCDiffs tab) can report gene names and protein translations of any identified variations in gene regions. The format of this file must match that of the GoldenPath "refGene.txt" file.
  - Known SNP annotation – The path to an annotation file describing the known SNP information for the reference sequences, so that the variation detection (HCDiffs tab) can link identified variations to the known SNP information. The format of this file must match that of the GoldenPath snp128.txt file.
  - Long file paths will be displayed in full by pausing the mouse cursor on the text area.

- Output Parameters
  - Include consensus – If this check box is selected, the application will generate all the output files, including the output files related to the generation of contigs and consensus sequence information. If it is not selected, the application will perform the alignments of read to the reference, and will output files associated with the read alignments, but will not output files or metrics involved with contig or consensus information.
    - Default: selected (Note: This value is not saved with a project, and is always reset to “selected” when a project is re-opened)
  - Pairwise Alignment – Determines whether the 454PairAlign.txt file is output; this file contains the alignments of read to the reference used by the GS Reference Mapper.
    - None – the file is not generated
    - Simple format – the file contains a human-readable view of the alignments
    - Tabbed format – the file contains tab-delimited lines of the overlaps

    Default is “Simple”. See section 6.4.2.11 for a description of the 454PairAlign.txt file.
  - Ace/Consed – Determines whether or what form of ACE file(s) are output by the application.
    - No files – no ACE file is generated
    - Single ACE file for small genomes – a single ACE file is generated if the reference is smaller than 40 million bases and fewer than 4 million input reads are given
    - Single Ace file – a single ACE file is generated containing the multiple alignments of all reads and contigs to the reference sequence(s)
    - ACE file per reference – a separate ACE file is generated for each sequence in the reference sequence file(s)
    - Complete consed folder – a “consed” folder is generated, containing all the directories and files necessary to run the consed software

    Default is “Single ACE file for small genomes”. See section 6.4.2.15 for a description of the 454Contigs.ace file.
  - Single read variant – If selected, SNPs and other local variations that occur in a single read will be output in the 454AllDiffs.txt file. If not selected, only variations that occur in multiple reads will be output.
    - Default: not selected
  - Ace read mode – when the ACE file is generated, output reads using either the raw, complete basecalled read or using just the trimmed portion of the reads (after low quality, vector and key trimming)
    - Default is set to output the entire raw reads, except for 454 Paired-End reads, which are output as trimmed (so that the linker sequence and the sequence of the other half of the Paired-End read don’t clutter the alignment)
    - Raw is to output the entire sequence of reads
    - Trimmed is to output trimmed reads
  - All contig threshold – the minimum number of bases for a contig to be output in the 454AllContigs.fna file
    - Default: 100
  - Large contig threshold – the minimum number of bases for a contig to be output in the 454LargeContigs.fna file
    - Default: 500
  - Minimum contigs depth – The minimum read depth used to determine the ends of contigs (*i.e.* contigs are ended whenever the read depth falls below this threshold, and started again when the depth rises to the threshold or above).
    - Default: 1

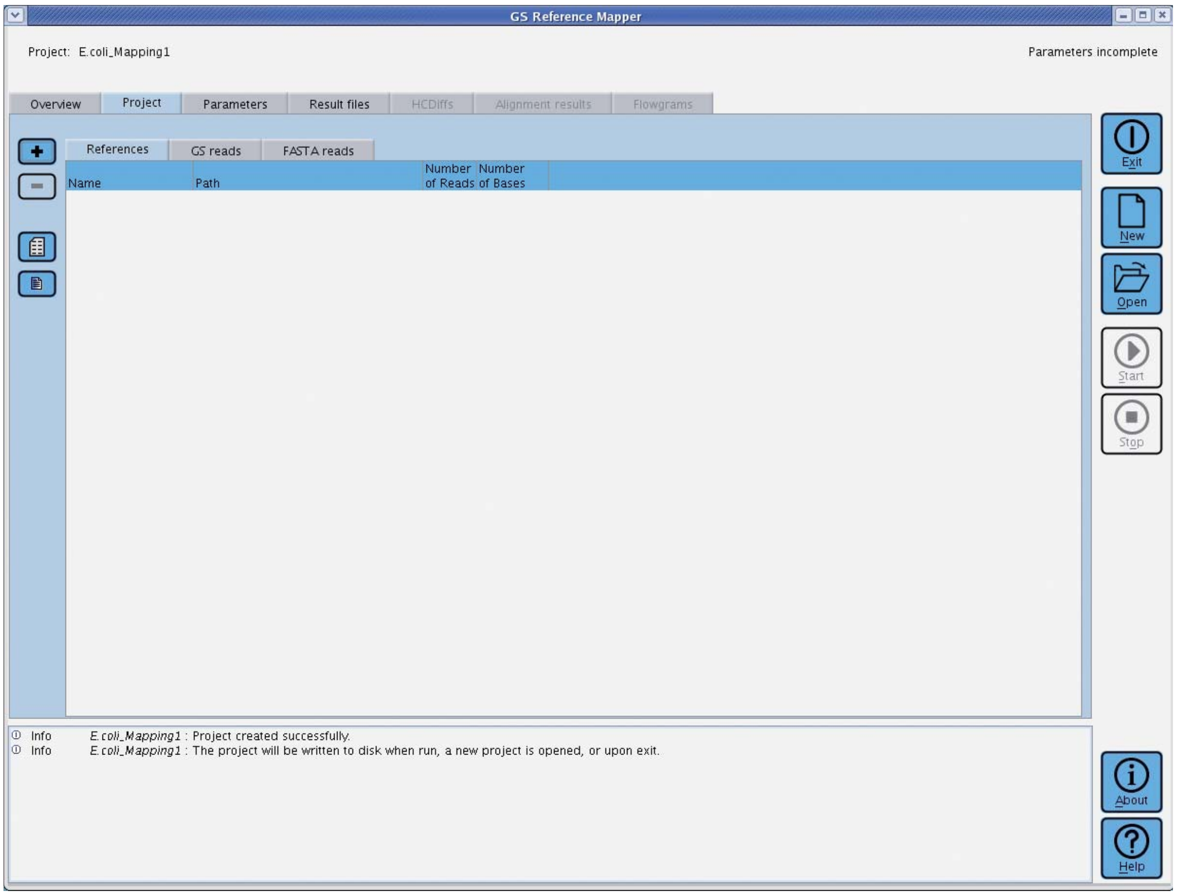
External files: FASTA files are not actually copied to the project directory. Rather, the file path is simply recorded in the project setup. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies to all instances of FASTA files that can be added to an Assembly project, whether done via the GUI or the command line:

- FASTA reads files, including Sanger reads (Project tab/FASTA reads sub-tab)
- Trimming database files (Parameters tab)
- Screening database files (Parameters tab)
- Reference sequence FASTA files (Project tab/References sub-tab)

Unlike with the GS Read Data files, no symbolic link to the file is created for FASTA files.

6.2.5 Adding Read Data to a Mapping Project

The Project tab of the main GS Reference Mapper application window contains three sub-tabs (References, GS reads, and FASTA reads) that are used to add reference files, GS Read Data files (SFF files and/or “PicoTiterPlate device regions”), and FASTA data files to mapping projects (Figure 6–7). The **Start** button in the toolbar, which is used to start a mapping computation, remains disabled and the “Parameters incomplete” message remains visible, until at least one GS Read Data file or FASTA data file (and one reference sequence FASTA file; see section 6.2.6) have been added to the project.

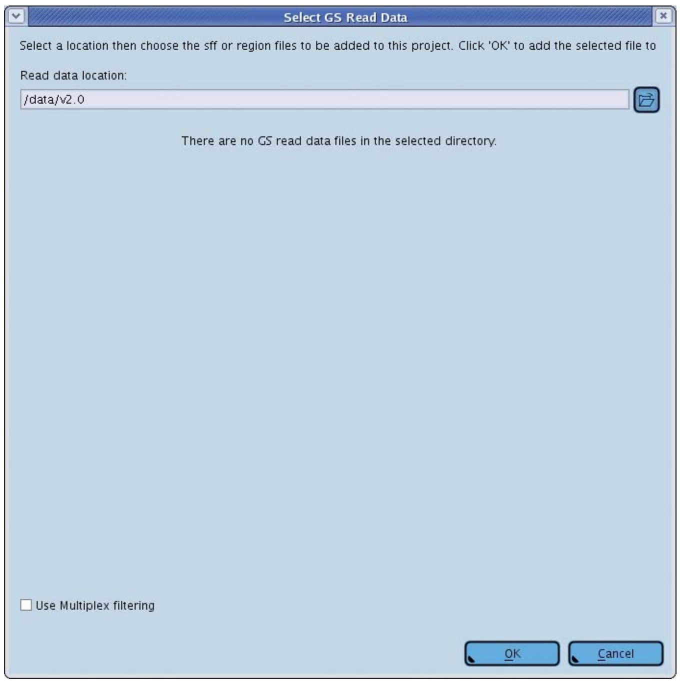


```
Info    E.coli_Mapping1 : Project created successfully.
Info    E.coli_Mapping1 : The project will be written to disk when run, a new project is opened, or upon exit.
```

Figure 6–7: The Project tab of the GS Reference Mapper application

6.2.5.1 Adding GS Read Data Sets to a Mapping Project

To add GS read data sets to a project, click on the GS reads sub-tab, then click the **Add** button ⊞ to open the "Select GS Read Data" window (Figure 6–8). The top portion of this window lists, for the specified location, the GS Read Data files that can be added to the project; if no files exist at this location, the window displays the message: "There are no GS Read Data files in the selected directory" (as in Figure 6–8). The bottom area (MID) is used to append MID filtering information to the read file(s), if MIDs were used in the design of your experiment and they are relevant to the mapping analysis. By default, the MID section is not visible, but it can be viewed by selecting the "Use MID/Multiplex filtering" checkbox located in the lower left corner of the window (Figure 6–9).


Select GS Read Data

Select a location then choose the sff or region files to be added to this project. Click 'OK' to add the selected file to

Read data location:

/data/v2.0

There are no GS read data files in the selected directory.

☐ Use Multiplex filtering

Figure 6–8: The Select GS Read Data window of the GS Reference Mapper application. In this example, the selected folder does not contain any SFF files.

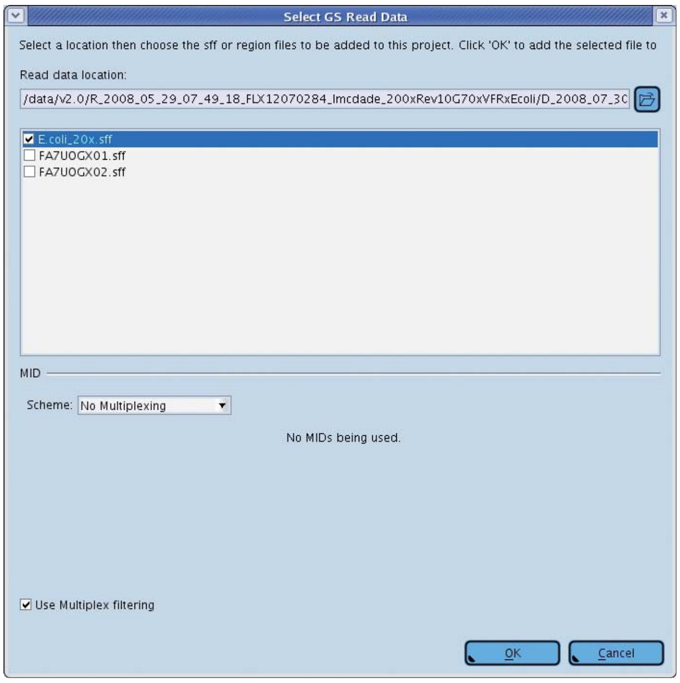

Select GS Read Data
Select a location then choose the sff or region files to be added to this project. Click 'OK' to add the selected file to
Read data location:
/data/v2.0/R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/D_2008_07_30
☑ E.coli_20x.sff
☐ FA7UOGX01.sff
☐ FA7UOGX02.sff
MID
Scheme: No Multiplexing
No MIDs being used.
☑ Use Multiplex filtering
OK
Cancel

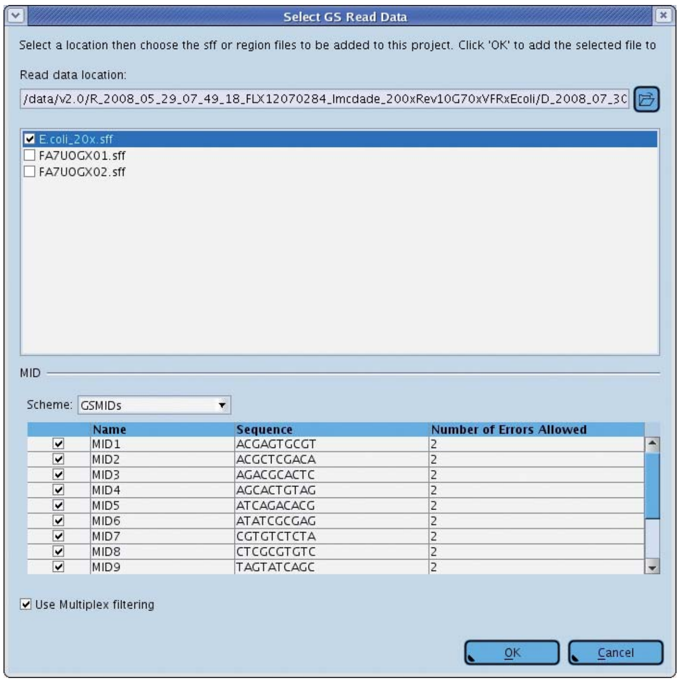
Figure 6–9: The Select GS Read Data window of the GS Reference Mapper application, with the "Use MID/Multiplex filtering" checkbox selected and the MID area showing

To change to a different location that contains Read Data files, click on the **"Change the read data location"** button (with the open-folder icon) to the right of the text field and use the "Select Read Data Directory" window (not shown) to navigate to and select the new location. You may also directly type or paste the desired path into the "File Name:" field of the navigation window, rather than browsing through the file system for the directory of interest. While using this interface, only directories will appear, not the files within those directories. When you return from the Select GS Read Data window, the list of available Read Data files found at the new location will be refreshed and appear in the field below the Read data location text area (as seen on Figure 6–9). Use the check boxes to select the file(s) you want to include in the mapping project. If MID multiplexing is not required for these files, select "No Multiplexing" for the MID scheme (drop down) menu (or do not select the "Use Multiplex filtering" checkbox), and click the **OK** button.

When the Use Multiplex Filtering checkbox is selected, special controls for MID filtering are displayed in the bottom half of the Select GS Read Data window. Multiplex Identifiers (MIDs) allow you to design an experiment whereby multiple libraries are prepared using distinct MID tags and sequenced together, in the same region of a PTP device. (This can greatly improve the workflow and cost effectiveness of your experiment.) In these conditions, the reads of all the pooled libraries are contained in a single Read Data (.sff) file; the MID controls on this window allow you to filter the reads from one (or more) library(ies) in the selected Read Data files, for inclusion in the Mapping project. Note, this filtering operation does not produce a different Mapping output for each specified MID. Rather, the reads of the SFF files are scanned for the presence of MIDs, and the reads containing the selected MIDs are made eligible for mapping in the project, as a group. Any reads without the specified MIDs will be effectively ignored. To produce independent Mapping output for different MIDs (or sets of MIDs), you must create independent Mapping projects for each of the desired MID subsets of the data.

The MID filtering information is contained in 'MID schemes', which list a set of MID tags along with their exact sequences and the number of sequencing errors permitted in each MID string that will still allow the tag to be recognized. The Genome Sequencer FLX System software comes with a standard MID scheme (GSMIDs) that contains 14 MIDs (of which MIDs 1 to 12 match the "MID Adaptors" available in kit form), each with two errors allowed. These MIDs are all 10-mers that were carefully chosen such that two errors in any of them can be corrected without confusing one MID with another MID of the set.

By default, no MID/Multiplexing scheme is associated with Read Data files. To use an MID scheme, select it using the Scheme drop down menu. When an MID scheme is selected, a table of the MIDs present in that scheme is displayed, as shown in Figure 6–10. Use the checkboxes on the left to select or deselect the MIDs to be included in the mapping. The names, sequences and error limits of the MIDs selected will be used to filter the reads (from the Read Data sets selected in the top part of the window). The result is that only the reads that contain the selected MIDs, within the number of errors specified in the scheme, will be included in the project.



Select a location then choose the sff or region files to be added to this project. Click 'OK' to add the selected file to

Read data location:

```
/data/v2.0/R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/D_2008_07_3C
```

- [✓] E.coli_20x.sff
- [ ] FA7UOGX01.sff
- [ ] FA7UOGX02.sff

MID

Scheme: GSMIDs

	Name	Sequence	Number of Errors Allowed
✓	MID1	ACGAGTGCGT	2
✓	MID2	ACGCTCGACA	2
✓	MID3	AGACGCACTC	2
✓	MID4	AGCACTGTAG	2
✓	MID5	ATCAGACACG	2
✓	MID6	ATATCGCGAG	2
✓	MID7	CGTGTCTCTA	2
✓	MID8	CTCGCGTGTC	2
✓	MID9	TAGTATCAGC	2

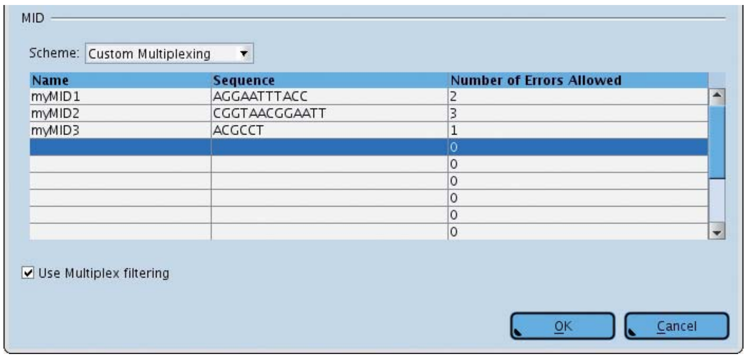
[✓] Use Multiplex filtering

Figure 6–10: The Select GS Read Data window of the GS Reference Mapper application's main window, showing the table of MIDs present in the MID scheme GSMIDs

The list of MID schemes provided in the Scheme drop down menu is read from the file MIDConfig.parse. For pre-existing MID schemes, the names and sequences of the MIDs, and the number of errors allowed, cannot be changed in the Select GS Read Data window. To modify these values, or to add new user-specified MID schemes, edit the MIDConfig.parse file using any text editor. The standard location of this file is /opt/454/config/MIDConfig.parse; more generally, if the software is installed to the */install/directory* folder, the configuration file will be found at */install/directory/*config/MIDConfig.parse. See section 13.4.15 to view the default MIDConfig.parse file and for a description of how to edit it.

You can also create new MIDs directly in the Select GS Read Data window, by selecting "Custom Multiplexing" from the Scheme drop-down menu. In this case, a table will appear and new MID names, sequences, and error limits can be typed into the table (Figure 6–11). However, these names, sequences and error limits will only apply to the current project.

There is no formal limit to the number or length of custom MIDs that one can define in a Mapping project. However, efficient demultiplexing requires that the MID sequences be divergent enough that the software will be able to differentiate them from one another (within the "number of errors allowed" specified).


MID

Scheme: Custom Multiplexing

Name	Sequence	Number of Errors Allowed
myMID1	AGGAATTTACC	2
myMID2	CGGTAACGGAATT	3
myMID3	ACGCCT	1
		0
		0
		0
		0
		0
		0

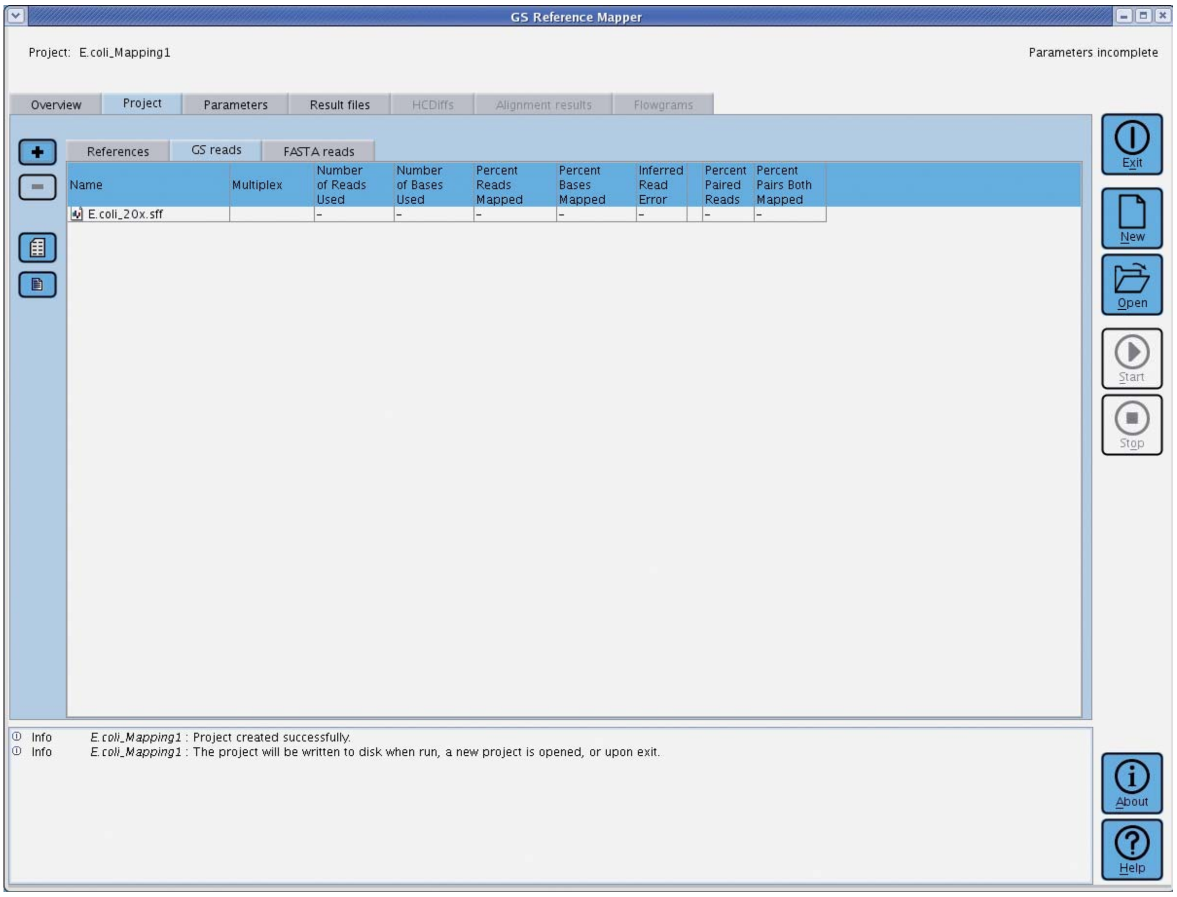
☑ Use Multiplex filtering

Figure 6–11: Creating custom MIDs

Once the MID scheme has been specified, click the **OK** button to add the selected data files to the list of GS read data files. These are displayed in tabular format (Figure 6–12). Except for the Name column and the Multiplex column, which contain comma-delimitted lists of the MIDs associated with the file, all columns with run statistics are initially filled with dashes. These data are updated in the table each time the project runs to completion.

You can add any number of Read Data files to a project, both by selecting multiple files present in a given directory (see Figure 6–9), or by using the **Add** button as many times as you need to navigate to all the files you want to include. You can even add multiple files that share the same name into a project, as long as they have different path locations in the file system (*e.g.* "/dir1/path1/reads.sff" and "/dir2/otherpath2/reads.sff"). In such a case, both files will be added to the Read Data list and displayed using the same name. To see the path to a file listed in the Read Data area of the main window (and the file's last modification date), pause the mouse over the filename of interest and a tooltip containing the file's path will be displayed. Files that have failed validation will have a red X left of the file name as an indication of failure. Pausing the mouse cursor over the red X will bring up a tooltip explaining the problem encountered.

External files: GS Read Data files (SFF files) are not actually copied to the project directory. Rather, a symbolic link to the file is created and placed in the project's sff directory. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies whether the files are added via the GUI or the command line.



Project: E.coli_Mapping1

Parameters incomplete

Name	Multiplex	Number of Reads Used	Number of Bases Used	Percent Reads Mapped	Percent Bases Mapped	Inferred Read Error	Percent Paired Reads	Percent Pairs Both Mapped
E.coli_20x.sff		–	–	–	–	–	–	–

```
Info    E.coli_Mapping1 : Project created successfully.
Info    E.coli_Mapping1 : The project will be written to disk when run, a new project is opened, or upon exit.
```

Figure 6–12: The GS Read Data sub-tab of the Project tab, in the GS Reference Mapper application's main window, showing 1 SFF file that was added to the project.

In certain circumstances, you may want to apply a different MID selection to different Read Data sets, within the same project. For example, you may have multiple libraries of the same organism, made with different MIDs (or with/without MIDs), that you want to map together against one or more reference sequences. In this case, select the Read Data set(s) that need to be filtered with any given MID (or combination of MIDs), and click **OK** to add the MID-filtered data file(s) to the project. Back on the Project tab of the GS Reference Mapper window (with the GS Reads sub-tab active) (Figure 6–12), click the **Add** button (⊞) again, and select other Read Data sets, to be filtered with different MIDs.

You can also remove currently included Read Data files from a project, by selecting them from the list in the Read Data area of the Mapper application's Parameters tab, and then clicking the **Remove** button ⊟ .

The **Remove** button removes the selected Read Data files from the list, but does not immediately remove the file(s) from the project's sff sub-directory or from any computed mapping results. Actual removal of the files from the project (and of the reads they contain from an existing alignments computed therefrom), occurs only when the mapping is re-computed (see section 6.2.7).

6.2.5.2 Adding FASTA Read Data Sets to a Mapping Project

The procedure for adding FASTA data files to a Mapping project is a nearly identical to the one for adding GS Read data (section 6.2.5.1). The only difference is that when a directory containing FASTA files is selected, the software examines the files in that directory to determine which files are FASTA files. This can take some time if there are many files in the directory, so a progress bar is displayed to show the progress of the search.

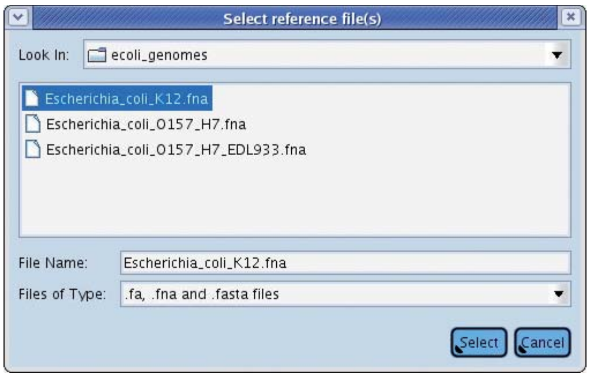
External files: FASTA files are not actually copied to the project directory. Rather, the file path is simply recorded in the project setup. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies to all instances of FASTA files that can be added to a Mapping project, whether done via the GUI or the command line:

- FASTA read files, including Sanger reads (Project tab/FASTA reads sub-tab)
- Trimming database files (Parameters tab)
- Screening database files (Parameters tab)
- Reference sequence FASTA files (Project tab/References sub-tab)

Unlike with the GS Read Data files, no symbolic link to the file is created for FASTA files.

6.2.6 Adding Reference DNA Sequence Files to a Mapping Project

When the References sub-tab is the currently selected tab, the **Add** button on the left-hand side of the Project tab (see Figure 6–7, above) can also be used to add reference sequence (FASTA) files to the project. Reference files contain the DNA sequence(s) against which the reads in the Read Data files will be aligned. The **Start** button in the toolbar, which is used to start a mapping computation, remains disabled until at least one FASTA file (and one GS Read Data or FASTA read file; see section 6.2.5) has been added to the project. Clicking on the **Add** button (with the References sub-tab active) opens the "Select reference file(s)" window (Figure 6–13), from which reference DNA sequence FASTA files can be selected and added to the project.



Look In: ecoli_genomes

Escherichia_coli_K12.fna
Escherichia_coli_O157_H7.fna
Escherichia_coli_O157_H7_EDL933.fna

File Name: Escherichia_coli_K12.fna
Files of Type: .fa, .fna and .fasta files

Figure 6–13: The Select reference file(s) window of the GS Reference Mapper application

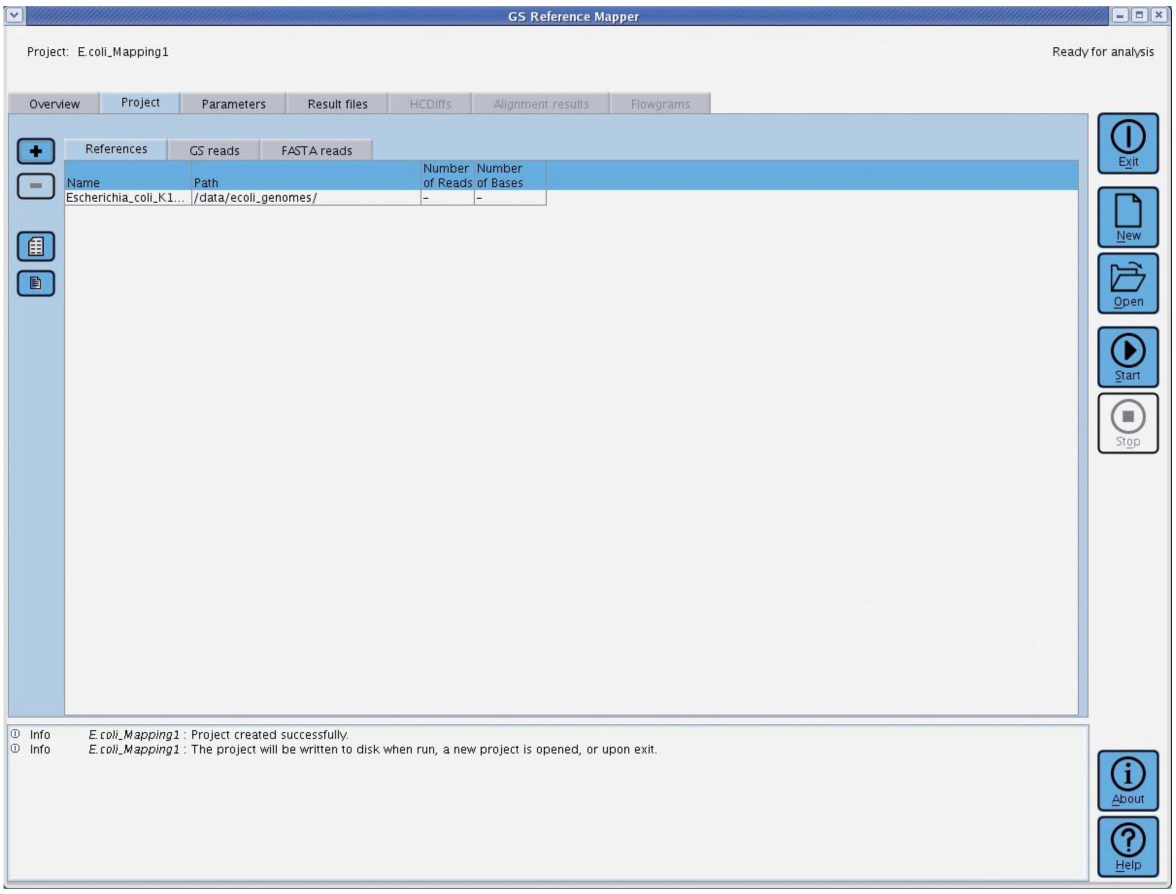
Although in many cases the file name extension for FASTA files may be **.fna**, there is no standardized extension for such files. If you cannot see your FASTA file, try setting the "Files of Type" field to "All Files".

External files: FASTA files are not actually copied to the project directory. Rather, the file path is simply recorded in the project setup. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies to all instances of FASTA files that can be added to a Mapping project, whether done via the GUI or the command line:

- FASTA read files, including Sanger reads (Project tab/FASTA reads sub-tab)
- Trimming database files (Parameters tab)
- Screening database files (Parameters tab)
- Reference sequence FASTA files (Project tab/References sub-tab)

Unlike with the GS Read Data files, no symbolic link to the file is created for FASTA files.

Reference DNA sequence files that have been added to the project appear in the References sub-tab of the Project tab (Figure 6–14). You can add any number of reference sequence (FASTA) files to a project, either by selecting multiple files present in a single directory (see Figure 6–13) or by using the **Add** button as many times as you need to navigate to all the files you want to include; in this case, the application treats the reference as the union of all the sequences in all the FASTA files (*i.e.* all the sequences are considered as one “genome” for the purposes of determining unique vs. repeat matches). The two rightmost columns of the references table remain empty until the project has been run, at which time these fields are populated.



Project: E.coli_Mapping1

Ready for analysis

Name	Path	Number of Reads	Number of Bases
Escherichia_coli_K1...	/data/ecoli_genomes/	–	–

```
Info    E.coli_Mapping1 : Project created successfully.
Info    E.coli_Mapping1 : The project will be written to disk when run, a new project is opened, or upon exit.
```

Figure 6–14: The References sub-tab

You can also remove currently included reference sequence FASTA files from a project, by selecting them from the list in the References area of the Mapper application’s Parameters tab, and then clicking the **Remove** button (- ; Figure 6–14).

The **Remove** button removes the selected reference sequence files from the list, but does not immediately remove the file(s) from any computed mapping results. Actual removal of the files from the project (and of the read alignments computed therefrom), occurs only when the mapping is re-computed (see section 6.2.7).

6.2.7 Computing the Read Alignments to the Reference

6.2.7.1 Computing the Alignments to the Reference for the First Time

When at least one Read Data file and one Reference sequence FASTA file have been added to the project and all parameters are within acceptable limits, the **Start** button becomes enabled (*e.g.* see Figure 6–14, above). Press the **Start** button to carry out the mapping (alignment) computation of the current project, *i.e.* as defined in the Project and Parameters tabs. As the computation progresses, the **Start** button is disabled, periodic progress messages appear in the progress box, at the bottom of the application's main window, and the current stage of the mapping computation along with a progress bar are displayed in the upper right corner of the window. When the mapping computation is complete, the **Start** button is re-enabled and the message "Ready for analysis" appears again in the upper right corner of the application window.

6.2.7.2 Re-Computing the Alignments to the Reference

After a mapping has been performed on a project, the alignment computation can be repeated on that project – with the same Read Data and reference sequence(s) or after addition or removal of one or more files – with the same mapping parameter settings or with different ones.

6.2.8. Stopping a Mapping Project Computation In Progress

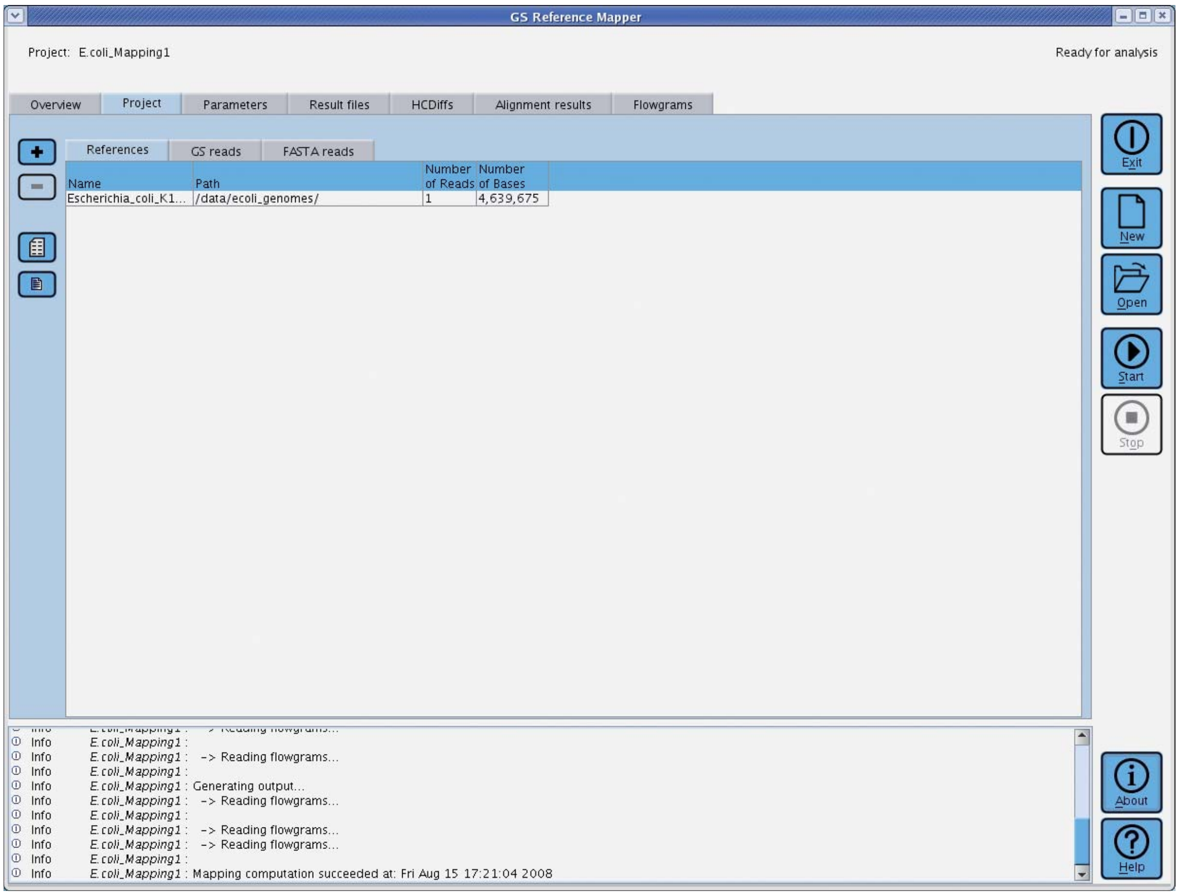
Clicking the **Stop** button while a Mapping project computation is in progress halts the computation. The effect may not be instantaneous, however, with any delay being a function of the complexity of the mapping being performed and of the stage of the computation at the time it was interrupted; an informational message appears in the progress box at the bottom of the application's main window, containing the text "Warning: stopRun called for this project. Stopping….". When the mapping computation has been halted, the **Stop** button is disabled and the **Start** button re-enabled.

6.2.9 Viewing the Results of a Mapping Computation

After a Mapping project has been computed, the results of the mapping can be viewed on the Overview, Project, Result Files, Alignment Results, and Flowgram tabs of the GS Reference Mapper application's main window.

6.2.9.1 Viewing the Reference, GS read, and FASTA read metadata: the Project Tab

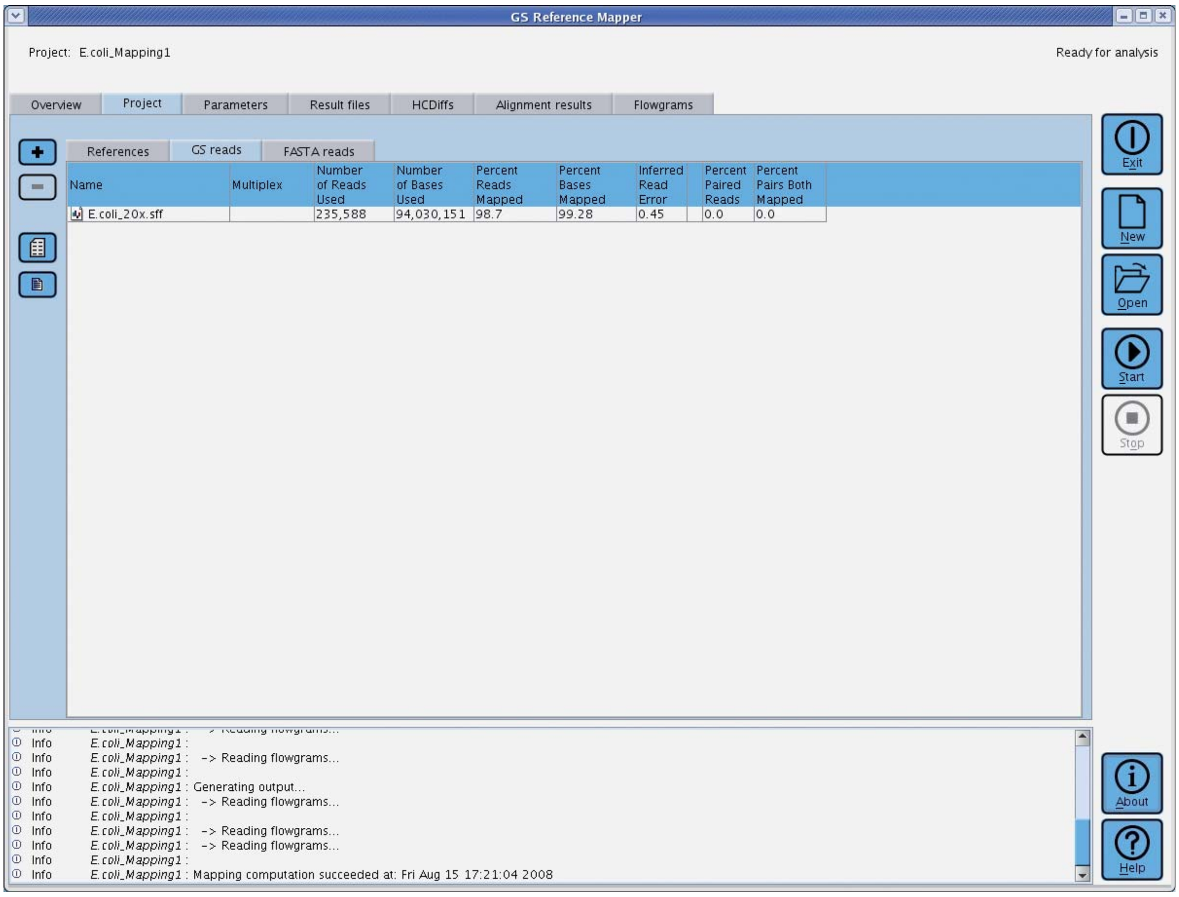
After a successful mapping, the data for the various columns of the reads in the References, GS reads, and FASTA reads tables will be updated (Figure 6–15 and Figure 6–16). Placing the mouse pointer over any cell in the table brings up a tool tip with additional information about that item. A successful mappping computation also updates data in the Overview tab.



Name	Path	Number of Reads	Number of Bases
Escherichia_coli_K1...	/data/ecoli_genomes/	1	4,639,675

```
Info   E.coli_Mapping1 :
Info   E.coli_Mapping1 :   -> Reading flowgrams...
Info   E.coli_Mapping1 :
Info   E.coli_Mapping1 : Generating output...
Info   E.coli_Mapping1 :   -> Reading flowgrams...
Info   E.coli_Mapping1 :
Info   E.coli_Mapping1 :   -> Reading flowgrams...
Info   E.coli_Mapping1 :   -> Reading flowgrams...
Info   E.coli_Mapping1 :
Info   E.coli_Mapping1 : Mapping computation succeeded at: Fri Aug 15 17:21:04 2008
```

Figure 6–15: An example of the GS Reference Mapper application window after a successful mapping computation. Here, the References sub-tab of the Project tab has been selected.



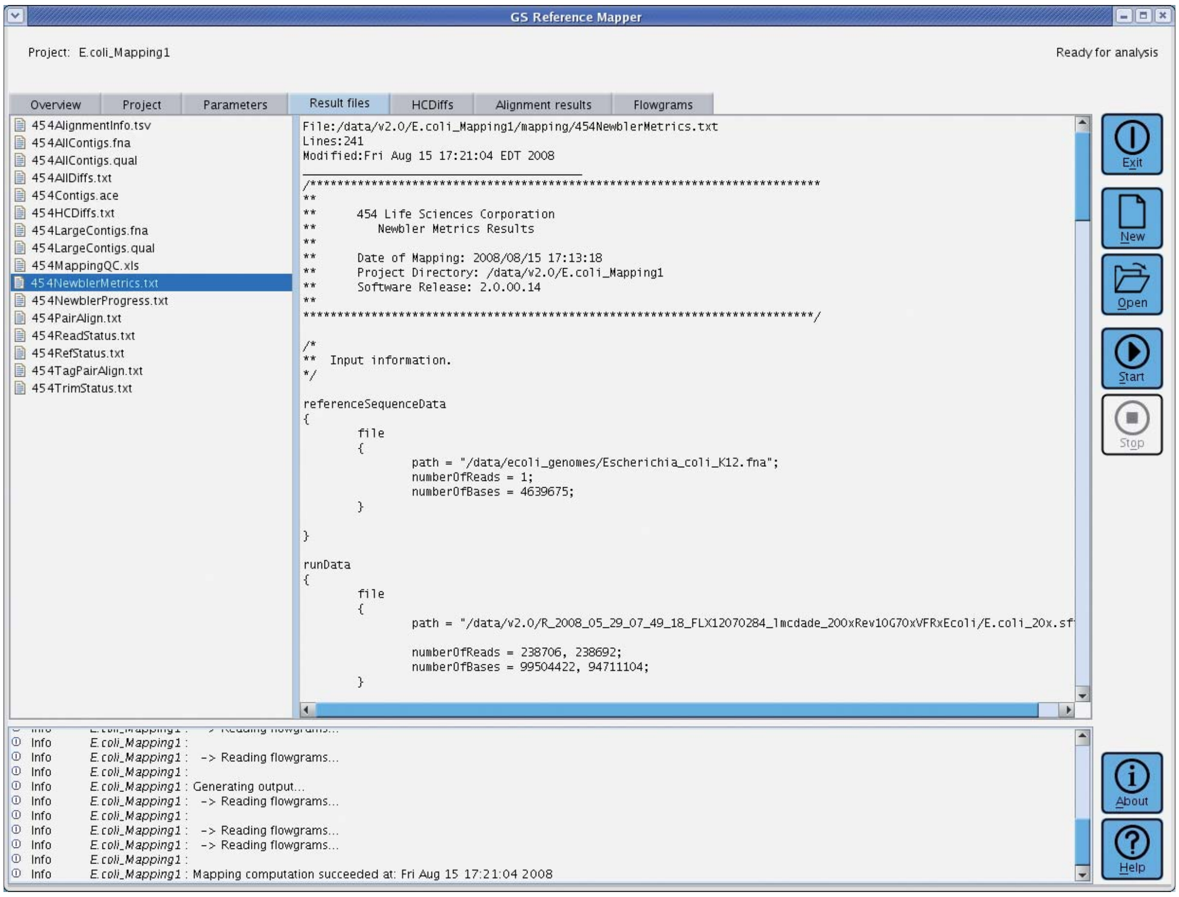
Name	Multiplex	Number of Reads Used	Number of Bases Used	Percent Reads Mapped	Percent Bases Mapped	Inferred Read Error	Percent Paired Reads	Percent Pairs Both Mapped
E.coli_20x.sff		235,588	94,030,151	98.7	99.28	0.45	0.0	0.0

```
Info      E.coli_Mapping1 :
Info      E.coli_Mapping1 :   -> Reading flowgrams...
Info      E.coli_Mapping1 :
Info      E.coli_Mapping1 : Generating output...
Info      E.coli_Mapping1 :   -> Reading flowgrams...
Info      E.coli_Mapping1 :
Info      E.coli_Mapping1 :   -> Reading flowgrams...
Info      E.coli_Mapping1 :   -> Reading flowgrams...
Info      E.coli_Mapping1 :
Info      E.coli_Mapping1 : Mapping computation succeeded at: Fri Aug 15 17:21:04 2008
```

Figure 6–16: An example of the GS Reference Mapper application window after a successful mapping computation. Here, the GS reads sub-tab of the Project tab has been selected.

6.2.9.2 Viewing the Mapping Metrics Files: the Result Files Tab

The Result Files tab allows you to view the various metrics files generated by the GS Reference Mapper software (described in detail in Section 6.4). Using the list in the left-hand panel of the tab, click on the name of the file output you want to examine; its content will be displayed on the right-hand panel of the tab (Figure 6–17). For very long files, such as the sequences of all contigs mapped to a large genome, this view displays the file truncated to 50,000 lines. When this occurs, a note to that effect specifies that it is so both at the beginning and at the end of the display. The file itself remains intact, however.



```
File:/data/v2.0/E.coli_Mapping1/mapping/454NewblerMetrics.txt
Lines:241
Modified:Fri Aug 15 17:21:04 EDT 2008
_______________________________________
/***************************************************************************
**
**      454 Life Sciences Corporation
**         Newbler Metrics Results
**
**      Date of Mapping: 2008/08/15 17:13:18
**      Project Directory: /data/v2.0/E.coli_Mapping1
**      Software Release: 2.0.00.14
**
***************************************************************************/


/*
**  Input information.
*/

referenceSequenceData
{
        file
        {
                path = "/data/ecoli_genomes/Escherichia_coli_K12.fna";
                numberOfReads = 1;
                numberOfBases = 4639675;
        }

}

runData
{
        file
        {
                path = "/data/v2.0/R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/E.coli_20x.sf

                numberOfReads = 238706, 238692;
                numberOfBases = 99504422, 94711104;
        }
```

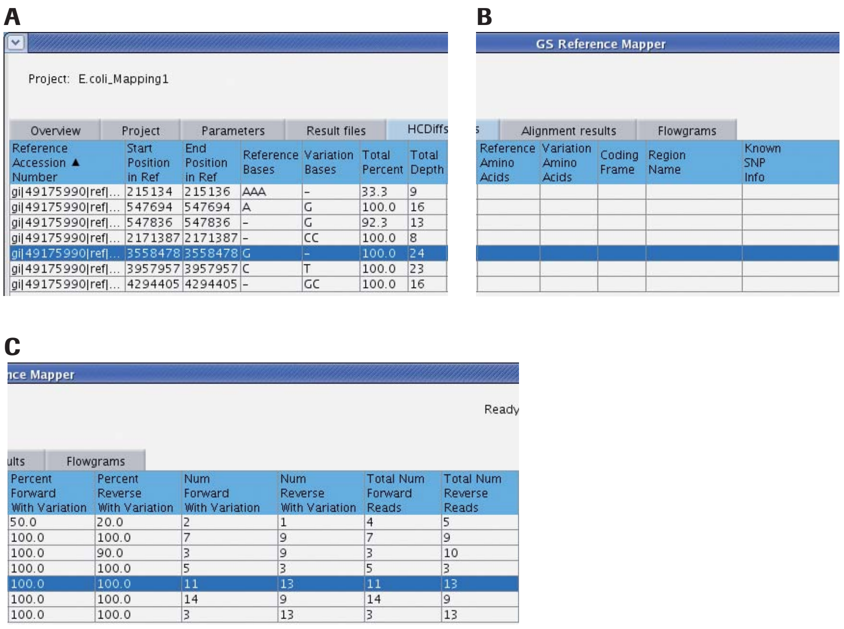
```
Info    E.coli_Mapping1 :
Info    E.coli_Mapping1 :  -> Reading flowgrams...
Info    E.coli_Mapping1 :
Info    E.coli_Mapping1 : Generating output...
Info    E.coli_Mapping1 :  -> Reading flowgrams...
Info    E.coli_Mapping1 :
Info    E.coli_Mapping1 :  -> Reading flowgrams...
Info    E.coli_Mapping1 :  -> Reading flowgrams...
Info    E.coli_Mapping1 :
Info    E.coli_Mapping1 : Mapping computation succeeded at: Fri Aug 15 17:21:04 2008
```

Figure 6–17: An example of the GS Reference Mapper application main window (Result Files tab) after a successful mapping computation. Clicking on a file from the list in the left column causes its contents to be displayed in the right panel. Files containing more than 50,000 lines will be truncated and flagged as such.

6.2.9.3 Viewing the High Confidence Differences: the HCDiffs Tab

The HCDiffs tab enables the user to view base variation in the aligned reads from the data set, compared to the reference sequences, as well as protein and SNP information about the region of the difference, when available. The tab can be divided in three main functional sections (Figure 6–18):

- The leftmost columns (from Reference Accession Number to Total Depth) contain general information on the high confidence differences (HCDiffs) that were found in the computed data, between the reads and the reference sequences.
- The center columns (from Reference Amino Acids to Known SNP Info) show gene annotation or known SNP file information on the regions of the HCDiffs. For such information to be available, Genome Annotation and Known SNP databases must have been specified on the Parameters Tab.
- The remaining columns on the right (from Percent Forward to Total Num Reverse Reads) contain a detailed breakdown of the computation data shown on the left.



Project: E.coli_Mapping1

Reference Accession ▲ Number	Start Position in Ref	End Position in Ref	Reference Bases	Variation Bases	Total Percent	Total Depth
gi|49175990|ref|…	215134	215136	AAA	–	33.3	9
gi|49175990|ref|…	547694	547694	A	G	100.0	16
gi|49175990|ref|…	547836	547836	–	G	92.3	13
gi|49175990|ref|…	2171387	2171387	–	CC	100.0	8
gi|49175990|ref|…	3558478	3558478	G	–	100.0	24
gi|49175990|ref|…	3957957	3957957	C	T	100.0	23
gi|49175990|ref|…	4294405	4294405	–	GC	100.0	16

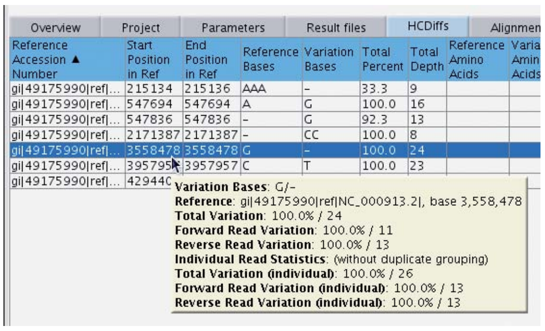
Reference Amino Acids	Variation Amino Acids	Coding Frame	Region Name	Known SNP Info

Percent Forward With Variation	Percent Reverse With Variation	Num Forward With Variation	Num Reverse With Variation	Total Num Forward Reads	Total Num Reverse Reads
50.0	20.0	2	1	4	5
100.0	100.0	7	9	7	9
100.0	90.0	3	9	3	10
100.0	100.0	5	3	5	3
100.0	100.0	11	13	11	13
100.0	100.0	14	9	14	9
100.0	100.0	3	13	3	13

Figure 6–18: The HCDiffs tab of the GS Reference Mapper application. This tab contains three main categories of information: (A) General information on the high confidence differences that were found in the computed data, between the reads and the reference sequences. (B) Annotations on the listed differences (from databases), if available. (C) More detailed statistics on the prevalence of the listed differences.

The HCDiffs tab has the following features:

- **Sorting**:
  - All columns are sortable by clicking on the column header. A small triangle appears to indicate the direction of the sort. Clicking on a column header of a column that is already sorted will reverse the order of the sort.
  - If multiple rows have the same value for the sorted column, these rows stay in the order in which they were before the sort, relative to one another. This allows for "nested sorting". For example, if you
    - sorting first based on "Start Position in Ref",
    - then sorting based on "Reference Accession Number",
    - results in a table where the high confidence differences are grouped by reference sequence, and the ones that belong to the same reference are displayed in order of their position on the reference sequence.
- **Summary tooltip:**
  - Pausing the mouse cursor over any row provides a tooltip that summarizes the basic statistics of the high confidence difference described in that row (Figure 6–19). In particular, the bases involved in the variation coupled with the total variation percentage, the depth of sequencing, and the separate statistics for both forward and reverse reads are presented.



Reference Accession ▲ Number	Start Position in Ref	End Position in Ref	Reference Bases	Variation Bases	Total Percent	Total Depth	Reference Amino Acids	Varia Amin Acids
gi|49175990|ref|...	215134	215136	AAA	–	33.3	9		
gi|49175990|ref|...	547694	547694	A	G	100.0	16		
gi|49175990|ref|...	547836	547836	–	G	92.3	13		
gi|49175990|ref|...	2171387	2171387	–	CC	100.0	8		
gi|49175990|ref|...	3558478	3558478	G	–	100.0	24		
gi|49175990|ref|...	395795	3957957	C	T	100.0	23		
gi|49175990|ref|...	42944C							

Variation Bases: G/–
Reference: gi|49175990|ref|NC_000913.2|, base 3,558,478
Total Variation: 100.0% / 24
Forward Read Variation: 100.0% / 11
Reverse Read Variation: 100.0% / 13
Individual Read Statistics: (without duplicate grouping)
Total Variation (individual): 100.0% / 26
Forward Read Variation (individual): 100.0% / 13
Reverse Read Variation (individual): 100.0% / 13

Figure 6–19: Summary tooltip for a high confidence difference listed on the HCDiffs tab

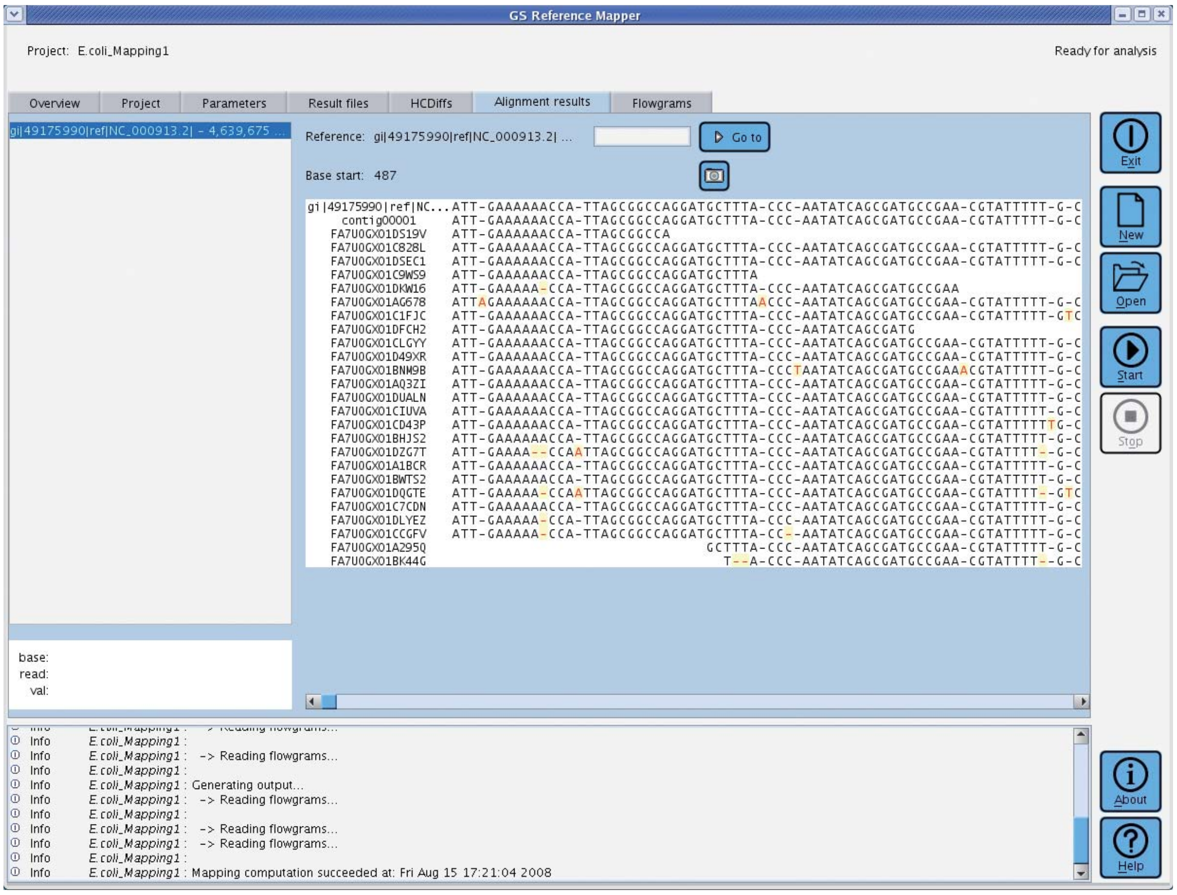
- Links to read alignment data:
  - Right-clicking in the row for a particular HCDiff variation opens a contextual menu with a single item: "Show Alignment". Selecting this loads the Alignment results tab with the alignment that supports the HCDiff in question (section 6.2.9.4). The first base of the variation as the leftmost base in the view.
  - You can then scroll the alignment to examine the context of the variation (to its left). Note in particular that, due to the manner in which the alignments coordinates are displayed, certain variations may be associated with the base that follows them in the alignment. In this case, viewing the full HCDiff variation may require backing up one base (click the left-arrow of the horizontal scrollbar) in the Alignment view.

Note on the handling of duplicate reads in computing HCDiffs:
Duplicate reads of a single DNA input are a known potential artifact of the emPCR amplification process which may occur, for example, when two or more beads are amplified in a single emulsion microreactor. In an attempt to compute more accurate variation percentages, the GS Reference Mapper, by default, groups individual reads into groups of duplicates (reads that start at the same base of the reference sequence). Groups of duplicate reads count as only one read in the computation of the HCDiffs. (However, the corresponding alignments, on the Alignment results tab, display all the reads whether or not they were considered duplicates.) When grouping occurred, the HCDiff summary tooltip provides the statistics for both grouped and for individual reads (see Figure 6–19).

On the command line, the "-ud" option (see sections 6.3.1 or 6.3.2.5) instructs the GS Reference Mapper to treat all reads individually and to not perform this grouping operation. This option is not directly available through the GUI, but if a project executed from the command line were computed in that fashion and then displayed in the GUI, the values in the HCDiffs table would be based on the un-grouped, individual reads instead of the default grouped duplicate reads.

6.2.9.4 Viewing the Alignment Data: the Alignment Results Tab

Click on the Alignment results tab to view the alignment data from the mapping computation (Figure 6–20), or access it using the "View Alignment" right-click menu for an HCDiff of interest, from the HCDiffs tab (section 6.2.9.3). The left panel of the Alignment results tab contains a list of the reference sequences used in the mapping project. To view the alignments to a reference, click on that reference on the list, and the multi-alignment of all of the reads that aligned to that reference will be displayed on the right panel. By default, the first Reference on the list will be automatically selected.



Project: E.coli_Mapping1

Ready for analysis

gi|49175990|ref|NC_000913.2| - 4,639,675 ...

Reference: gi|49175990|ref|NC_000913.2| ...

Base start: 487

```
gi|49175990|ref|NC...ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
       contig00001   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01DS19V   ATT-GAAAAAACCA-TTAGCGGCCA
    FA7U0GX01C828L   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01DSEC1   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01C9WS9   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA
    FA7U0GX01DKW16   ATT-GAAAAA-CCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA
    FA7U0GX01AG678   ATTAGAAAAAACCA-TTAGCGGCCAGGATGCTTTAACCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01C1FJC   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-GTC
    FA7U0GX01DFCH2   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATG
    FA7U0GX01CLGYY   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01D49XR   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01BNM9B   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCCTAATATCAGCGATGCCGAAACGTATTTTT-G-C
    FA7U0GX01AQ3ZI   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01DUALN   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01CIUVA   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01CD43P   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTTTG-C
    FA7U0GX01BHJS2   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01DZG7T   ATT-GAAAA--CCAATTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTT--G-C
    FA7U0GX01A1BCR   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01BWTS2   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01DQGTE   ATT-GAAAAA-CCAATTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTT--GTC
    FA7U0GX01C7CDN   ATT-GAAAAAACCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01DLYEZ   ATT-GAAAAA-CCA-TTAGCGGCCAGGATGCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01CCGFV   ATT-GAAAAA-CCA-TTAGCGGCCAGGATGCTTTA-CC--AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01A295Q                                GCTTTA-CCC-AATATCAGCGATGCCGAA-CGTATTTTT-G-C
    FA7U0GX01BK44G                                  T--A-CCC-AATATCAGCGATGCCGAA-CGTATTTT--G-C
```

base:
read:
val:

```
ⓘ Info    E.coli_Mapping1 :
ⓘ Info    E.coli_Mapping1 :   -> Reading flowgrams...
ⓘ Info    E.coli_Mapping1 :
ⓘ Info    E.coli_Mapping1 : Generating output...
ⓘ Info    E.coli_Mapping1 :   -> Reading flowgrams...
ⓘ Info    E.coli_Mapping1 :
ⓘ Info    E.coli_Mapping1 :   -> Reading flowgrams...
ⓘ Info    E.coli_Mapping1 :   -> Reading flowgrams...
ⓘ Info    E.coli_Mapping1 :
ⓘ Info    E.coli_Mapping1 : Mapping computation succeeded at: Fri Aug 15 17:21:04 2008
```

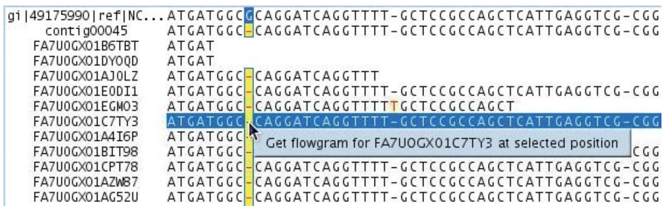
Figure 6–20: The read alignments of a selected reference

The multi-alignment panel has the following features:

- The larger area, on white background, contains the multi-alignment proper:
  - The selected reference sequence is shown on top of the panel, gapped for insertions in the aligned reads and contigs
  - All the contigs and reads (with contigs, if present, appearing before reads) that aligned to this reference sequence appear underneath, gapped and showing bases that diverge from the reference as red dashes on yellow background
  - The names of the reference sequence and of all the contigs and reads in the multi-alignment appear in a column, on the left side of the multi-alignment area
  - Scroll bars allow you to navigate the alignment manually:
    - Clicking on the arrowheads on either side of a scroll bar scrolls the alignment by one base (horizontal) or one read (vertical) in the direction of the arrowhead
    - Clicking on the light-colored area on either side of the horizontal scroll bar "handle" scrolls the alignment by 40 bases is the corresponding direction
    - Clicking and dragging the "handle" of a scroll bar allows you to move larger distances rapidly
  - Right clicking on a GS read will produce a "Get flowgram for … at selected position" menu item which, if selected, will activate the Flowgrams tab and display the flowgram data for the read; indicating the flow corresponding to the base on which the user clicked to activate the option. This capability is not active for FASTA reads or for the reference/contig sequences themselves, for which no flowgrams are available.
- Above that area are some informational and navigational controls:
  - **Reference:** The name of the selected reference sequence and its length
  - **Go To**: A data entry field and a **Go To** button allow you to navigate directly to a position of interest: enter a number not larger than the length of the selected reference sequence in the data entry field, and click the **Go To** button.
  - **Base start**: indicates the first base of the selected reference sequence that is currently displayed in the multi-alignment (*i.e.* per scrolling)
  - **Camera icon**: saves an image of the part of the multi-alignment table currently visible on screen, to a file in .png format.
- The Alignment Results tab also features a "Mouse Tracker" area, in the left panel, below the list of reference sequences. If you pause the mouse pointer over a base in the multi-alignment, the following information is provided about that base:
  - **Base**: the position of that base relative to the selected reference sequence; gaps in the reference sequence are displayed as the following base of the reference
  - **Read**: the name of the read or contig to which this base belongs
  - **Val**: the "value" of the base under the mouse pointer, in that read, *i.e.* A, G, T, or C.

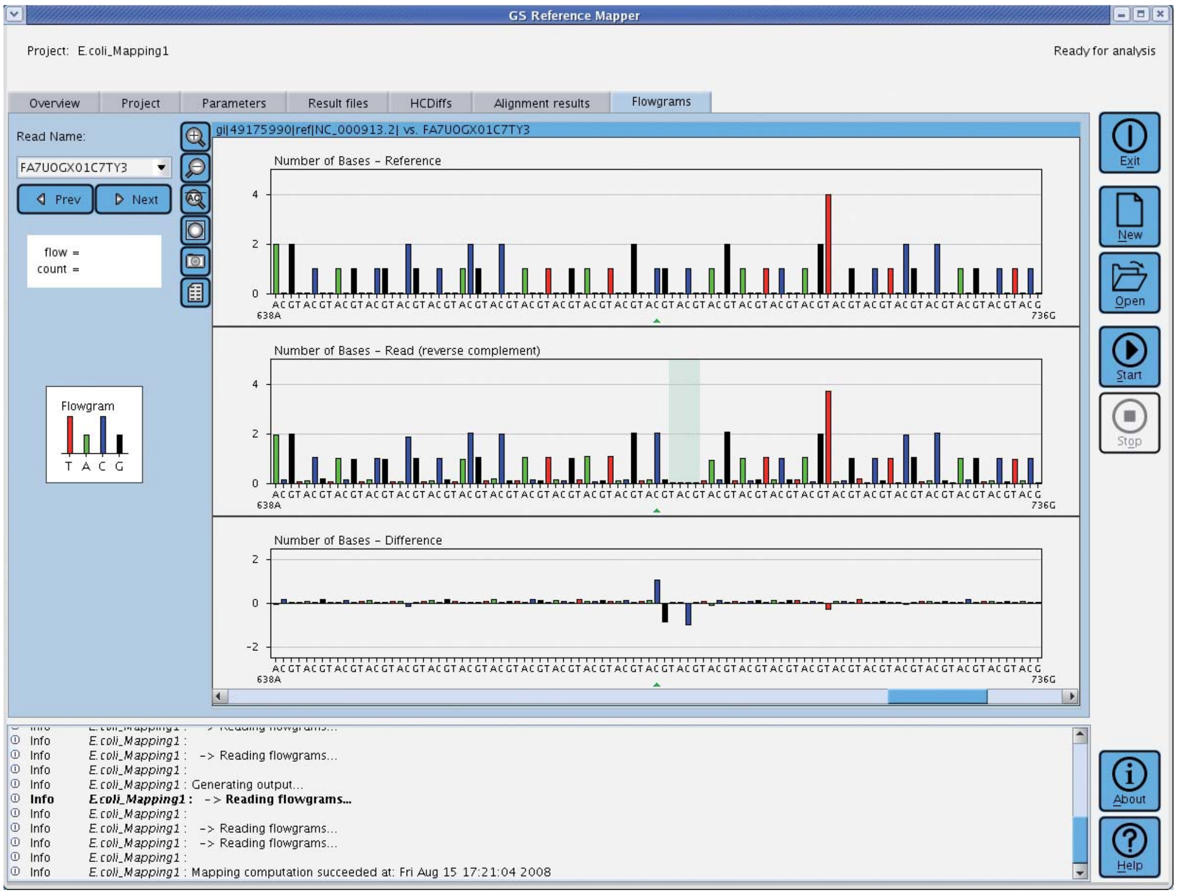
6.2.9.5 Viewing the Flowgram Data: the Flowgrams Tab

To view the flowgram from any read that aligned to a reference sequence, first select that reference sequence in the Alignment results tab, and then right-click on any base in the read of interest to open a contextual menu that contains a single item (starting with "Get flowgram:"; Figure 6–21). Selecting that item will display the flowgram for this read, in the Flowgrams tab (Figure 6–22). The term "flowgram" is defined in the Glossary (see the *Genome Sequencer FLX Operator's Manual*), and a full description of flowgram views is given in the section on the Amplicon Variant Analysis (section 9.8).



```
gi|49175990|ref|NC...ATGATGGCGCAGGATCAGGTTTT-GCTCCGCCAGCTCATTGAGGTCG-CGG
   contig00045      ATGATGGC-CAGGATCAGGTTTT-GCTCCGCCAGCTCATTGAGGTCG-CGG
  FA7U0GX01B6TBT    ATGAT
  FA7U0GX01DY0QD    ATGAT
  FA7U0GX01AJ0LZ    ATGATGGC-CAGGATCAGGTTT
  FA7U0GX01E0DI1    ATGATGGC-CAGGATCAGGTTTT-GCTCCGCCAGCTCATTGAGGTCG-CGG
  FA7U0GX01EGM03    ATGATGGC-CAGGATCAGGTTTTTGCTCCGCCAGCT
  FA7U0GX01C7TY3    ATGATGGC-CAGGATCAGGTTTT-GCTCCGCCAGCTCATTGAGGTCG-CGG
  FA7U0GX01A4I6P    ATGATGGC-
  FA7U0GX01BIT98    ATGATGGC-                                       CGG
  FA7U0GX01CPT78    ATGATGGC-CAGGATCAGGTTTT-GCTCCGCCAGCTCATTGAGGTCG-CGG
  FA7U0GX01AZW87    ATGATGGC-CAGGATCAGGTTTT-GCTCCGCCAGCTCATTGAGGTCG-CGG
  FA7U0GX01AG52U    ATGATGGC-CAGGATCAGGTTTT-GCTCCGCCAGCTCATTGAGGTCG-CGG
```

Figure 6–21: Selecting a read for flowgram display



```
Info    E.coli_Mapping1 :
Info    E.coli_Mapping1 :   -> Reading flowgrams...
Info    E.coli_Mapping1 :
Info    E.coli_Mapping1 : Generating output...
Info    E.coli_Mapping1 :   -> Reading flowgrams...
Info    E.coli_Mapping1 :
Info    E.coli_Mapping1 :   -> Reading flowgrams...
Info    E.coli_Mapping1 :   -> Reading flowgrams...
Info    E.coli_Mapping1 :
Info    E.coli_Mapping1 : Mapping computation succeeded at: Fri Aug 15 17:21:04 2008
```

Figure 6–22: The Flowgrams tab of the GS Reference Mapper application

The flowgram view as displayed on the Flowgrams tab of the GS Reference Mapper is most similar to the tri-flowgram view described in detail in sectionsection 9.8.2 (for the AVA) in that it shows 3 plots:

- The top plot is an idealized flowgram for the portion of the reference sequence that corresponds to the mapped region of the selected read, including the insertions of any flow cycle shifts that may be needed to maintain the alignment to the selected read's actual flowgram.
- The center plot is the aligned flowgram for the selected read, including the insertions of any flow cycle shifts that may be needed to maintain the alignment to the reference sequence's idealized flowgram.

- The lower plot is the difference between the upper two flowgrams.

A small green triangle will indicate the flow corresponding to the base of the read on which you right-clicked when launching the Flowgrams tab view.

The Flowgram tab provides various navigation options and other features:

- The top-left corner of the tab provides two controls that allow the user to change the display to a different read:
  - A drop down menu allows you to select from a list of all the reads aligned to the currently selected Reference (Figure 6–23), and which span the base position of the alignment that was used when initially launching the Flowgrams tab. That base position may be quickly updated by navigating back to the Alignment results tab and selecting a new alignment column by simply left-clicking in that column. Then switch back to the Flowgrams tab (by clicking on the tab). The drop down menu will be updated to include only those reads that have flowgrams and also which intersect with the new column of interest. The read previously displayed in the flowgram view will remain displayed (unless the new column of interest is not spanned by that read), but the green triangle will be updated to point to the flow corresponding to the base in the new column of interest. If the previously displayed read does not span the new column of interest, then the display will automatically go to the first read, topmost in the displayed alignment, that does span the column.
  - A pair of buttons ("Prev" and "Next"; see Figure 6–22, above) allows you to change the display to the previous or the next contig or read of the reference sequence's multiple alignment (in the order shown in the Alignment results tab).
  - You can also return to the Alignment results tab and select a different read, from the alignment to the same or to any other reference sequence.
- Various zooming, saving, and mousing functions are available on the Flowgrams tab, some via a column of buttons located to the upper-left of the graph area (Figure 6–23). These buttons and functions operate in ways very similar to their counterparts on the AVA (see section 9.8.2 for a full description), and include:
  - Various zooming buttons to zoom in, zoom out, zoom to labels, and zoom to fit.
  - A Snapshot and a Spreadsheet button, to save the data in image (.png) or spreadsheet (.tsv) form, respectively.
  - A mouse tracker area that shows the values for the nucleotide flow under the mouse pointer, when you pause it over the graph area:
    - position of the nucleotide on the reference sequence,
    - name of the nucleotide, and
    - the "y" value on the plot
      - number of bases elongated during that flow (idealized per the reference sequence or observed in the read), or
      - the value of the "difference", where bases in the read that are absent in the reference are positive differences.
  - There is also a free hand zoom function that allows you to zoom in to any area of a graph by drawing a box with the left mouse button. This function has the following unique features (not available in the corresponding GS Run Browser freehand zoom in functions) to facilitate the comparison between the three plots:
    - zooming on the reference or on the read plots will apply to the Y axis of both those plots, and the X-axis of all three plots (including the difference plot) (Figure 6–24).
    - zooming on the difference plot zooms its Y axis alone, but zooms the X axis of all three plots (not shown).

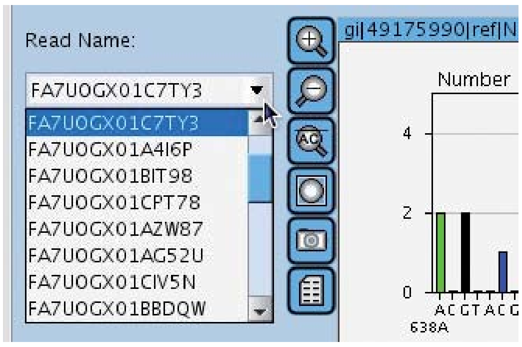
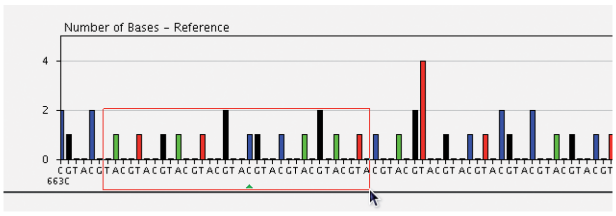
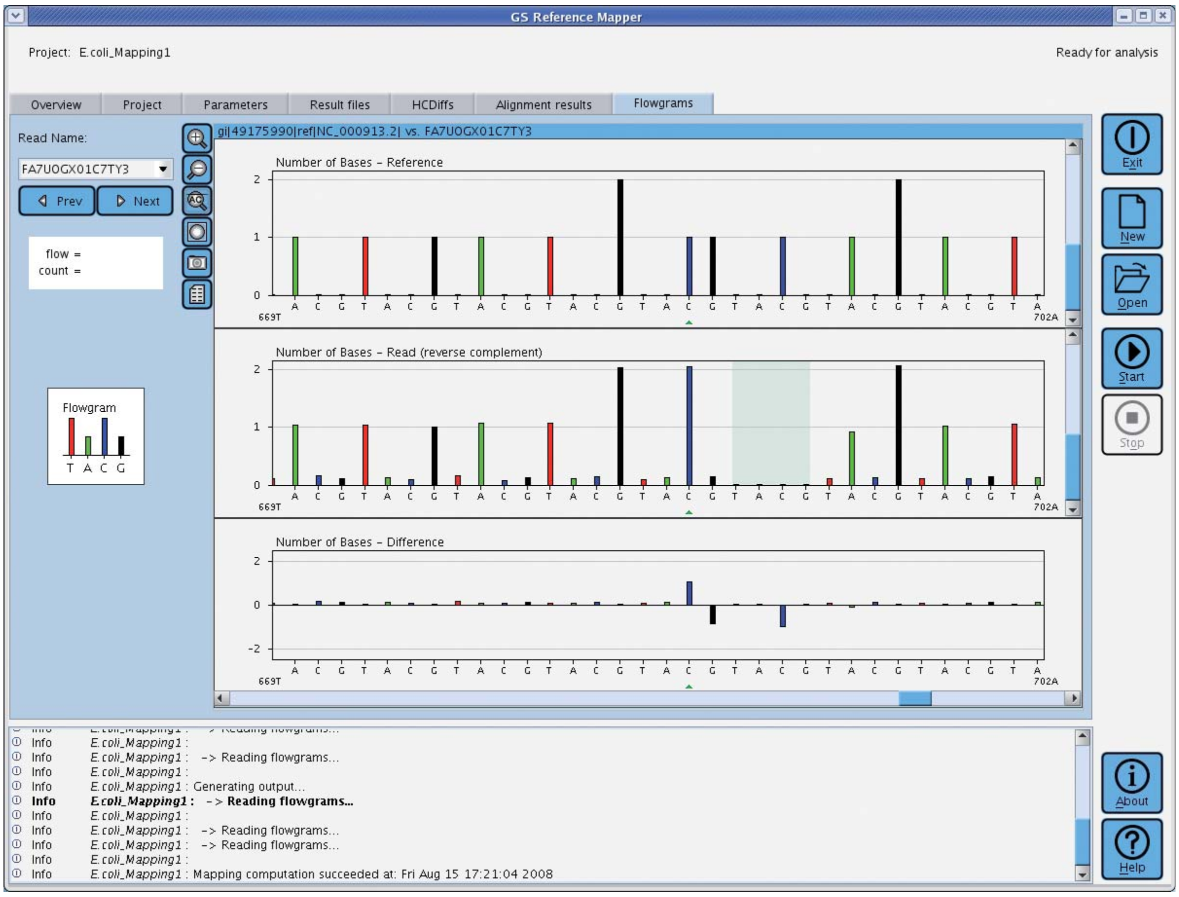


Figure 6–23: Selecting a read from the drop down menu in the Flowgrams tab

A



B



```
ⓘ Info      E.coli_Mapping1 :
ⓘ Info      E.coli_Mapping1 :   -> Reading flowgrams...
ⓘ Info      E.coli_Mapping1 :
ⓘ Info      E.coli_Mapping1 : Generating output...
ⓘ Info      E.coli_Mapping1 :   -> Reading flowgrams...
ⓘ Info      E.coli_Mapping1 :
ⓘ Info      E.coli_Mapping1 :   -> Reading flowgrams...
ⓘ Info      E.coli_Mapping1 :   -> Reading flowgrams...
ⓘ Info      E.coli_Mapping1 :
ⓘ Info      E.coli_Mapping1 : Mapping computation succeeded at: Fri Aug 15 17:21:04 2008
```

Figure 6–24: Using the mouse to zoom into a sub-region of a flowgram

6.3 Using the GS Reference Mapper Via the Command Line Interface

Mapping can be performed either using the Graphical User Interface described above, or using the Linux command line, described in this section.

6.3.1 One-Step Mapping: the runMapping Command

If all the reads to be mapped to the reference sequence are available at once, the GS Reference Mapper application can process them with a single command, which has the following command line structure:

```
runMapping [options] refdesc [MIDList@]filedesc…
```

…where:

- "[options]" are zero or more of the command line options (listed below),
- the "refdesc" is one of the following:
  - `reffasta`
    - a FASTA file containing the reference sequence(s),
  - `refdirname`
    - a name or path for a directory containing the reference sequence (all FASTA files in the directory will be used as the reference),
  - `refgenome`
    - a GoldenPath genome name (see the fifth Note, below),
  - `–ref (reffasta | refdirname)… -read`
    - the string "`-ref`", followed by multiple FASTA file names or directories, followed by the string "`-read`" (to mark the beginning of the read data files)
- each "filedesc" is one of the following:
  - `sfffilename or`
  - `[regionlist:]datadir or`
  - `readfastafile`
- and each "MIDList" is a multiplexing information string used to filter the set of file reads to be used in the mapping (see the third Note below for the format of the MIDList information). If MIDs were used in the generation of the datafile, then an MIDList string must be specified in order for the GS Reference Mapper to properly handle the file's reads.

External files:

- GS Read Data files (SFF files) are not actually copied to the project directory. Rather, a symbolic link to the file is created and placed in the project's sff directory. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies whether the files are added via the GUI or the command line.
- FASTA files are not actually copied to the project directory. Rather, the file path is simply recorded in the project setup. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies to all instances of FASTA files that can be added to a Mapping project, whether done via the GUI or the command line:
  - FASTA read files, including Sanger reads
  - Trimming database files
  - Screening database files
  - Reference sequence FASTA files

Unlike with the GS Read Data files, no symbolic link to the file is created for FASTA files.

Command	Description
runMapping	Complete set of mapping algorithms, whereby the read flowgrams of one or more sequencing Runs are mapped to the reference sequence(s), resulting in a final consensus sequence of the sample DNA library, and a list of differences with the reference sequence(s) (mutations). (See the third Note, below.)

There are a large number of command line options to this command, only a subset of which are commonly used or important. The following are the most common command line arguments:

Option	Description
[-nrm]	An option to allow the mapping results to be used for further project-based mapping, by "not removing" the project files and folders at the end of the mapping processing (as described in section 6.3.2; see also the first Note below)
[-o dir]	An option to specify the directory where all the output files should be written. If this option is omitted, the program will generate the 'P_' directory in the current working directory at the time the command is launched.
[-no]	With the "no output" option, the program will perform all the mapping of the reads to the reference, but will not generate consensus/contig output. This is useful to avoid wasting computing time, *e.g.* when the user knows that more Runs will be added to the project. (However, the project and alignment metrics are still output.)
[-noace]	With this option, the ACE file is not generated.
[-ace]	With this option, the command generates a single ACE file containing the mapping of all the reads and contigs to the reference sequence(s).
[-acedir]	With this option, the command generates an "ace" sub-directory containing a separate ACE file for each sequence in the reference(s) (see Figure 1–5).
[-consed]	With this option, the command generates a "consed" sub-directory containing all the directories and files needed to run the consed application on the results of the mapping (see Figure 1–5).
[-pair]	This option outputs the pairwise overlaps used in the mapping, in simple text format. By default, these alignments are not output.
[-pairt]	This option outputs the pairwise overlaps used in the mapping, in tab-delimited format. By default, these alignments are not output.
[-p (sfffilename or [regionlist:]datadir)]	An option to specify that the SFF file or Data Processing folder whose path immediately follows contains the results of a Paired End sequencing Run, to be used to order and orient the contigs of the Shotgun sequencing Run(s) (if present). Multiple Paired End Runs may be included in the Analysis. The –p option forces the software to consider a data set as Paired End; all Runs whose reference is not preceded by " p" will be determined as being Paired End or Shotgun sequencing Runs, by virtue of the presence or absence of the Paired End linker sequence in the read data they contain: if at least 25% of the first 500 reads in the file (or 25% of all the reads if there are fewer than 500 reads in the file) contain the linker, the file is recognized as a Paired End file. The details for Shotgun Run datasets (below) also apply to Paired End Run datasets.
[--help]	Displays a usage line and short description of the command.
[--version]	Displays the current software version of the GS Reference Mapper.

Option	Description
MIDList@] (sfffile or [*regionlist*:]datadir or readfastafile)	Path to an SFF file or to a Data Processing folder containing the results of a Shotgun sequencing Run, created by the GS Run Processor application (either on a Genome Sequencer FLX Instrument or on a DataRig or cluster), and containing the SFF files for the regions of the sequencing Run (in the "sff" sub-folder; see Figure 1–2). There can be any number of SFF files and/or Data Processing folders given as argument (and thus included in the mapping), an optional list of regions may be prepended to each data directory argument (see the second Note below), and an optional multiplexing information string can be prepended to each file/data-directory argument (see the third Note below). [If SFF files are not present in a Data Processing folder (*e.g.* for a Run whose data has been processed with a version of the Genome Sequencer System software anterior to 1.0.52), the signal processing step of the GS Run Processor application must be rerun on the Data Processing folder.] Alternatively, path to a FASTA file containing Sanger reads to be included in the mapping. These reads (or contigs) must be shorter than 2000 bases (longer sequences are ignored) and longer than 50 bases (shorter sequences are ignored). See section 13.2 for input FASTA file format requirements.
refdesc	Path to a single FASTA file containing the reference sequence(s) to which the input reads will be mapped, a single directory containing FASTA files for the reference, a single GoldenPath genome name, or multiple FASTA files or directories surrounded by "-ref" and "-read" strings. (See the fourth Note, below.)

Input read size constraints: The reads (or contigs) input for mapping computation must be shorter than 2000 bases per read (longer sequences are ignored) and longer than 50 bases (shorter sequences are ignored).

A second set of arguments are less commonly used, but are useful to configure the output generated by the command or to adjust the execution of the computation:

Option	Description
[-a #]	This option sets the minimum length for a contig to appear in the 454AllContigs.fna file. The default is 100 bases.
[-ar]	An option telling the GS Reference Mapper to output the full “raw” read sequence when generating an ACE file or consed folder.
[-at]	An option telling the GS Reference Mapper to output only the “trimmed” sequences for each read when generating an ACE file or consed folder.
[-e #]	An option telling the GS Reference Mapper that the “expected depth” of the data is at a certain level. The GS Reference Mapper’s variation detection has been optimized for datasets in the 10–50× oversampling size, and this option helps the variation detection with datasets that have a higher oversampling level. A value of 0 resets the GS Reference Mapper computation to use its default algorithms.
[-l #]	This option sets the minimum length for a contig to appear in the 454LargeContigs.fna file. The default is 500 bases.
[-mcf *filename*]	This option specifies a different MID configuration file to be used by the GS Reference Mapper for decoding the multiplex information appearing on the command line.
[-nobig]	This option turns off the generation of the “big” output files, namely the ACE/consed files, the 454PairAlign.txt file and the 454AlignmentInfo.tsv file.
[-notrim]	This option turns off the default quality and primer trimming that the GS Reference Mapper performs on all input reads (454 Sequencing reads and Sanger reads).
[-v fastafile] or [-vt fastafile]	This option specifies a “vector trimming database”, or FASTA file of sequences to be used to trim the ends of input reads (for cloning vectors, primers, adapters or other end sequences).
[-vs fastafile]	This option specifies a “vector screening database”, or FASTA file of sequences to be used to screen the input reads for contaminants. Reads that completely align against the screening database are trimmed completely (so that it is not used in the computation); if only part of a read aligns with the screening database, no action is taken.

A third set of arguments support region-based resequencing experiments, such as the NimbleGen Sequence Capture process, as well as support the reporting of genome annotation information in the variation detection output:

Option	Description
[-n or –nimblegen]	This option tells the GS Reference Mapper that the reads come from a NimbleGen Sequence Capture experiment. The reads will be automatically trimmed for the Sequence Capture primer sequence.
[-reg (accno:#-# or regstringfile or NimblegenDARFile)]	This option tells the GS Reference Mapper to only output specific regions of the reference, and the reads aligned to those regions. The reads are mapped against the complete reference, but the output files will only report on the specified regions. The option argument can be either: • a "accno:#-#" string, *e.g.* "chr9:10549-30594" specifies bases 10549 to 30594 of the chr9 reference sequence; • the path to a "regstringfile" which is a file containing lines of "accno:#-#" strings (one per line); • the path to a GFF DAR file created during a NimbleGen Sequence Capture experiment, describing the "primary_target_region" regions of the experiment.
[-annot *filename*]	An optional file describing the gene/coding-region annotations for the reference sequences, so that the variation detection can report gene names and protein translations of any identified variations in gene regions (on the HCDiffs tab). The format of this file must match that of the GoldenPath "refGene.txt" file. (See the sixth Note below for the case where the GS Reference Mapper automatically reads a gene annotation file.)
[-noannot]	Tells the GS Reference Mapper not to use an annotation file during the mapping. This option must be used to disable the automatic reading of annotation files.
[-snp *filename*]	An optional file describing the known SNP information for the reference sequences, so that the variation detection can link identified variations to the known SNP information (on the HCDiffs tab). The format of this file must match that of the GoldenPath snp128.txt file, and the contents must be for the same reference given to the GS Reference Mapper. (See the sixth Note below for the case where the GS Reference Mapper automatically reads a known SNP file.)
[-nosnp]	Tells the GS Reference Mapper not to use a known SNP file during the mapping. This option must be used to disable the automatic reading of known SNP files.

A final set of options exist, that can be used to adjust the mapping algorithm parameters (equivalent to the Parameter tab settings in the GUI), or minor options that are more rarely used. See section 6.2.4 for a more complete description of the Parameter tab settings. The mapping computation has been optimized to give its best results using the default parameters, but these are provided for users who wish to experiment with adjusting parameters for their data sets.

Option	Description
[-ais]	This option sets the “Alignment identity score” parameter.
[-ads]	This option sets the “Alignment difference score” parameter.
[-fd]	This option tells the GS Reference Mapper to output “full description” lines for each of the detected variations (high confidence differences between the reads in the data set and the reference sequences). By default, only some of the columns on the one-line descriptions are output in the 454AllDiffs.txt and 454HCDiffs.txt files. See Section 6.4.2.6 for the details of the one-line descriptions.
[-hll]	This option sets the “Hit location limit” parameter.
[-hsl]	This option sets the “Hit-per-seed limit” parameter.
[-m]	An option telling the GS Reference Mapper to keep all sequence data “in memory” throughout the computation. This option can speed up the execution of the GS Reference Mapper, but requires the use of more computer memory.
[-mi]	This option sets the “Minimum overlap identity” parameter.
[-ml]	This option sets the “Minimum overlap length” parameter.
[-nor]	This option turns off the automatic “rescore” function that generates new quality scores for each SFF read using the new quality score algorithms.
[-sc]	This option sets the “Seed count” parameter.
[-rst]	This option sets the “Repeat score threshold” parameter.
[-sl]	This option sets the “Seed length” parameter.
[-ss]	This option sets the “Seed step” parameter.
[-ud]	An option telling the GS Reference Mapper to treat each read as a separate read, and not group them into duplicates for consensus calling or variation detection.

- The runMapping command is actually a wrapper program around the newMapping and related commands used in incremental mapping (see section 6.3.2). After the last incremental mapping command is completed, runMapping then transforms the project files and folders into this one-step form (which more closely matches the output structure of older versions of the “Mapping” application). The actions taken are:
  - All the mapping output files are moved up from the mapping sub-directory into the main folder
  - All internal data files in the mapping sub-directory are deleted
  - The 454MappingProject.xml and 454Project.xml files are deleted
  - The mapping sub-directory is deleted

  If the –nrm option is given, runMapping does not perform this transformation, and the resulting folder can be used for further project-based mapping (the structure of the files/folders will match that of a mapping project).

- The path given for any of the data directories may be prepended with an optional list of regions, separated from the path by a colon. For example:

  1-3,4,6-8:D_2005_01_01_01_01_01_testuser_runImagePipe

  The list of regions must have the following format:

  - It must be a comma-separated list, ending with a colon.
  - Each element of the list can either be a single region number or a dash-separated range of region numbers.
  - Duplicate regions may be specified, and the regions may be specified in any order, but duplicates will be removed and the regions will be processed and reported in numerical order.
  - No spaces or other characters are allowed in the string.

  If the region list is not present for a data directory path, all regions of the Run for that directory will be used in the mapping. Also, any combination of explicit SFF files and data directory paths (with optional region lists) may be specified on the command line. For each data directory path given, the runMapping command will read the existing SFF files in the "sff" sub-directory of the data directory. If SFF files are not present in a data directory (*e.g.* for a Run whose data has been processed with a version of the Genome Sequencer System software anterior to 1.0.52), the signal processing step of the GS Run Processor application must be rerun on the Data Processing folder.

- The path for any filename or data directory name can be prepended with a multiplexing information string, separated from the filename/directory-string by an "@" sign. Multiplexing information strings should be used when multiple different samples are prepared using different initial "tag" sequences (such as those provided by the GS Multiplex Identifiers (MID) Kits), then mixed together for sequencing. These initial sequences are used to segregate the reads from each sample, by matching the initial bases of the reads to the differentiating tag sequences used in the preparation of the libraries. If a multiplex string is given, the GS Reference Mapper will automatically match the 5' bases of each read against the multiplex sequences, and will only use the reads that match the specified sequences (and reset the trim point of those reads so that the multiplex sequences are not used in the mapping itself).

  Three examples of multiplex information strings are the following:

  mid2@myreads.sff

  GSMIDs:mid1,mid4,mid8@/home/xxx/morereads.sff

  aattctc/1@1-3,4,6-8:D_2005_01_01_01_01_01_testuser_runImagePipe

- The first example gives "myreads.sff" as the input SFF file and tells the GS Reference Mapper to only use reads whose initial 5'-sequence matches the "mid2" MID (as listed in the MID configuration file). The second example tells the GS Reference Mapper to use the file "/home/xxx/morereads.sff" and filter that file, using only the reads starting with the mid1, mid4 and mid8 sequenced, as defined in the "GSMIDs" MID set. The third example tells the GS Reference Mapper to read regions 1, 2, 3, 4, 6, 7 and 8 of the "D_2005_..." sequencing Run, but only use the reads whose 5'-end matches the DNA sequence "AATTCTC" with 1 error or less.

  The format of the multiplex information string is the following:

  ```
  [setname:]mid[/#](,mid[/#]…)@
  ```

  …where

  - the "setname" is an optional name of an MID set found in the MID configuration file and can be given in uppercase or lowercase letters (the matching is case-insensitive);
  - each "mid" is either an MID name found in the configuration file or a DNA sequence; and
  - each "#" is an optional allowed number of errors.

- In other words, the format consists of:
  - An optional MID set name, followed by a colon
    - This name must occur in the MID configuration file
  - One or more MID names or DNA sequences, separated by commas. Each name/sequence can be followed by an optional slash and number of allowed errors.
    - The MID names must occur in the MID configuration file, and if the same MID name occurs in multiple MID sets in the file, then the MID set name must be given. (If the MID name is unique over all of the names in the file, then no MID set name is required.)
  - A "@" sign to end the multiplex information string.

  No spaces are allowed in the multiplex information string, and no colons, slashes or "@" signs are allowed in the MID set names or MID names.

  The off-instrument installation contains a default MID configuration file, found by default at /usr/local/rig/config/MIDConfig.parse. This file is read by the GS Reference Mapper and used to match MID set names and MID names with their multiplexing information. Users can edit this file to add their own MID sets (following the format and syntax described in the file), or can copy this file to create their own separate MID configuration file (and then use the "-mcf" option to specify that as the MID configuration file to be used). See Section 13.4.15 for the contents of the MIDConfig.parse file.

  - Contrary to the situation with project-based mapping (see the setRef command, section 6.3.2.2), you can use only one reference FASTA file, directory or GoldenPath genome name with the runMapping command, unless the –ref/-read options are including to separate the reference files from the read data files. All the files can still contain multiple reference sequences (see section 13.2).

  - The GS Reference Mapper is aware of the folder/file structure of the UCSC GoldenPath genome databases (most specifically the human genome and the other major eukaryotic genome databases). If one or more of those databases are downloaded onto the DataRig, and the user sets a GOLDENPATH environment variable to the root directory containing those databases, then the genome name can be given as the reference for runMapping (*i.e.*, the command "runMapping hg18 reads.sff" will map reads.sff against the GoldenPath hg18 human genome). The GS Reference Mapper will automatically read the chromosome FASTA files, the gene annotation file and the known SNP information file.

    Specifically, the Mapping application looks for the reference FASTA files in the "chromosomes" sub-directory, and files named "refGene.txt" and "snp???.txt" (*i.e.*, a file like "snp128.txt" that begins with "snp", has three digits, then ends with ".txt") in the "database" sub-directory. This matches the organization of the human genome database. Other GoldenPath databases (which don't match this organization) can be used by the GS Reference Mapper application if the files are modified to match this organization. See the UCSC GoldenPath "Downloads" page for information on how their genome databases can be downloaded.

  - In addition to automatically reading the gene/coding-region annotation and known SNP files when the reference is a GoldenPath genome, the GS Reference Mapper application automatically searches for specific annotation/SNP files for any reference. If the reference files are located in a directory named "chromosomes", the GS Reference Mapper will look for refGene.txt and snp???.txt files in a neighboring "database" directory (inside the same parent directory as "chromosomes"). The GS Reference Mapper also searches the same directory as the reference FASTA files for a refGene.txt and "snp???.txt". Finally, if the reference consists of a single FASTA file, then the software will look for files with ".refGene" and ".snp" suffixes (*e.g.*, if the reference file is "mygenome.fna", then files "mygenome.refGene" and "mygenome.snp" will be tried.

    To disable this automated reading of the annotation/SNP files, the –noannot and –nosnp options must be included as part of the command line options.

- Some of the runMapping command options listed above are mutually exclusive, for obvious functional reasons. Specifically:
  - Don't use -no or -nobig together with any of the following: -ace, -acedir, consed, -pair, or -pairt
  - Use no more than one of each of the following sets:
    - -noace, -ace, -acedir, or -consed
    - -ar or -at
    - -a # or -g
    - -annot or -noannot
    - -snp or -nosnp

6.3.2 Project-Based Mapping: the newMapping and Related Commands

This section describes the main commands of the GS Reference Mapper application, to be used when one or more Runs are to be mapped against a reference sequence or database, as part of a mapping project. These commands allow you to add Runs over time and incrementally align them to the reference; and to change the reference sequence and restart mapping without losing the dataset of Runs. With these commands, the execution of the mapping and generation of the results can be controlled by command line options and configuration parameters (paralleling the equivalent controls in the GUI application). Such incremental mapping can be useful when, for example, you want to see intermediate mapping results on existing sequencing Runs to determine if you need to carry out further Runs to reach a desired depth of coverage (or just to monitor the project).

Five commands are used in a mapping project: newMapping, setRef, addRun, removeRun and runProject. These are described in the subsections below.

6.3.2.1 The newMapping Command

The newMapping command is used to initiate a mapping project and set a mapping project folder to contain the project data (see Figure 1–7). Its command line structure is:

```
newMapping [dir]
```

Command	Description
newMapping	Creates a mapping "project" folder where the project-based mapping is performed and the results are stored.

Argument	Description
[dir]	An optional argument to set the directory where all the project input, status and output files should be written. If a directory is specified, that directory is used (created if it does not already exist) as the project directory. If the command is executed with no argument, it creates a new "P_..._runMapping" directory as the project directory, in the current working directory at the time the command is launched. Note: The directory name may not begin with a dash ('-').

6.3.2.2 The setRef Command

The setRef command is used to assign the reference sequence(s) to which the reads from the sequencing Run(s) included in the project will be mapped. Its command line structure is:

```
setRef [dir] refdesc…
```

…where each "refdesc" is one of the following:

- `reffasta`
  - a FASTA file containing the reference sequence(s),
- `refdirname`
  - a name or path for a directory containing the reference sequence (all FASTA files in the directory will be used in the reference),
- `refgenome`
  - a GoldenPath genome name (see the Fifth Note in the runMapping Section 6.3.1 above).

Command	Description
setRef	Sets or resets the reference sequence(s) for the mapping project to the sequence(s) in the fastafile(s) given on the command line.

Argument	Description
[dir]	An optional argument (must be first in the list of arguments) to identify the project folder for the command. If the command is executed with no "directory" argument, it verifies whether the current working directory is either a mapping project folder or the "mapping" sub-directory inside a mapping project folder, and uses that if so. This way, users can create a mapping project and then simply change directory ("cd") into it, and all the commands will work on the project they are in.
refdesc	Path to a FASTA file, directory or GoldenPath genome name containing the reference sequence(s) to which the reads from one or more sequencing Run(s) will be mapped. Multiple FASTA files can be used as argument, and each FASTA file can contain multiple sequences (see section 13.2). When multiple files are given, the command treats the reference as the union of all the sequences in all the FASTA files, (*i.e.* all the sequences are considered as one "genome" for the purposes of determining unique vs. repeat matches). (See Caution, below.)

External files: FASTA files are not actually copied to the project directory. Rather, the file path is simply recorded in the project setup. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies to all instances of FASTA files that can be added to a Mapping project, whether done via the GUI or the command line:

- FASTA read files, including Sanger reads
- Trimming database files
- Screening database files
- Reference sequence FASTA files

Unlike with the GS Read Data files, no symbolic link to the file is created for FASTA files.

- Contrary to the situation with one-step mapping (see the runMapping command, section 6.3.1), multiple FASTA files can be specified as arguments to the setRef command, each of which may contain multiple reference sequences (see section 13.2); mapping will be performed against the union of all the sequences contained therein.
- The setRef command can be run multiple times for a project (in any combination with the addRun, removeRun and runProject commands). When setRef is executed, it resets (not "adds") the reference sequence(s) used for the mapping. If previous mapping results exist for the project, they will be discarded at the next execution of the runProject command (as it resets the mapping results to remap the reads against the new reference).
- The commands setRef, addRun, removeRun, and runProject can be executed in any combination and in any order.

6.3.2.3 The addRun Command

The addRun command is used to add Read Data sets to existing mapping projects. Its command line structure is:

```
addRun [options] [MIDList@]filedesc…
```

…where:

- "[options]" are zero or more of the command line options (listed below),
- each "filedesc" is one of the following:
  - `sfffilename or`
  - `[regionlist:]datadir or`
  - `readfastafile`
- and each "MIDList" is a list of multiplexing information used to filter the set of file reads used in the assembly (see the Note in section 6.3.1 for the format of the MIDList information). If MIDs were used in the generation of the datafile, then an MIDList string must be specified in order for the GS Reference Mapper to properly handle the file's reads.

External files:

- GS Read Data files (SFF files) are not actually copied to the project directory. Rather, a symbolic link to the file is created and placed in the project's sff directory. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies whether the files are added via the GUI or the command line.
- FASTA files are not actually copied to the project directory. Rather, the file path is simply recorded in the project setup. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies to all instances of FASTA files that can be added to a Mapping project, whether done via the GUI or the command line:
  - FASTA read files, including Sanger reads
  - Trimming database files
  - Screening database files
  - Reference sequence FASTA files

Unlike with the GS Read Data files, no symbolic link to the file is created for FASTA files.

Command	Description
addRun	Adds the reads from one or more Read Data sets to an existing Mapping project.

Option	Description
[-p]	An option to specify that the Read Data given on the rest of the command line should be treated as the results of Paired End sequencing Runs. (Note: Unlike the runMapping command (see section 6.3.1), this -p option only needs to be specified once, and applies to all Read Data sets on this execution of addRun.) If the "-p" option is not given, each Read Data set given in argument will be determined as being Paired End or Shotgun sequencing Runs, by virtue of the presence or absence of the Paired End linker sequence in the read data they contain: if at least 25% of the first 500 reads in the file (or 25% of all the reads if there are fewer than 500 reads in the file) contain the linker, the file is recognized as a Paired End file. The details for Shotgun Run datasets (below) also apply to Paired End Run datasets.
[-mcf *filename*]	This option specifies an MID configuration file (other than the default file) to be used by the GS Reference Mapper for decoding the multiplex information appearing on the command line.

Argument	Description
[dir]	An optional argument (must be first in the list of arguments) to identify the project folder for the command. If the command is executed with no "directory" argument, it verifies whether the current working directory is either a mapping project folder or the "mapping" sub-folder inside a mapping project folder, and uses that if so. This way, users can create a mapping project and then simply change directory ("cd") into it, and all the commands will work on the project they are in.
[MIDList@] (sfffile or [*regionlist*:]datadir or readfastafile)	Path to an SFF file or to a Data Processing folder containing the results of a Shotgun sequencing Run or Paired End sequencing Run, created by the GS Run Processor application (either on a Genome Sequencer FLX Instrument or on a DataRig or cluster), and containing the SFF files for the regions of the sequencing Run (in the "sff" sub-folder; see Figure 1–2). There can be any number of SFF files and/or Data Processing folders given as argument (and thus included in the mapping), an optional list of regions may be prepended to each data directory argument (see Note in section 6.3.1), and a multiplexing information string can be prepended to each argument (see Note in section 6.3.1). [If SFF files are not present in a Data Processing folder (*e.g.* for a Run whose data has been processed with a version of the Genome Sequencer System software anterior to 1.0.52), the signal processing step of the GS Run Processor application must be rerun on the Data Processing folder.] Alternatively, path to a FASTA file containing Sanger reads to be included in the mapping. These reads (or contigs) must be shorter than 2000 bases (longer sequences are ignored) and longer than 50 bases (shorter sequences are ignored). See section 13.2 for input FASTA file format requirements.

- The addRun command can be executed multiple times for a mapping project (in any combination with the setRef, removeRun, and runProject commands). When addRun is executed, it adds (not "resets") the given Runs/regions or SFF files to the list of sequencing data used in the project. (It does not reset the list of sequence data). The reads used in the mapping (runProject, see section 6.3.2.5, below) are the union of the data from all executions of addRun for the project.
- The commands setRef, addRun, removeRun, and runProject can be executed in any combination and in any order, in a mapping project.

Input read size constraints: These reads (or contigs) input for assembly computation must be shorter than 2000 bases per read (longer sequences are ignored) and longer than 50 bases (shorter sequences are ignored).

6.3.2.4 The removeRun Command

The removeRun command is used to remove Read Data sets from existing mapping projects. Its command line structure is:

```
removeRun [dir] (sffname or readfastafilepath)…
```

Command	Description
removeRun	Removes the reads of one or more sequencing Runs from an existing Mapping project.

Argument	Description
[dir]	An optional argument (must be first in the list of arguments) to identify the project folder for the command. If the command is executed with no "directory" argument, it verifies whether the current working directory is either a mapping project folder or the "mapping" sub-folder inside a mapping project folder, and uses that if so. This way, users can create a mapping project and then simply change directory ("cd") into it, and all the commands will work on the project they are in.
sffname or readfastafilepath	The name of an SFF file currently included in the project, as given in the sff sub-directory of the project folder; or the path to an input read FASTA file, as given in the 454MappingProject.xml file for the project.

- This command is more conveniently carried out from the GUI application. When run from the command line, it requires the SFF file name(s) *given in the project sff sub-directory*, or the FASTA file name(s) *given in the project configuration file 454MappingProject.xml*, which may not match the original names of the corresponding files or may not be known to the user, especially if the software had to rename any of them to ensure name uniqueness (see section 6.4.2.16 for details on this).
- The execution of this command does not physically remove the file(s) from the project sff sub-directory or from the existing mapping alignments. The file(s) and the reads they contain are only marked for removal by the removeRun command, and are actually removed only the next time the project is computed (runProject command).
- The commands setRef, addRun, removeRun, and runProject can be executed in any combination and in any order, in a mapping project.

6.3.2.5 The runProject Command

The runProject command performs the actual mapping computation for a project and generates the results of the mapping. Its command line structure is:

```
runProject [options] [dir]
```

Command	Description
runProject	Executes the GS Reference Mapper application algorithms, mapping any new Runs present in the project directory to the reference sequence and generating the combined results for all the Runs currently included in the project.

There are a large number of command line options to this command, only a subset of which are commonly used or important. The following are the most common command line arguments:

Option	Description
[-r]	This option restarts the mapping computation anew, removing any incremental mapping results that may exist in the project directory.
[-no]	With the "no output" option, the program will perform all the mapping of the reads to the reference, but will not generate consensus/contig output. This is useful to avoid wasting computing time, *e.g.* when the user knows that more Runs will be added to the project. (However, the project and alignment metrics are still output.)
[-noace]	With this option, the ACE file is not generated.
[-ace]	With this option, the command generates a single ACE file containing the mapping of reads and contigs to the reference sequence(s).
[-acedir]	With this option, the command generates an "ace" sub-directory containing a separate ACE file for each sequence in the reference(s) (see Figure 1–7).
[-consed]	With this option, the command generates a "consed" sub-directory containing all the directories and files needed to run the consed application on the results of the mapping (see Figure 1–7).
[-p] or [-pair]	This option outputs the pairwise alignments found in the mapping, in simple text format. By default, these alignments are not output.
[-pt] or [-pairt]	This option outputs the pairwise alignments found in the mapping, in tab-delimited format. By default, these alignments are not output.
[--help]	Displays a usage line and short description of the command.
[--version]	Displays the current software version of the GS Reference Mapper.

Argument	Description
[dir]	An optional argument (follows any options) to identify the project folder for the command. If the command is executed with no "directory" argument, it verifies whether the current working directory is either a mapping project folder or the "mapping" sub-folder inside a mapping project folder, and uses that if so. This way, users can create a mapping project and then simply cd into it, and all the commands will work on the project they are in.

External files: FASTA files are not actually copied to the project directory. Rather, the file path is simply recorded in the project setup. Consequently, if the original file is moved or erased from the file system, the project will not operate correctly. This applies to all instances of FASTA files that can be added to a Mapping project, whether done via the GUI or the command line:

- FASTA read files, including Sanger reads
- Trimming database files
- Screening database files
- Reference sequence FASTA files

Unlike with the GS Read Data files, no symbolic link to the file is created for FASTA files.

A second set of arguments are less commonly used, but are useful to configure the output generated by the command or to adjust the execution of the computation:

Option	Description
[-a #]	This option sets the minimum length for a contig to appear in the 454AllContigs.fna file. The default is 100 bases.
[-ar]	An option telling the GS Reference Mapper to output the full "raw" read sequence when generating an ACE file or consed folder.
[-at]	An option telling the GS Reference Mapper to output only the "trimmed" sequences for each read when generating an ACE file or consed folder.
[-ad]	An option telling the GS Reference Mapper to use the default method of outputting reads when generating an ACE file or consed folder (which, for mapping, is to use the raw sequences for all reads except 454 Paired End reads, which are output in trimmed form).
[-e #]	An option telling the GS Reference Mapper that the "expected depth" of the data is at a certain level. TheGS Reference Mapper's variation detection has been optimized for datasets in the 10–50× oversampling size, and this option helps the variation detection with datasets that have a higher oversampling level. A value of 0 resets the GS Reference Mapper computation to use its default algorithms.
[-l #]	This option sets the minimum length for a contig to appear in the 454LargeContigs.fna file. The default is 500 bases.
[-nobig]	This option turns off the generation of the "big" output files, namely the ACE/consed files, the 454PairAlign.txt file and the 454AlignmentInfo.tsv file.
[-notrim]	This option turns off the default quality and primer trimming that the GS Reference Mapper performs on all input reads (454 Sequencing reads and Sanger reads).
[-trim]	This option turns on the default quality and primer trimming that the GS Reference Mapper performs on all input reads (454 Sequencing reads and Sanger reads). This is the default for mapping.
[-v fastafile] or [-vt fastafile]	This option specifies a "vector trimming database", or FASTA file of sequences to be used to trim the ends of input reads (for cloning vectors, primers, adapters or other end sequences).
[-nov] or [-novt]	This option specifies no trimming database should be used, removing a vector database that had been specified in an earlier execution of runProject.
[-vs *fastafile*]	This option specifies a "vector screening database", or FASTA file of sequences to be used to screen the input reads for contaminants. Reads that completely align against the screening database are trimmed completely (so that they are not used in the computation); if only part of a read aligns with the screening database, no action is taken.
[-novs]	This option specifies no screening database should be used, removing a screening database that had been specified in an earlier execution of runProject.

A third set of arguments support region-based resequencing experiments, such as the NimbleGen Sequence Capture process, as well as support the reporting of genome annotation information in the variation detection output:

Option	**Description**
[-n or -nimblegen]	This option tells the GS Reference Mapper that the reads come from a NimbleGen Sequence Capture experiment. The reads will be automatically trimmed for the Sequence Capture primer sequence.
[-non or -nonimblegen]	This option turns off the NimbleGen mapping mode.
[-reg (accno:#-# or regstringfile or NimblegenDARFile)]	This option tells the GS Reference Mapper to only output specific regions of the reference, and the reads aligned to those regions. The reads are mapped against the complete reference, but the output files will only report on the specified regions. The option argument can be either: ▶ a “accno:#-#” string, *e.g.* “chr9:10549-30594” specifies bases 10549 to 30594 of the chr9 reference sequence; ▶ the path to a “regstringfile” which is a file containing lines of “accno:#-#” strings (one per line); ▶ the path to a GFF DAR file created during a NimbleGen Sequence Capture experiment, describing the “primary_target_region” regions of the experiment.
[-noreg]	This option turns off the region-based generation of the output.
[-annot *filename*]	An optional file describing the gene/coding-region annotations for the reference sequences, so that the variation detection can report gene names and protein translations of any identified variations in gene regions (on the HCDiffs tab). The format of this file must match that of the GoldenPath “refGene.txt” file. (See the second Note below for the case where the GS Reference Mapper automatically reads a gene annotation file.)
[-noannot]	Tells the GS Reference Mapper not to use an annotation file during the mapping. This option must be used to disable the automatic reading of annotation files.
[-snp filename]	An optional file describing the known SNP information for the reference sequences, so that the variation detection can link identified variations to the known SNP information (on the HCDiffs tab). The format of this file must match that of the GoldenPath snp128.txt file, and the contents must be for the same reference given to the GS Reference Mapper. (See the second Note below for the case where the GS Reference Mapper automatically reads a known SNP file.)
[-nosnp]	Tells the GS Reference Mapper not to use a known SNP file during the mapping. This option must be used to disable the automatic reading of known SNP files.

A final set of options exist, that can be used to adjust the mapping algorithm parameters (equivalent to the Parameter tab settings in the GUI). See section 6.2.4 for a more complete description of the parameter tab settings. The mapping computation has been optimized to give its best results using the default parameters, but these are provided for users who wish to experiment with adjusting parameters for their data sets.

Option	Description
[-ais]	This option sets the "Alignment identity score" parameter.
[-ads]	This option sets the "Alignment difference score" parameter.
[-hll]	This option sets the "Hit location limit" parameter.
[-hsl]	This option sets the "Hit-per-seed limit" parameter.
[-m]	An option telling the GS Reference Mapper to keep all sequence data "in memory" throughout the computation. This option can speed up the execution of the GS Reference Mapper, but requires the use of more computer memory.
[-mi]	This option sets the "Minimum overlap identity" parameter.
[-ml]	This option sets the "Minimum overlap length" parameter.
[-nor]	This option turns off the automatic "rescore" function that generates new quality scores for each SFF read using the new quality score algorithms.
[-sc]	This option sets the "Seed count" parameter.
[-rst]	This option sets the "Repeat score threshold" parameter.
[-sl]	This option sets the "Seed length" parameter.
[-ss]	This option sets the "Seed step" parameter.
[-ud]	An option telling the GS Reference Mapper to treat each read as a separate read, and not group them into duplicates for consensus calling or variation detection.

- The commands setRef, addRun, removeRun, and runProject can be executed in any combination and in any order, in a mapping project.
- Some of the runProject (for Mapping) command options listed above are mutually exclusive, for obvious functional reasons. Specifically:
  - Don't use -no or -nobig together with any of the following: -ace, -acedir, consed, -pair, or –pairt
  - Use no more than one of each of the following sets:
    - -noace, -ace, -acedir, or –consed
    - -pair (-p) or -pairt (-pt)
    - -ar, -at, or –ad
    - -a # or –g
    - -trim or –notrim
    - -vs or –novs
    - -vt (-v) or -novt (-nov)
    - -annot or –noannot
    - -snp or –nosnp
    - -n (-nimblegen) or -non (-nonimblegen)
    - -reg or -noreg

6.4 GS Reference Mapper Application Output

The GS Reference Mapper application uses a folder on the file system to hold the mapping project information (whether the mapping of the reads, the computation, is carried out in project-based mode through the GUI application or through the newMapping and related commands) and to hold the output files generated during the mapping computation. The contents of this folder and the names of the files generated by the application are the same for any mapping project or output folder.

6.4.1 Project/Output Folder Structure

Since the mapping computation is often performed on a pool of sequencing Runs (or Read Data files) rather than on any single Run, the result files it generates are not deposited in a Run folder. Two general cases exist.

- If the mapping is performed using the "one-step" command runMapping, a folder with a 'P_' prefix (for 'P'ost-Run Analysis) is created in the user's current working directory on the DataRig at the time the application is launched, or written to a directory specified by the user on the command line (or its GUI equivalent), to contain these files (see Figure 1–5). The name structure for this folder is as follows (see also section 6.4.2):

  "P_yyyy_mm_dd_hh_min_sec_runMapping"

The "-o" option can be specified on the command line to change the directory where the output files should be written (see section 5.3.1). If the specified directory exists, the output files will be written to that directory. If the specified directory does not exist, the program will create it, if possible.

- For "project-based" mapping, using the GUI application or using newMapping and related commands, the output is placed in a "project" folder (see Figure 1–7 and section 6.3.2). You can specify any name for your project folder; it will be recognized as a project folder by virtue of the "454Project.xml" file that will be automatically created within it. If you do not specify a directory name on the newMapping command line, the software will use the same default name as the runMapping command:

  "P_yyyy_mm_dd_hh_min_sec_runMapping"

As shown in Figure 1–7, the project-based mapping folder contains additional folders and files that mark the folder as a project folder and that store configuration information and internal data for the GS Reference Mapper application. A project folder comprises two sub-folders: a "mapping" sub-folder which contains the project state and output files; and an "sff" sub-folder containing the copies and/or symbolic links for the SFF files used as input to the mapping project. An additional file, "454Project.xml", is also placed there to identify the folder as a 454 project folder (and as a place holder for later integration, where mapping, assembly and/or amplicon "projects" can all reside in a single project folder and interact with the same data set(s), in an integrated fashion).

6.4.2 GS Reference Mapper Output Files

The GS Reference Mapper application creates the files listed in Table 6–1. Each time a mapping computation is carried out (no matter how many sequencing Runs or analysis regions are included), the files produced will have the names shown in Table 6–1, with the exception of the *.ace files (if individual, per reference sequence; see section 6.4.2.15) and the files in the "sff" subdirectory (see section 6.4.2.16, below). You can view the output files from the sample datasets provided in the accompanying DVD, using the GS Run browser application.

File Name Structure	Definition
454AllContigs.fna	FASTA file of all the consensus basecalled contigs longer than 100 bases.
454AllContigs.qual	Corresponding Phred-equivalent quality scores for each base in the consensus contigs in 454AllContigs.fna.
454LargeContigs.fna	FASTA file of all the "large" consensus basecalled contigs contained in 454AllContigs.fna. What constitutes a large contig is currently set to 500 bp.
454LargeContigs.qual	Corresponding Phred-equivalent quality scores for each base in the "large" consensus contigs in 454LargeContigs.fna.
454AllDiffs.txt	Text file listing all the differences between the reference sequence(s) and the reads included in the mapping, including a display of the multiple alignments of the differences and frequency statistics for each difference.
454HCDiffs.txt	Text file containing only the high-quality differences between the reference sequence(s) and the reads, including a display of the multiple alignment of the differences and frequency statistics for each of these differences.
454NewblerMetrics.txt	File providing the summary metrics for the mapping computation, including the number of input Runs and reads, the number and size of the large consensus contigs and the number of all consensus contigs.
454NewblerProgress.txt	A text log of the messages sent to standard output during the mapping computation.
454MappingQC.xls	A tab-delimited text file, suitable for import into MS Excel, containing detailed metrics about the mapping (see below).
454AlignmentInfo.tsv	Tab-delimited file giving position-by-position reference vs. consensus information.
454PairAlign.txt	A text file giving the pairwise alignment(s) found for each read mapped (only produced when using the –p option [or –pt option for the tab-delimited version of the file]).
454ReadStatus.txt	A tab-delimited text file providing a per-read report of the status of each read in the mapping.
454RefStatus.txt	A tab-delimited text file giving the mapping results for each of the sequences in the reference that had one or more reads mapping to them. This is useful *e.g.* when mapping survey or metagenomic sequences against a sequence database.
454TrimStatus.txt	Tab-delimited text file providing a per-read report of the original and revised trimpoints used in the mapping.

▶▶▶

File Name Structure	Definition
454Contigs.ace or ace/*refaccno*.ace or consed/…	ACE format file that can be loaded by third-party viewer programs that understand the ACE format, or consed sub-directory that can be loaded by the consed software. This can be a single file for the entire project or a folder containing individual files for each sequence in the reference (see Figure 1–7).
sff/*sfffilename*	The SFF file(s) used as input by the mapping computation.
454PairStatus.txt[1]	Tab-delimited text file providing a per-pair report of the location and status of how each Paired End pair of reads were used in the mapping.
454TagPairAlign.txt[1]	A text file giving the pairwise alignments used in the mapping computation for Paired End reads shorter than 50 bases (which are not part of the main mapping computation, but are mapped to the reference in a later computation step).

Table 6–1: Files generated by the GS Reference Mapper application. Words in *italics* are generic; real file names would have the actual accession number of the reads (*refaccno*) or full name of the SFF files. [1]Files produced only when the Paired End option is used.

6.4.2.1 454AllContigs.fna

This file contains the nucleotide sequences of all the contigs with a default minimum length of 100 nucleotides, produced by the GS Reference Mapper application. You can change the minimum length by using the [-a #] option. For a general description of the ".fna" (FASTA) file format, see section 13.3.6. For a sample of a Consensus Basecalled Contig file such as the 454AllContigs.fna file, see section 13.4.5.

6.4.2.2 454AllContigs.qual

This file contains the nucleotide Quality Scores (Phred-equivalent) for all the contigs with a default minimum length of 100 nucleotides, produced by the GS Reference Mapper application. You can change the minimum length by using the [-a #] option. For a description of the process used to compute individual base Quality Scores, see section 3.2.2.2. It should be remembered that Quality Scores indicate the probability that an individual called base is correct in the sequence, but provide no information on the possibility that a given homopolymer stretch might be longer than called. For a general description of the ".qual" file format, see section 13.3.6. For a sample of a quality score file for Consensus Basecalled Contigs, such as the 454AllContigs.qual file, see section 13.4.5.

6.4.2.3 454LargeContigs.fna

This file contains the same type of information as the 454AllContigs.fna file, but restricted to contigs longer than 500 called bases. The default value, set at 500 bases, can be modified by using the [-1 #] option.

6.4.2.4 454LargeContigs.qual

This file contains the same type of information as the 454AllContigs.qual file, but restricted to the long contigs listed in the 454LargeContigs.fna file (longer than 500 bases, by default. Use the [-1 #] option to modify the length).

6.4.2.5 454AllDiffs.txt

This file contains the list of variations where at least 2 reads differ either from the reference sequence or from other reads aligned at a specific location. SNPs, insertion-deletion pairs, multi-homopolymer insertion or deletion regions, and single-base overcalls and undercalls are reported. Also, in order for a difference to be identified and reported, there must be at least two non-duplicate reads that (1) show the difference, (2) have at least 5 bases on both sides of the difference, and (3) have few other isolated sequence differences in the read. In addition, if the –e option is used to set an expected depth, then there must be at least 5% of that depth in differing reads. Finally, for single-base overcalls or undercalls to be reported, they must have a flow signal distribution that differs from the signal distribution of the reads matching the reference (*i.e.* not all overcalls and undercalls are reported as variations). Once the difference is identified, all reads that fully span the difference location and have at least 5 additional flanking nucleotides on both sides are used in reporting the difference. For a sample 454AllDiffs.txt file, see section 13.4.8.

The file consists of one tab-delimited summary line for each difference, plus a multiple alignment of the reads spanning the difference location. Each of the difference summary lines begin with a '>' character, so the summary list of differences can be extracted from the file using the command "fgrep '>' 454AllDiffs.txt". The full summary lines contain fifteen columns of information, of which the first seven columns are always output, columns eight through eleven are output if gene annotations or known SNP information is given to the GS Reference Mapper (or the –fd option is given), and the rest are output only if the –fd option is given to the GS Reference Mapper. The summary lines contain the following columns:

1	The accession number of the reference sequence in which the difference was detected
2	The start position within the reference sequence, where the difference occurs
3	The end position within the reference sequence, where the difference occurs
4	The reference nucleotide sequence at the difference location
5	The differing nucleotide sequence at the difference location
6	The total number of reads that fully span the difference location
7	The percentage of different reads versus total reads that fully span the difference location
8	The reference amino acid sequence at the difference location, if it occurs within the coding region of an annotated gene
9	The differing amino acid sequence at the difference location, if it occurs within the coding region of an annotated gene
10	The gene name at the difference location, if it occurs with the region of an annotated gene
11	The list of known SNP ID's that occur at the difference location
12	The number of forward reads that include the difference
13	The number of reverse reads that include the difference
14	The total number of forward reads that fully span the difference location
15	The total number of reverse reads that fully span the difference location

The multiple alignment for a difference shows the difference location plus approximately 30 bases on either side. The alignment is divided into two sections, the first section displaying the reads found to contain the difference and the second, the reads not found to have the difference. Each line of the alignment displays the following information for a read:

1	The identifier for the read
2	The number of duplicate reads whose alignment matches this read (when there are duplicates of the read)
3	The position of the read's first base displayed in the alignment region
4	The orientation of the read in the displayed alignment ("+" is forward orientation, "-" is reverse-complement orientation)
5	The aligned bases of the read
6	The position of the read's last base displayed in the alignment region

■

6.4.2.6 454HCDiffs.txt

This file contains the same type of information as the 454AllDiffs.txt file (section 6.4.2.5, above), but restricted to the "High-Confidence" differences. The GS Reference Mapper application uses a combination of flow signal information, quality score information and difference type information to determine if a difference is High-Confidence. The general rules are:

- There must be at least 3 non-duplicate reads with the difference, unless the –e option is specified, in which case there must be 10% of the expected depth having the difference
- There must be both forward and reverse reads showing the difference, unless there are a at least 5 reads with quality scores over 20 (or 30 if the difference involves a 5-mer or higher)
- If the difference is a single-base overcall or undercall, then the reads with the difference must form the consensus of the sequenced reads (*i.e.*, at that location, the overall consensus must differ from the reference) and the signal distribution of the differing reads must vary from the matching reads (and the number of bases in that homopolymer of the reference).

6.4.2.7 454NewblerMetrics.txt

The 454NewblerMetrics.txt file is a 454 parser file that reports the key input, algorithmic and output metrics for the GS Reference Mapper application. Below is a description of all the sections present in this file, with their named groups and keywords. Note that the GS Reference Mapper application uses the same basic algorithms as the GS *De Novo* Assembler application to perform the initial alignment of the reads to the reference sequence, resulting in this 454NewblerMetrics.txt file; however, each application tailors its output to its specific purpose, so the content of the file is different. For a full description of the 454NewblerMetrics.txt file output by the GS *De Novo* Assembler application, see section 5.4.2.5. For a general description of the parser file format, see section 13.3.2. For a sample 454NewblerMetrics.txt file for the GS Reference Mapper application, see section 13.4.9.

1. referenceSequenceData group – contains information about the reference sequence file(s)
   a. path – the path to the directory with the FASTA reference sequence file
   b. numberOfReads – the number of reference sequences in the FASTA reference sequence file
   c. numberOfBases – the total number of bases in the FASTA reference sequence file

2. runData group – contains information about the inputs
   a. run group – information about each Run included in the mapping computation (for data directories only; not for explicit SFF files)
      I. path – the path to the data directory for this Run
      II. region group – information about each of the regions of this Run that are included in the mapping computation
         01. name – the name of the region
         02. sffFile – the name of the SFF file in the sff sub-directory of the GS Reference Mapper application's output directory
         03. numberOfReads – the number of reads that occur in the SFF file (*i.e.* that passed filtering), and then the number of reads used after primer trimming, quality trimming and vector screening
         04. numberOfBases – the total number of bases in the trimmed reads in the SFF file, and then the number of bases after primer trimming, quality trimming and vector screening
   b. file group – information about each input read SFF or FASTA file included in the mapping computation (for each explicit input read SFF or FASTA file on the command line; not for data directories)
      I. sffFile – the name of the SFF file in the sff sub-directory of the GS Reference Mapper application's output directory (SFF files only)
      II. path – the original path to the file given on the command line
      III. numberOfReads – the number of reads that occur in the file (*i.e.* that passed filtering, for SFF files), and then the number of reads used after primer trimming, quality trimming and vector screening
      IV. numberOfBases – the total number of bases in the trimmed reads in the file, and then the number of bases after primer trimming, quality trimming and vector screening

▶▶▶

3 pairedReadData group – contains information about the Paired End input data [Paired End only; 454 Sequencing reads only (not Sanger reads)]

a. run group – information about each Run included in the mapping computation (for data directories only; not for explicit SFF files)
   - I. path – the path to the data directory for this Run
   - II. region group – information about each region of this Run that are included in the assembly computation
     - *01.* name – the name of the region
     - *02.* sffFile – the name of the SFF file in the sff sub-directory of the GS Reference Mapper application's output directory
     - *03.* numberOfReads – the number of reads that occur in the SFF file (*i.e.* that passed filtering), and then the number of individual sequence reads used after dividing each SFF file read into its left and right half sequence reads, and after primer trimming, quality trimming and vector screening
     - *04.* numberOfBases – the number of bases in the trimmed reads in the SFF file, and then the number of bases used after dividing into left and right half sequence reads, primer trimming, quality trimming and vector screening
     - *05.* numberWithPairedRead – the number of reads found to have a valid paired read sequence

b. file group – information about each SFF file included in the mapping computation (for each explicit SFF file on the command line; not for data directories)
   - I. sffFile – the name of the SFF file in the sff sub-directory of the GS Reference Mapper application's output directory
   - II. path – the original path to the SFF file given on the command line
   - III. numberOfReads – the number of reads that occur in the SFF file (*i.e.* that passed filtering), and then the number of individual sequence reads used after dividing each SFF file read into its left and right half sequence reads, and after primer trimming, quality trimming and vector screening
   - IV. numberOfBases – the number of bases in the trimmed reads in the SFF file, and then the number of bases used after dividing into left and right half sequence reads, primer trimming, quality trimming and vector screening
   - V. numberWithPairedRead – the number of reads found to have a valid paired read sequence

►►►

4 runMetrics group – contains information about the mapping computation

a. numberOfReferenceSequences – the total number of reference sequences used for mapping purposes (equal to the sum of numberOfReads for each reference sequence in the referenceSequenceData group)
b. totalReferenceNumberOfBases – the total number of reference sequence bases used for mapping process (equal to the sum of numberOfBases for each reference sequence in the referenceSequenceData group)
c. totalNumberOfReads – the number of reads used in the mapping computation (equal to the sum of the numberOfReads for all the Runs/regions or SFF/FASTA files in the runData group)
d. totalNumberOfBases – the number of read's bases used in the mapping computation (equal to the sum of the numberOfBases for all the Runs/regions or SFF/FASTA files in runData group)
e. numberSearches – the number of alignment searches performed when aligning the reads to the reference
f. seedHitsFound – internal 454 metric giving the total number and average per search of the initial seed hits found during the exact k-mer matching phase of the alignment algorithm (similar to the initial exact k-mer matching used by blastn and other assemblers)
g. overlapsFound – internal 454 metric giving the total number, average per search and percentage of seedHitsFound of the alignments that passed the initial matching criteria
h. overlapsReported – internal 454 metric giving the total number, the average per search and the percentage of overlapsFound that were used after a filtering step that looks at all the alignments of a read to the same region of the reference sequence(s)
i. overlapsUsed – internal 454 metric giving the total number, the average per search and the percentage of overlapsReported that were used after a filtering step that looks at all the alignments found for that read

5 readMappingResults group – contains information about the mapping process per each input file (SFF, FASTA, or Run regions from wells file)

a. run group – information about each Run included in the mapping computation (for data directories only; not for explicit SFF files)
   I. path – the path to the data directory for this Run
   II. region group – information about each region of this Run that are included in the mapping computation
      01. name – the name of the region
      02. sffFile – the name of the SFF file in the sff sub-directory of the GS Reference Mapper application's output directory
      03. numMappedReads – the number of mapped reads (fully, partially and multiply mapped) from this input file (the percentage represents numMappedReads over numberOfReads from the runData group)
      04. numMappedBases – the number of mapped bases (from fully, partially and multiply mapped reads) from this input file (the percentage represents numMappedBases over numberOfBases from the runData group)
      05. inferredReadError – percentage of total number of differences (mapped errors) such as "indels" found in mapped reads over numMappedBases (the second number represents the number of differences)
      06. numUniquelyMapped – the number of uniquely mapped reads (fully and partially) from this input file [output only if targeted regions are specified]
      07. numUniqueInRegions – the number of uniquely mapped reads from this input file that align in the targeted regions [output only if targeted regions are specified]
      08. numUniqueOutOfRegions – the number of uniquely mapped reads from the input file that align outside of the targeted regions [output only if targeted regions are specified]

▶▶▶

5 b. file group – information about each SFF or FASTA file included in the mapping computation (for each explicit SFF or FASTA file on the command line; not for data directories)

- I. sffFile – the name of the SFF file in the sff sub-directory of the GS Reference Mapper application's output directory (SFF files only)
- II. path – the original path to the file given on the command line
- III. numMappedReads – the number of mapped reads (fully, partially and multiply mapped) from this input file (the percentage represents numMappedReads over numberOfReads from the runData group)
- IV. numMappedBases – the number of mapped bases (from fully, partially and multiply mapped reads) from this input file (the percentage represents numMappedBases over numberOfBases from the runData group)
- V. inferredReadError – percentage of total number of differences (mapped errors) such as "indels" found in mapped reads over numMappedBases (the second number represents the number of differences)
- VI. numUniquelyMapped – the number of uniquely mapped reads (fully and partially) from this input file [output only if targeted regions are specified]
- VII. numUniqueInRegions – the number of uniquely mapped reads from this input file that align in the targeted regions [output only if targeted regions are specified]
- VIII. numUniqueOutOfRegions – the number of uniquely mapped reads from the input file that align outside of the targeted regions [output only if targeted regions are specified]

6 pairedReadResults group – contains information about the Paired End input data [Paired End only; 454 Sequencing reads only (not Sanger reads)]

a. run group – information about each Run included in the mapping computation (for data directories only; not for explicit SFF files)

- I. path – the path to the data directory for this Run
- II. region group – information about each region of this Run that are included in the mapping computation
  - *01.* name – the name of the region
  - *02.* sffFile – the name of the SFF file in the sff sub-directory of the GS Reference Mapper application's output directory
  - *03.* numMappedReads – the number of mapped reads (fully, partially and multiply mapped) from this input file (the percentage represents numMappedReads over numberOfReads from the runData group)
  - *04.* numMappedBases – the number of mapped bases (from fully, partially and multiply mapped reads) from this input file (the percentage represents numMappedBases over numberOfBases from the runData group)
  - *05.* inferredReadError – percentage of total number of differences (mapped errors) such as "indels" found in mapped reads over numMappedBases (the second number represents the number of differences)
  - *06.* numUniquelyMapped – the number of uniquely mapped reads (fully and partially) from this input file [output only if targeted regions are specified]
  - *07.* numUniqueInRegions – the number of uniquely mapped reads from this input file that align in the targeted regions [output only if targeted regions are specified]
  - *08.* numUniqueOutOfRegions – the number of uniquely mapped reads from the input file that align outside of the targeted regions [output only if targeted regions are specified]
  - *09.* numberWithBothMapped – the number of reads where both halves uniquely mapped to the reference
  - *10.* numberWithOneUnmapped – the number of reads where one half failed to map to the reference
  - *11.* numberMultiplyMapped – the number of reads where one or both halves aligned equally well to multiple places in the reference
  - *12.* numberWithBothUnmapped – the number of reads where neither half aligned to the reference

▶▶▶

6 b. file group – information about each SFF or FASTA file included in the mapping computation (for each explicit SFF or FASTA file on the command line; not for data directories)
   - I. path – the original path to the file given on the command line
   - II. numMappedReads – the number of mapped reads (fully, partially and multiply mapped) from this input file (the percentage represents numMappedReads over numberOfReads from the runData group)
   - III. numMappedBases – the number of mapped bases (from fully, partially and multiply mapped reads) from this input file (the percentage represents numMappedBases over numberOfBases from the runData group)
   - IV. inferredReadError – percentage of total number of differences (mapped errors) such as “indels” found in mapped reads over numMappedBases (the second number represents the number of differences)
   - V. numUniquelyMapped – the number of uniquely mapped reads (fully and partially) from this input file [output only if targeted regions are specified]
   - VI. numUniqueInRegions – the number of uniquely mapped reads from this input file that align in the targeted regions [output only if targeted regions are specified]
   - VII. numUniqueOutOfRegions – the number of uniquely mapped reads from the input file that align outside of the targeted regions [output only if targeted regions are specified]
   - VIII. numberWithBothMapped – the number of reads where both halves uniquely mapped to the reference
   - IX. numberWithOneUnmapped – the number of reads where one half failed to map to the reference
   - X. numberMultiplyMapped – the number of reads where one or both halves aligned equally well to multiple places in the reference
   - XI. numberWithBothUnmapped – the number of reads where neither half aligned to the reference

7 consensusDistribution group – contains information about the consensus signals and basecalling thresholds

a. fullDistribution group – a histogram of the signal intensities of the consensus signal averages

b. distributionPeaks group – the identified peaks used as the mean signal for a n-mer homopolymer stretch

c. thresholdsUsed group – the thresholds used to perform basecalling of the signals, along with an interpolation amount used when the signal is above the detectable peaks

▶▶▶

8 consensusResults group – contains information about the results

a. readStatus group – contains information about the fates of the reads
   I. numMappedReads – the total number of fully, partially and multiply mapped reads (the percentage represents numMappedReads over the sum of numberOfReads across the input files)
   II. numMappedBases – the total number of mapped bases from fully, partially and multiply mapped reads (the percentage represents numMappedBases over the sum of numberOfBases across the input files)
   III. inferredReadError – percentage of total number of differences (mapped errors) such as “indels” found in mapped reads over numMappedBases (the second number represents the number of differences)
   IV. numberFullyMapped – the number of reads fully mapped into the contigs
   V. numberPartiallyMapped – the number of reads partially mapped into the contigs
   VI. numberUnmapped – the number of reads that did not overlap with any of the reference sequences
   VII. numberRepeat – the number of reads mapped to multiple places in one or more reference sequences.
   VIII. numberTooShort – the number of reads whose trimmed sequence was too short to be used
   IX. numUniquelyMapped – the total number of uniquely mapped reads (fully and partially) [output only if targeted regions are specified]
   X. numUniqueInRegions – the total number of uniquely mapped reads that align in the targeted regions [output only if targeted regions are specified]
   XI. numUniqueOutOfRegions – the total number of uniquely mapped reads that align outside of the targeted regions [output only if targeted regions are specified]

b. pairedReadStatus – contains information about the fates of the paired reads [Paired End only (454 Sequencing reads or Sanger reads)]
   I. numberWithBothMapped – the number of reads where both halves align uniquely to the reference
   II. numberWithOneUnmapped – the number of reads where one half failed to align to the reference
   III. numberMultiplyMapped – the number of reads where one or both halves aligned equally well to multiple places in the reference
   IV. numberWithBothUnmapped – the number of reads where neither half aligned to the reference
   V. library – information for each Paired End library in the input data. Sanger reads explicitly identify the library they came from; each 454 Paired End region or SFF file is treated by default as a separate library
      01. libraryName – the name of the Paired End library
      02. pairDistanceAvg – the average distance between both halves, when both halves align on the same contig
      03. pairDistanceDev – the standard deviation of the pairDistanceAvg distribution

▶▶▶

8

c. largeContigMetrics group – contains information about the large contigs
   I. numberOfContigs – the number of large contigs identified
   II. numberOfBases – the total number of bases in the large contigs
   III. avgContigSize – the average large contig size
   IV. N50ContigSize – the N50 contig size
   V. largestContigSize – the size of the largest contig
   VI. Q40PlusBases – the number and percentage of bases called that have a quality score of 40 or above
   VII. Q39MinusBases – the number and percentage of bases called that have a quality score of 39 or below
   VIII. numUndercalls – total number of undercalls (bases missing in the consensus), comparing the reference and the contig consensus
   IX. numOvercalls – total number of overcalls (extra bases in the consensus), comparing the reference and the contig consensus
   X. numHCUndercalls – number of undercalls whose computed probability is equal to a quality score of 40 or above
   XI. numHCOvercalls – number of overcalls where the consensus bases have a quality score of 40 or above
   XII. consensusAccuracy – overall accuracy of the consensus, as compared to the reference
   XIII. HCconsensusAccuracy – accuracy of the Q40Plus bases of the consensus, as compared to the reference

d. allContigMetrics group – contains information about all the contigs generated
   I. numberOfContigs – the number of contigs identified
   II. numberOfBases – the total number of bases in the contigs

6.4.2.8 454NewblerProgress.txt

This file represents the text log of the messages sent to standard output by the runProject command (showing the progress of the execution of the mapping computation). If Runs are added incrementally and multiple executions of runProject occur, the output messages are appended to this file. If the "Incremental reference mapping analysis" checkbox option is not selected in the GUI application, or the "-r" option is given on the runProject command line, the GS Reference Mapper deletes intermediate data and "restarts" the mapping computation, and this file is deleted and restarted as well. For a sample 454NewblerProgress.txt file, see section 13.4.10.

6.4.2.9 454MappingQC.xls

This file contains a number of detailed metrics regarding the mapping results; its format and structure are intended for reading by MS Excel or a similar spreadsheet program, so that the metrics can be easily visualized using Excel's charting/graphing tools. The file is in tab-delimited format, and contains six main sections (Note: The section titles given here are not displayed in the file.)

1 Summary Statistics

a. Num. Reads – the number of input reads used in the mapping computation
b. Num. Bases – the number of bases in the input reads
c. Mapped Reads – the number and percentage of reads that uniquely mapped to the reference, followed by the number and percentage of reads that uniquely or multiply mapped
d. Mapped Bases – the number and percentage of bases that uniquely mapped to the reference, followed by the number and percentage of reads that uniquely or multiply mapped
e. Inf. Read Error – the "inferred read error" percentage and quality score (calculated as the number of read alignment differences over the number of mapped bases), along with the counts of the number of read alignment differences and mapped bases
f. Exp. Read Error – the expected read error computed from the input read quality scores, given as a percentage, quality score and expected number of alignment differences. This is computed by summing the expected number of errors for each quality score value (*i.e.* number of bases with a quality score times the accuracy rate of that quality score).
g. Last 100 Base IRE – the "inferred read error" numbers, using only the last (3') 100 bases of each read
h. Last 50 Base IRE – the "inferred read error" numbers, using only the last (3') 50 bases of each read
i. Last 20 Base IRE – the "inferred read error" numbers, using only the last (3') 20 bases of each read
j. Genome Size – the number of bases in the reference
k. Num. Large Contigs – the number of large contigs reported in the 454LargeContigs.fna file
l. Num. Large Contig Bases – the number of bases in the large contigs
m. Avg. Depth – the average alignment depth (*i.e.* how many reads aligned to each position of the reference)
n. Avg. Map Length – the average length of the alignment of a read (the read's "map length")

2 Read Error Histogram

a. This section breaks down the number of reads based on their read status and/or number of alignment differences, and displays percentages and counts per category.
b. Reads that did not map to the reference ("Unmapped"), reads where only part of the sequence mapped to the reference ("Partial") and reads that mapped to multiple locations in the reference ("Multiple") are displayed as one category each.
c. Reads that mapped fully to the reference (meaning every base of the read occurred in the alignment) are then divided by the number of alignment differences found, and percentages and number of reads having 0, 1, 2, …, 9 or 10+ (10 or more) errors are shown.

►►►

3 Overcall/Undercall Tables

a. These two tables display the percentages and numbers of homopolymer accuracy, using the read alignments to count when the read contains the same or different homopolymer length as compared to the reference.

b. The rows are the read homopolymer length and the columns are the reference homopolymer length.

c. When scanning the read alignments, the reference sequence homopolymer is first determined, then the alignment of the read to that homopolymer is evaluated and counted

I. Some reads may have multiple homopolymers aligned to a single reference homopolymer (like "ACA" align to "AAA"). These are counted and displayed separately on a "Mult" row.

II. The percentages shown in the first overcall/undercall table are given as a percentage of the column (*e.g.* what percent of the time at a reference 5-mers did the read have a 4-mer). Also, the percent table does not show the percentage of the correct alignments (*e.g.* 5-mer to 5-mer), nor does it show percentages less than 0.1% (in order to highlight the overcall/undercall trend).

III. Below the counts table, a "%Ident" row displays the percentages for each reference n-mer where the read was called correctly (*i.e.* its homopolymer length matched the reference).

4 GC Content Information

a. GC Observed/Expected – the two lines below this display the GC content percentages (from 0 to 100) and the observed over expected mapping depth. This is calculated by first counting the number of reads with particular GC content and counting the GC content of all windows of the reference (where the window length matches the average read flowspace or nucleotide length). Then the two counts (read and reference) for a specific GC content value are divided by the read/reference totals to compute the percentage of the reads/references with that GC content. The observed/expected value is the ratio of those two percentages.

b. GC Std. Dev. – this is the standard deviation of the GC Observed/Expected (based on the sampling at that GC content value). The values on this line are useful for setting the "Y Error Bars" information in Excel, if an "XY (Scatter)" chart is made using the GC Observed/Expected two lines as the source data. This line can then be used as the "+" and "-" data of the "Custom" Error amount, found inside the "Y Error Bars" tab of the "Format Data Series" dialog box).

5 Quality Score Information

a. Predicted Score – The quality score values, from 0 to 60

b. Observed Quality – The observed quality score obtained from the read alignments (computed as "Observed Num. Errors" over "Num. Bases With Score" values, see below)

c. Observed Accuracy – The observed quality score expressed as an accuracy percentage

d. Num. Bases With Score – the number of mapped bases having the Predicted Score (only mapped bases are used, because they can be evaluated for accuracy)

e. Expected Num. Errors – the expected number of errors for a quality score, given the number of mapped bases with that quality score

f. Observed Num. Errors – the number of bases which did not match in the read alignment (*i.e.* the alignment column containing that base was not an identity)

▶▶▶

6 Histogram/Window-based Information

This final section contains the data for larger histograms and datasets, and is organized column-wise, instead of the row-wise layout of the previous sections. It contains three sub-sections

a. Read Length and Map Length Histograms – histograms showing the number of reads of each read length (the “Read Length Histogram” column) and number of reads at each length of the aligned regions per the reference sequence, *i.e.* counting only the read bases in the alignment, not the alignment length (the “Map Length Histogram” column). Histogram values are displayed up to 400 bases.

b. Errors by Base Position – plot values showing position-by-position errors in the reads, *i.e.*, how many errors occurred at the N’th base across all the reads. The four columns show the accuracy percentage and equivalent quality score of the accuracy at a specific position (the “Errors by Base Position” columns) and the cumulative accuracy up to that position (the “Cumul. Errors by Base Position” columns)

   I. Note: if an alignment column contains a gap in the read, that is counted as an error at the previous base position (*i.e.*, any alignment gaps between base 5 and 6 in a read are counted as errors at position 5)

c. Cross-Reference Depth and GC Information

   The last six columns of this section contain region-by-region statistics of the alignments across the reference, where the reference is evenly divided into 1000 regions

   This division of the reference into regions has no understanding of repeat regions, and simply reports on the alignments of the uniquely mapping reads. Since repeat reads are not aligned to the reference, the values in this column will count repeat regions as unaligned regions.

   I. The first column displays the position in the reference sequence at the center of the region
   II. Avg. Depth – the average alignment depth in the region
   III. Min. Depth – the minimum alignment depth in the region
   IV. Max. Depth – the maximum alignment depth in the region
   V. Depth Score – a score that is indicative of the shallowness of the alignment in the region. Each alignment column in the region is given a score of “max(0, 4-depth)” where “depth” is the alignment depth of the column. The Depth Score for a region is the sum of the column scores.
      01. This score is a very sensitive metric for use in resequencing projects, in order to gauge when enough sequencing has been performed (and the addition of more reads will not fill in any more unaligned or shallowly aligned regions of the reference)
   VI. GC – the average GC content of the region

6.4.2.10 454AlignmentInfo.tsv

The 454AlignmentInfo.tsv file contains position-by-position summary information about the alignment of the reference sequence and the consensus generated from the aligned reads, listed one nucleotide per line (in a tab-delimited format). For a sample 454AlignmentInfo.tsv file, see section 13.4.11.

The columns of each line contain the following information:

1	Position – the position in the reference sequence (or the position of the last reference base if the alignment has a gap for the reference)
2	Reference – the alignment character for the reference, either a reference base or a gap ('-')
3	Consensus – the alignment character of the consensus of the aligned reads, either a nucleotide or a gap ('-')
4	Quality Score – the quality score of the consensus base, or the quality score giving the probability of an undercall at that location, in the case of a gap for the consensus
5	Depth – the number of reads that align at that position in the alignment
6	Signal – the average signal of the read flowgrams, for the aligned flows that correspond to that position in the alignment
7	StdDevation – the standard deviation of the read flowgram signals at the corresponding flows

This file only contains regions of the reference sequence where the aligned depth is greater than zero and, prior to each region of lines, a header line beginning with a '>' displays the accession number of the sequence in the reference and starting position of the next region with a non-zero aligned depth.

6.4.2.11 454PairAlign.txt

This file contains the pairwise alignments that were found when mapping the reads against the reference. By default, this file is not generated, but if the "-p" or "-pt" options are given on the runProject command line, this file will be generated either in a human-readable text format ("-p") or in tab-delimited format ("-pt").

Each of the displayed alignments contain the following information (these are the columns in the tab-delimited format):

1	QueryAccno – accession number of the read (the "query sequence")
2	QueryStart – starting position of the alignment in query sequence
3	QueryEnd – ending position of the alignment in query sequence
4	QueryLength – length of the query sequence
5	SubjAccno – accession number of the sequence in the reference where the alignment occurs (the "subject sequence")
6	SubjStart – starting position of the alignment in subject sequence
7	SubjEnd – ending position of the alignment in subject sequence
8	SubjLength – length of the subject sequence
9	NumIdent – number of identities in the pairwise alignment, *i.e.* where query and subject characters match
10	AlignLength – the length of the pairwise alignment
11	QueryAlign – query alignment sequence
12	SubjAlign – subject alignment sequence

For each uniquely mapped read (the 'Full' and 'Partial' reads), the file contains the unique alignment found. For reads that map to multiple locations in the genome (some of the 'Repeat' reads), all the alignments to those multiple locations will be computed. However, only the highest quality alignments will be output to the file (*i.e.*, if a 99% identity alignment is found, a 90% identity alignment will not be output, as it is not a likely mapping location of the read to the genome, given that a 99% identity location was found).

6.4.2.12 454ReadStatus.txt

The 454ReadStatus.txt file contains the status identifiers for all the reads used in the mapping computation, listed one per line, in tab-delimited format. The status string describes the read's fate in the alignment, and can be one of the following four values:

- Full – the read is fully aligned to the reference (every base)
- Partial – only part of the read aligned to the reference
- Repeat – the read aligned equally well to multiple locations in the reference
- Unmapped – the read did not align to the reference
- TooShort – the trimmed read was too short to be used in the computation (*i.e.*, shorter than 50 bases, unless 454 Paired End Reads are included in the dataset, in which case, shorter than 15 bases).

The lines for reads marked 'Full' and 'Partial' also contain information about the alignment identity/length. The format for a 'Full' line is "*accno*\tFull\t*pctIdent*", where *accno* is the read's accession number, *pctIdent* is the percentage identity of the alignment (rounded to the nearest whole number) and "\t" represents the tab character separator. The format for a 'Partial' line is "*accno*\tPartial\t*pctIdent*\t*pctAligned*", where *pctIdent* is the percentage identity of the alignment (considering only the part of the read found to actually align to the reference, per the computation), and *pctAligned* is the percentage of the read that occurs in the alignment.

For a sample 454ReadStatus.txt file resulting from a mapping computation, see section 13.4.12. Note that as the possible fates of reads in a mapping computation are different from possible fates in an assembly computation, the file of the same name, 454ReadStatus.txt, produced by the GS *De Novo* Assembler (see sections 5.4.2.10 and 13.4.7) is completely different from that produced by the GS Reference Mapper.

6.4.2.13 454RefStatus.txt

The 454RefStatus.txt file contains the summary mapping results for the individual sequences in the reference, listed one per line, in tab-delimited format. For a sample 454RefStatus.txt file, see section 13.4.13. The lines contain the following information:

- Reference Accession – the accession number of the sequence in the reference
- Num Unique Matching Reads – the number of reads that uniquely map to this sequence
- Pct of All Unique Matches – the percentage of uniquely mapping reads that map to this sequence (*i.e.*, Num Unique Matching Reads divided by the total number of uniquely mapping reads)
- Pct of All Reads – the percentage of all reads that uniquely map to this sequence (*i.e.*, Num Unique Matching Reads divided by the total number of input reads)
- Coverage of Reference – the percentage of the sequence "covered" by uniquely aligning reads (where at least one read aligns at that position of the sequence)

Only sequences in the reference where at least one read uniquely maps to the sequence are listed. This file is useful when using the GS Reference Mapper application to map reads to a database of sequence, such as in a survey-sequencing or a metagenomics project.

6.4.2.14 454TrimStatus.txt

This file contains a per-read report of the original and revised trimpoints used in the assembly, where the revised trimpoints include the effects of primer, vector and/or quality trimming of the original input reads. Each line contains the following information (these are the columns in the tab-delimited format):

1. Accno – accession number of the input read
2. Trimpoints Used – the final trimpoints used in the assembly, in #-# format
3. Trimmed Length – the final trimmed length of the read
4. Orig. Trimpoints – the original trimpoints of the read, found in the SFF or FASTA file
5. Orig. Trimmed Length – the original trimmed length of the read
6. Raw Length – the length of the raw read (without any trimming)

6.4.2.15 454Contigs.ace or ace/refaccno.ace or consed/...

This viewer-ready genome file shows all the sequences in the reference and allows the display of how the reads and contigs aligned to those sequences, in an ACE format file suitable for use in various third-party sequence finishing programs. (The freeware "clview" application can be downloaded from: http://www.tigr.org/tdb/tgi/software/; a full description of the .ace file format can be found at: http://bozeman.mbt.washington.edu/consed/consed.html.) It should be noted, however, that such third-party viewing software will not be able to make full use of the flowspace assembly information available with Genome Sequencer reads, and that conversely some of the third-party program's functions (*e.g.* involving sequence chromatogram input) are not usable with Genome Sequencer datasets. Nonetheless, these programs may be useful to view and assess read characteristics and coverage depth in regions of interest.

The number of ACE files generated during a mapping computation is determined by which ACE option is checked, as described in sections 6.2.4 (for project-based mapping using the GUI) and 6.3.2.5 (for mapping carried out from the command line).

The GS Reference Mapper application can also output the mapping results in a complete directory structure suitable for use as input to the consed software (see the above link for a description of this structure and its files). When the appropriate option is selected, a "consed" sub-directory is produced in the output (or project) directory. This directory contains the sub-directories, the ACE file and the PHD file, so that the functions of consed (viewing traces, editing reads, autofinishing) can be performed on the mapping results. In order to integrate the 454 Sequencing reads (and their SFF files) into the consed data, an extra "sff_dir" directory is created in the consed sub-directory, and a number of consed options are automatically specified in this directory (see the "consed/edit_dir/.consedrc" file for the options specified by the generated structure). One option tells consed to use the "sff2scf" command to access any trace information requested by a user. See Section 7.3 for a description of the sff2scf command, and how it can generate synthetic traces from SFF data, as well as provide a pass-through access to SCF data for Sanger reads.

The structure of the ACE file produced by the GS Reference Mapper application differs from the traditional assembly ACE files, because it is displaying the sequence information within the context of the given reference sequence(s). Each individual sequence in the reference is output as a "contig" in the ACE file (so that the alignments of the 454 reads to each reference sequence can be viewed relative to the reference sequence bases in the "contig alignment viewers" that most third-party ACE file viewers contain). The 454 reads, as well as the contigs generated by the GS Reference Mapper application, are output in the ACE file as "reads", again to organize all the data relative to the reference.

6.4.2.16 sff/sfffilename

The SFF files used for mapping are the same as for assembly. See section 5.4.2.13.

6.4.2.17 454PairStatus.txt

This file contains the per-pair report of the location and status of how each Paired End pair of reads were mapped. Each line contains the following information (these are the columns in the tab-delimited format):

1. Template – template string for the pair (this will be the original 454 accession for 454 Paired End reads, and the "template" string for Sanger reads)
2. Status – the status of the pair in the mapping, with the following possible values
   a. BothUnmapped – both halves of the pair were unmapped
   b. OneUnmapped – one of the reads in the pair were unmapped
   c. MultiplyMapped – one or both of the reads in the pair were marked as Repeat
   d. TruePair – both halves of the pair were mapped into the same reference sequence, with the correct relative direction, and are within the expected distance of each other
   e. FalsePair – the halves were mapped to the same reference sequence, but the directions of the alignment is inconsistent with a Paired End pair or the distance between the halves is outside the expected distance
3. Distance – for "TruePair" or "FalsePair" pairs, the distance between the halves.
4. Left Contig – the contig where the left half was mapped, or "-" if the read was Unmapped or Repeat.
5. Left Pos – the position in the contig where the 5' end of the left half was mapped
6. Left Dir – the direction ('+' for the forward strand of the reference sequence and '-' for reverse strand) in which the left half was mapped
7. Right Contig – the contig where the right half was mapped, or "-" if the read was Unmapped or Repeat.
8. Right Pos – the position in the contig where the 5' end of the right half was mapped
9. Right Dir – the direction ('+' for the forward strand of the reference sequence and '-' for reverse strand) in which the right half was mapped
10. Left Distance – the distance from the Left Pos to the respective end of the reference sequence (for forward matches, this is the distance to the 3' end of the sequence, for reverse matches, to the 5' end)
11. Right Distance – the distance from the Right Pos to the respective end of the reference sequence (for forward matches, this is the distance to the 3' end of the sequence, for reverse matches, to the 5' end).

6.4.2.18 454TagPairAlign.txt

This file contains the pairwise alignments of "short" reads that were found during the mapping computation. When 454 Paired End reads are included in the dataset, reads that are between 15 and 50 bases long are not included in the main overlap computation, but are mapped to the references in a later step of the computation (using different alignment detection settings). This file reports the alignments of the uniquely mapping 15–50 bp reads against the reference (this version of software is not capable of reporting the alignments for multiply mapped reads). By default, this file is not generated, but if the "-p" or "-pt" options are given on the runProject command line, this file will be generated either in a human-readable text format ("-p") or in tab-delimited format ("-pt"). The format of this file is identical to the 454PairAlign.txt file described in section 6.4.2.11.

7. SFF Tool Commands

The Genome Sequencer FLX System off-instrument software package contains five programs related to the handling of Standard Flowgram Format (SFF) files or other read data:

- for combining files into the Standard Flowgram Format (SFF) and filtering reads present in SFF files [sfffile]; for accessing SFF file information [sffinfo];
- for dynamically generating an SCF trace file suitable for display by the consed software [sff2scf];
- for preparing FASTA files with the necessary annotations for use by the GS *De Novo* Assembler [fnafile]; and
- for rescoring older SFF files (generated prior to the 1.1.03 software) using the new phred-based quality scores [sffrescore].

These tools are all evoked at the UNIX command line level. The descriptions below assume that the reader is familiar with the Genome Sequencer FLX System data and formats, including the SFF file format. For more details on the SFF and other GS FLX file formats, see section 13.

7.1 sfffile

The sfffile command constructs a single SFF file containing the reads from a list of SFF files and/or datasets from sequencing Runs. The reads written to the new SFF file can be filtered using inclusion and exclusion lists of accession numbers, and their sequence trimming points or flowgram lengths can be adjusted. This command is used to pool the results of multiple sequencing Runs (or specified regions of one or multiple Runs) into a single SFF file, to simplify further handling of the combined data.

The sfffile command uses the following syntax:

```
    sfffile [options...] [MIDList@](sfffile | datadir)...
```

…where "MIDList" is a multiplexing information string used to filter the set of file reads output (see the fourth Note below for the format of the MIDList information).

Command	Description
sfffile	Constructs a single SFF file containing the reads from a list of SFF files and/or datasets from older sequencing Runs.

Option / Argument	Description
-o output	The default output of the sfffile program is a single SFF file containing all the reads of the SFF files/sequencing Runs given in argument on the command line, output to the file "454Reads.sff". The "-o" option allows the user to specify a different filename.
-r	This option re-generates the phred-based quality scores for each of the input reads using the current quality scoring table, and overwrites the existing quality scores with these new quality scores in the output file.

▶▶▶

Option / Argument	**Description**
-i accnofile	By default, all reads in the inputs are written to the output SFF file. If the "-i" (Include only these reads in output) or "-e" (Exclude these reads from output) options are used, the accession numbers given in the files specified will change the reads that are output. (If both -i and -e are used, the output will include the reads that are in the -i file and not in the -e file.) Those files should list the accessions one per line, where the accession should be the first word on the line after an optional '>' (*i.e.*, if the line begins with '>', that character is skipped, and the following characters up to the first whitespace character is read as the accession). These options are cumulative, *i.e.* if multiple -i options are given, the reads included will be the union of the -i accession lists.
-e accnofile	
-t trimfile or -tr trimfile	These two options adjust the trim points for some or all reads in the output SFF file. The specified "trimfile" should contain one or more lines consisting of (1) a read accession number, (2) a starting trimpoint, and (3) an ending trimpoint, separated by whitespace characters or where the trimpoints are separated by a dash (*e.g.*, "accno 12 543" or "accno 12-543"). The trimpoint values are 1-based positions that denote the first and last base of the trimmed region (*e.g.* for a read 800 bases in length, the lines above specify that bases 1-11 and 544-800 should be ignored, and bases 12-543 form the trimmed region of the read). A value of 0 specifies that the beginning or end of the read should be used (*e.g.* for a read 800 bases in length, the line "accno 12 0" sets the trimmed region to 12-800). The –t option will merge the given trimpoints with any existing trim points for the input read, writing the largest starting trimpoint and smallest ending trimpoint into the output SFF file. By contrast, the –tr option will "reset" the trimpoints, using only the trimpoint information occurring in this file. Note: The use of this option does not limit the reads that will be written into the output SFF file. To only output the reads in the trim file, the –i option must also be used, with this file as its argument.
-c # or -gs20 or –20 or -gsflx or –flx	These options change the length of the flowgrams written to the output SFF file, shortening or lengthening the flowgrams (and associated basecalled sequence) to the given number of cycles in the argument. The -c option allows an arbitrary number of cycles to be specified, while the –gs20 and –20 options are a shortcut for "–c 42" and the –gsflx and –flx options are a shortcut for "–c 100" (the number of cycles for sequencing Runs carried out using the GS 20 chemistry and the GS FLX standard chemistry, respectively). If a flowgram is lengthened, 0.0 flowgram values will be added to the end of the flowgram, and no bases will be called for those flowgram values. If a flowgram is shortened, the ending flows, and all basecalls made from those flows, will be removed from the read's entry. Reads whose flowgram length matches the specified number of cycles will not be altered. Note: If a flowgram and its read are shortened, the removed data does not exist in, and is not recoverable from, the output SFF file.

▶▶▶

Option / Argument	Description
-s setname	This option “splits” the input reads into separate output files, based on the multiplexing sequences occurring at the beginning of the reads. The argument to the option is an MID set name (occurring in the MID configuration file). Each input read is matched against each MID in the set, and output to a separate file. The output files will be named by adding the MID name as a suffix to the output file (but before the “.sff” suffix). So, if the default GS Multiplex Identifiers (MID) Kit set is given as the set name and the output filename is “454Reads.sff”, the output files will be “454Reads.MID1.sff”, “454Reads.MID2.sff”, and so on. Note: If no reads match a given MID sequence, the corresponding output file is not written (*i.e.*, only files with one or more reads will be output).
[-mcf *filename*]	This option specifies a different MID configuration file to be used by the command for decoding the multiplex information appearing on the command line.
[-pick #] or [-pickb #]	This option tells sfffile to generate an output file containing a certain number of bases, by randomly “picking” reads from the input. The argument can be a number, optionally followed by either a ‘k’ or ‘m’ to specify thousands or millions of bases, respectively. For example, “-pick 3000000” and “-pick 3m” both tell sfffile to pick random reads from the input and generate an output file containing 3 million bases. Note: Reads are not broken to achieve exactly the number of bases specified, so the actual number of bases in the output file may differ slightly from the desired number.
[-pickr #]	Path to one or more SFF files and/or Data Processing directories (“D_...”). Reads in any input SFF files are merged directly into the single output SFF file generated by this command. If SFF files are not present in a Data Processing folder (*e.g.* for a Run whose data has been processed with a version of the Genome Sequencer System software anterior to 1.0.52), the signal processing step of the GS Run Processor application must be rerun on the Data Processing folder, and can then merged into the output SFF file generated by the sfffile command. Input “D_...” directories may be prepended with a list of regions separated by a colon. For example, “1,3-5,7:R_dir/D_dir” tells sfffile to use regions 1, 3, 4, 5 and 7 of R_dir/D_dir. An optional multiplexing information string can be prepended to each file/data-directory argument (see the fourth Note below). This option tells sfffile to generate an output file containing a certain number of reads, by randomly “picking” reads from the input. The argument number can be followed by a ‘k’ or ‘m’ to specify thousands or millions of reads, respectively.
[-nmft]	By default, a “manifest” listing the set of sequencing Runs that were the source of the reads in an SFF file is written into the index section of the SFF file (to provide an explicit traceback to the origin of the reads). This includes the Run name (the R_... name), the Data Processing name (the D_... name) and the full path to the Data Processing directory used in the conversion from Run data to SFF file(s). The “-nmft” option prevents the propagation of the manifest from the input SFF files into the output SFF file.
[--help]	Displays a usage line and short description of the command.
[--version]	Displays the current software version of the command.
[MIDList@] (sfffile | datadir)	Path to one or more SFF files and/or Data Processing directories (“D_...”). Reads in any input SFF files or the SFF files in the Data Processing directories are merged directly into the single output SFF file generated by this command. Input “D_...” directories may be prepended with a list of regions separated by a colon. For example, “1,3-5,7:R_dir/D_dir” tells sfffile to use regions 1, 3, 4, 5 and 7 of R_dir/D_dir. An optional multiplexing information string can be prepended to each file/data-directory argument (see the fourth Note below).

- One of the most common uses of the sfffile program is to collect and organize your sequences in terms of your projects, instead of the standard timestamp- and Run-based organization generated by the Genome Sequencer FLX Instrument. For example, multiple sequencing Runs for a large sequencing project can easily be combined into a single "mygenome.sff" file using "sfffile -o mygenome.sff path_to_D1 path_to_D2 path_to_D3…". This file could then be given to the GS Reference Mapper and GS *De Novo* Assembler software, or carved and divided into separate pieces (see below), while still preserving the traceback information to the source data, since:

  - The universal accessions for each read encode the Run timestamp, the region and the X,Y location of the read, and all but ensure uniqueness for the accessions.

  - The manifest of the SFF file provides the mapping from the Run prefix of the universal accessions to the actual Run and Data Analysis directories used to generate the read. When universal accessions are given to the reads, the manifest is kept updated through multiple invocations of sfffile (as best it can).

  So, if a read needs more investigation, you can invoke the sffinfo command (see section 7.2, below) for that read to see the traceback information, including the source Run and Data Analysis directories. The GS Run Browser (see section 8) could be used to evaluate the read within the context of the Run [where the Find location feature on the Wells tab allows entry of a universal accession and will then center the image and place a marker at the read's X,Y location].

- Another common use of sfffile is to extract subsets of the reads from a sequencing Run or a set of sequencing Runs, to perform further processing. Using the –i and –e options, one can construct an SFF file containing only the reads of interest, and then use that for further processing, *e.g.* input it into to the GS Reference Mapper or GS *De Novo* Assembler applications. For example, you can extract the singletons from an assembly (when performing, say, survey sequencing) by executing the following commands (which assume that the current working directory is the output directory of the assembly):

```
fgrep Singleton 454ReadStatus.txt > singles.txt
sfffile -o singles.sff -i singles.txt sff/*
sffinfo -s singles.sff > singles.fna
```

  Or, after running the GS Reference Mapper application, you can retrieve the reads that did not map to your reference sequence (so that, for example, you can give them as input to the GS *De Novo* Assembler software to see if there are any contigs that are part of what you sequenced but are not part of the reference) by using the following commands (which assume that the current working directory is the output directory of the assembly):

```
grep Unmapped 454ReadStatus.txt > unmapped.txt
sfffile -o unmapped.sff –i unmapped.txt sff/*
runAssembly unmapped.sff
```

  …where the Unix grep command extracts the lines in the 454ReadStatus.txt file that list the accession numbers of the unmapped reads, then the –i option in the sfffile command is used to signify that the command should "include" those reads in the new SFF file. (Similarly, there is a -e option to "exclude" reads from the output file.) Note that the -i and -e options cannot take a list of files, so to specify the inclusion or exclusion of multiple files on the command, you must precede each with an -i or -e option.

- The path for any filename or data directory name can be prepended with a multiplexing information string, separated from the filename/directory-string by an "@" sign. Multiplexing information strings should be used when multiple different samples have been prepared using different initial "tag" sequences (such as those provided by the GS Multiplex Identifiers (MID) Kits), then mixed together for sequencing. The software uses these tag sequences to segregate the reads from each sample, by matching the initial bases of the reads to the known tag sequences used in the preparation of the libraries. If a multiplex string is given, sfffile will automatically match the 5' bases of each read against the multiplex sequences, and will only use the reads that match the specified sequences (and reset the trim point of those reads so that the multiplex sequence is trimmed off the output read sequence).

  Three examples of multiplex information strings are the following:

  mid2@myreads.sff
  GSMIDs:mid1,mid4,mid8@/home/xxx/morereads.sff
  aattctc/1@1-3,4,6-8:D_2005_01_01_01_01_01_testuser_runImagePipe

- The first example gives "myreads.sff" as the input SFF file and tells sfffile to only use reads whose initial 5' sequence matches the "mid2" MID (as listed in the MID configuration file). The second example tells sfffile to use the file "/home/xxx/morereads.sff" and filter that file, using only the reads starting with the mid1, mid4 and mid8 sequences, as defined in the "GSMIDs" MID set. The third example tells sfffile to read regions 1, 2, 3, 4, 6, 7 and 8 of the "D_2005_..." sequencing Run, but only use the reads whose 5' end matches the DNA sequence "AATTCTC" with 1 error or less.

  The format of the multiplexing information string is the following:

  ```
  [setname:]mid[/#](,mid[/#]…)@
  ```

  …where

  - the "setname" is an optional name of an MID set found in the MID configuration file and can be given in uppercase or lowercase letters (the matching is case-insensitive);
  - each "mid" is either an MID name found in the configuration file or a DNA sequence; and
  - each "#" is an optional allowed number of errors.

- In other words, the format consists of:
  - An optional MID set name, followed by a colon
    - This name must occur in the MID configuration file
  - One or more MID names or DNA sequences, separated by commas. Each name/sequence can be followed by an optional slash and number of allowed errors.
    - The MID names must occur in the MID configuration file, and if the same MID name occurs in multiple MID sets in the file, then the MID set name must be given. (If the MID name is unique over all the names in the file, then no MID set name is required.)
  - A "@" sign to end the multiplexing information string.

- No spaces are allowed in the multiplexing information string, and no colons, slashes or "@" signs are allowed in the MID set names or MID names.

  The off-instrument installation contains a default MID configuration file, found by default at /usr/local/rig/config/MIDConfig.parse. This file is read by sfffile and used to match MID set names and MID names with their multiplexing information. Users can edit this file to add their own MID sets (following the format and syntax described in the file), or can copy this file to create their own separate MID configuration file (and then use the "-mcf" option to specify that newly created file as the MID configuration file to be used). See section 13.4.15 for the contents of the MIDConfig.parse file.

7.2 sffinfo

The sffinfo command extracts read information from an SFF file, and reports it in text form. This is used for "human" examination of the data, or for generating the FASTA and quality score files of the read sequences, which can be given to standard bioinformatics tools. For an example of this output, see section 13.4.14.

The sffinfo command uses the following syntax:

```
sffinfo [options...] [- | sfffile] [accno...]
```

Command	Description
sffinfo	The sffinfo command extracts read information from an SFF file, and reports it in text form. By default, a text summary of all the read information is output, but the output can be limited to only the sequences, quality scores, flowgrams or manifest. All output is written to standard output.

Option / Argument	Description
-a or -accno	This option limits the output to only the accession numbers.
-s or -seq	This option limits the output to only the sequences.
-q or -qual	This option limits the output to only the quality scores.
-f or -flow	This option limits the output to only the flowgrams.
-t or -tab	The default format for the sequences, quality scores and flowgram is FASTA. This option changes this to tab-delimited lines.
-n or -notrim	By default, the trimmed data is output. The -notrim option changes this to output the untrimmed sequences or quality scores.
-m or -mft	This command does not output the manifest by default. This option outputs the manifest text, if it exists in the SFF file. In this case, the -tab and -notrim options and the accnos on the command line are ignored.
[--help]	Displays a usage line and short description of the command.
[--version]	Displays the current software version of the command.
- | sfffile	Path to the SFF file whose data will be output. If this is replaced by a "-" on the command line, then standard input is read for the SFF contents.
accno...	List of the accession numbers of the reads whose data will be output. If no accnos are given on the command line, the information from all reads in the file are output. If an SFF file is specified and it contains an sfffile-created index, then an index-based lookup is used and the reads are output in the order they appear on the command line. If not, the complete SFF file is scanned and the reads are output in the order they appear in the file.

Only one of -accno, -seq, -qual, -flow or -mft may be specified: the program uses the last one on the command line.

7.3 sff2scf

The sff2scf command converts the flowgram information from a read's SFF entry into an SCF file, creating a synthetic "trace" for the read. In this software version, it has been written mainly for program-triggered execution by the consed software, and minimal support is provided for direct command line use. (The main reasons for this are that the command converts a 1000 byte SFF entry into a 45,000 byte SCF file, and that the synthetic trace generated cannot be used by other software, like polyphred, that expect a trace from a Sanger read.)

The sff2scf command uses the following syntax:

```
sff2scf locationstring [outputfile]
```

Command	Description
sff2scf	The sff2scf command extracts one read's information from an SFF file and converts it into an SCF file (or performs "callthroughs" to access SCF data for Sanger reads).

Option / Argument	Description
[--help]	Displays a usage line and short description of the command.
[--version]	Displays the current software version of the command.
locationstring	A "locationstring" that tells sff2scf what path or what command to use to access the read's trace information. See below for a description of the format of this string.
outputfile	The path to where the SCF file should be written. If no outputfile is given, the output is written to "/tmp/*locationstring*" (the location that consed expects to read the data).

The locationstring argument can take one of 4 forms, which allow the sff2scf command to fully support accessing SFF and SCF data for the reads that may appear in an assembly or mapping output from the Genome Sequencer System. The forms are the following:

1. If the locationstring begins with "sff:", then it specifies an SFF file and read accession to get the SFF data, and will cause the generation of a synthetic SCF trace file. The format of the locationstring should be either "sff:*path*:*accno*" or "sff:-f:*path*:*accno*" and is processed using the following rules:
   a. Any instance of the string ":_:" in the path will be translated into the character "/" (so that the locationstring can be used as a simple filename when executed by consed, even though it encodes a path through the directory structure).
   b. If the path begins with the character "/" (after rule a), it is treated as the absolute path to the SFF file.
   c. If the path does not begin with "/", both "path" and "../sff_dir/path" are checked for the existence of a file (the "../sff_dir/path" format is the form used by the GS *De Novo* Assembler and the GS Reference Mapper applications, when generating a full consed directory structure).
   d. The command "sffinfo truepath accno" is executed to access the read's SFF data (where "truepath" is the path determined by rules b and c).
   e. If the "-f" string appears immediately following "sff:", the read's flowgram is reverse-complemented (or "flipped") prior to the generation of the SCF file. (This occurs when the read is the left half of a 454 Paired End read. The consed software (and other software) assume a directionality of Paired End reads, where the clone that was used to generate the Paired End read is assumed to extend off the 3' end of both reads in the pair. In the processing of 454 Paired End reads, this is not the case for the left half of the Paired End read (the clone extends from the 5' end of this half). To support consed and the standard view of Paired End reads, the GS *De Novo* Assembler reverse-complements all left halves of paired end reads, and so this command must support the reverse-complementing in the generation of the SCF trace).

▶▶▶

2. If the locationstring begins with “file:”, it is treated as specifying the path to an SCF file, of the form “file:*path*”, and is processed using the following rules:
    a. Any instance of the string “:_:” in the path will be translated into the character “/” (so that the locationstring can be used as a simple filename when executed by consed, even though it encodes a path through the directory structure).
    b. If the path begins with the character “/” (after rule a), it is treated as the absolute path to the SCF file.
    c. If the path does not begin with “/”, both “path” and “../chromat_dir/path” are checked for the existence of a file.
    d. The bytes of the file (determined by rules b and c) are copied into the output file without change.
3. If the locationstring begins with “cmd:” it is treated as specifying a command string to be executed to access the SCF contents, is assumed to have the form “cmd:*commandline*” and is processed using the following rules:
    a. Any instance of the string “:_:” in the commandline will be translated into the character “/” (so that the locationstring can be used as a simple filename when executed by consed, even though it encodes a path through the directory structure)
    b. After rule a, any colon “:” will be translated into a space “ ” (so that the location-string can be used as a CHROMAT string, which cannot contain whitespace, in the ACE and PHD files used by consed (and will be the strings passed to sff2scf file by consed)).
    c. The resulting commandline string is executed using the proper command, and the standard output of the command is copied into the output file without change.
4. Otherwise, the locationstring is treated as a path to an SCF file, and processed using the following rules:
    a. If the path begins with “/”, that is treated as the absolute path to the file
    b. If the path does not begin with “/”, both “path” and “../chromat_dir/path” are checked for the existence of a file, where the “../chromat_dir/path” form exists to support the addition of new reads when running consed itself (the default operation of consed, looking in the ../chromat_dir directory to find the SCF file for a read).
    c. The bytes of the file (as determined by rules a and b) are copied into the output file.

The reason for the somewhat arcane syntax of the location strings is that when sff2scf is used within consed, (1) it is responsible for accessing the traces for all the reads in the assembly (454 Sequencing reads with SFF data, Sanger reads with SCF files, or commands that generate SCF data); (2) it is only given a single argument (the “CHROMAT” string in the ACE file or PHD file, which cannot contain whitespace) and must produce a file named by that same argument, and (3) that argument string must be a valid simple filename (not a path or command), even if the source of the data is a full path or command string.

7.4 fnafile

The fnafile command provides similar functionality for FASTA files as the sfffile command provides for SFF files, and in addition provides a mechanism to easily generate and add annotation strings to the description lines of reads in the file. The command constructs a single FASTA file containing the reads from a list of FASTA, PHD or SCF files, or directories containing FASTA, PHD or SCF files. The reads written to the new FASTA file can be filtered using inclusion and exclusion lists of accession numbers, and their sequence trimming points or annotation strings can be adjusted. This command is used to pool the results of many Sanger reads into a single FASTA file, to simplify further handling of the combined data.

The fnafile command uses the following syntax:

```
fnafile [options...]
    (fastafile or PHDfile or SCFfile or directory)...
```

Command	**Description**
fnafile	Constructs a single FASTA file (plus associated quality score file) containing the reads from a list of FASTA files (possibly with associated quality score files), PHD files, SCF files or directories containing FASTA, PHD or SCF files.

Option / Argument	**Description**
-o output	The default output of the fnafile program is a single file containing all the reads of the files given on the command line, output to the file “Reads.fna” (and “Reads.qual”, if quality scores occur in the input). The “-o” option allows the user to specify a different filename.
-i accnofile	By default, all reads in the inputs are written to the output file. If the “-i” (Include only these reads in output) or “-e” (Exclude these reads from output) options are used, the accession numbers given in the files specified will change the reads that are output. (If both -i and -e are used, the output will include the reads that are in the -i file and not in the -e file.) Those files should list the accession numbers one per line, where the accession number should be the first word on the line after an optional ‘>’ (*i.e.*, if the line begins with ‘>’, that character is skipped, and the following characters up to the first whitespace character is read as the accession number). These options are cumulative, *i.e.* if multiple -i options are given, the reads included will be the union of the -i accession lists.
-e accnofile	

▶▶▶

Option / Argument	Description
-t trimfile or -tr trimfile	These two options adjust the trim points for some or all reads in the output file. The specified “trimfile” should contain one or more lines consisting of (1) a read accession number, (2) a starting trimpoint and (3) an ending trimpoint, separated by whitespace characters or where the trimpoints are separate by a dash (*e.g.*, “accno 12 543” or “accno 12-543”). The trimpoint values are 1-based position values that denote the first and last base of the trimmed region (*e.g.* for a read 800 bases in length, the lines above specify that bases 1-11 and 544-800 should be ignored, and bases 12-543 form the trimmed region of the read). A value of 0 specifies that the beginning or end of the read should be used (*e.g.* for a read 800 bases in length, the line “accno 12 0” sets the trimmed region to 12-800). The –t option will merge the given trimpoints with any existing trim points for the input read, writing the largest starting trimpoint and smallest ending trimpoint into the output. By contrast, the –tr option will “reset” the trimpoints, using only the trimpoint information occurring in this file. Note: The use of this option does not limit the reads that will be written into the output file. To only output the reads whose accessions appear in the trim file, the –i option must also be used, with the same file as its argument.
-af filename	This option specifies a file that contains adjustments to the description line for some or all reads. Each line should contain the adjustments for a single read, and the line must begin with the accession number of the read (preceded by an optional ‘>’ character). The text that follows the accession number (and subsequent whitespace characters) can take one of two forms. ▶ If the text to the right of the accession number does not begin with a ‘>’, it is appended to the description line for the read. For example, the line `DJS045A03F template=DJS045A03 dir=F library=DJS045` …will append the text “template=DJS045A03 dir=F library=DJS045” to the end of the current description line for read “DJS045A03F”. ▶ If the text to the right of the accession number does begins with a ‘>’, it will completely replace the description line for the input read, *including changing the accession number for the read*. For example, the line `gi|tl|300103402131 >DJS045A03F template=DJS045A03...` …will change the description line for input read “gi|tl|00103402131” to be “>DJS045A03F template=DJS045A03…” in the output FASTA file that is written (and effectively change the accession number of the read for any program which uses the output FASTA file). This feature is useful for translating less descriptive read accession numbers (such as the NCBI Trace Archive accessions) into more descriptive read accession numbers (such as the genome project accession numbers, which encode library, plate-well and direction in the accession numbers).

▶▶▶

Option / Argument	Description
-ac command	This option specifies a command to be executed for each input read, to determine adjustments to make to the read's description line in the output FASTA file. The fnafile command will execute the command string "*command accno* '*description*'" for each read, where "*command*" is the value of the –ac option, "*accno*" is the first non-whitespace string on the input read's description line, and "*description*" is the input read's full description line (including the accno, but removing the first character '>'). (Note the use of quotes in the command string, meaning that the command will get exactly two command line arguments, where the first is the accno and the second is the full description line, even if the description contains whitespace.) This command should process its two arguments, then write up to one line of output to standard output. That line of output should be formatted as described above for the "text after the accession" for the –af option and will be processed by the fnafile command the same way. If no line of output is written, the input read's description line will remain unchanged.
[--help]	Displays a usage line and short description of the command.
[--version]	Displays the current software version of the command.
fastafile or PHD file or SCF file or directory	Path to a FASTA file, PHD file, SCF file or a directory containing FASTA, PHD or SCF files. The reads in any input files are merged directly into the single output FASTA file generated by this command. If a directory is given, all the files it contains are read, and any files in the directory that are FASTA, PHD or SCF are processed and their reads are included in the output. Note: This command is *not* aware of consed's phd_dir versioning of reads. If a phd_dir directory is given that contains multiple versions of the PHD file for a single read, all versions will be read and used by the command, and multiple versions of those reads will appear in the output FASTA file. The same holds true if a phd.ball file and individual PHD files are given on the command line.

7.5 sffrescore

The sffrescore command can be used to rewrite existing SFF files with the new Phred-like quality scores. It recursively searches through a list of files or directories, identifies all of the SFF files, determines which SFF files contain the older quality scores, runs the sfffile command (with the –r option) to generate a new SFF file with the new quality scores, then overwrites the existing SFF file with the new file. The effect is that the reads, flowgrams and basecalls in the files are left unchanged, but the files now contain new quality scores.

The sffrescore command uses the following syntax:

```
sffrescore [-f] (file | directory)...
```

Command	Description
sffrescore	The sffrescore command recursively searches through a list of files or directories, and rescores any SFF file that contains the older quality scores. Any file rescored is rewritten in place with a new version of the SFF file.

Option / Argument	Description
-f	This option forces the rescoring of all SFF files (by default, any SFF file found to already have the new scores are not rewritten).
file or directory...	List of the files and directories to be processed. The command will recursively search through the directories to find all of the SFF files in the directory trees.

8. GS Run Browser Application

8.1 Introduction to the GS Run Browser

The GS Run Browser software is an interactive application that allows the user to view and analyze the results of a sequencing Run performed on a Genome Sequencer FLX Instrument, including graphic representations of the contents of various metrics files. It can be used to assess the general quality of a Run, and for troubleshooting in case problems are observed. Its display of filtering results also offer a convenient way to assess the results of library titrations (full sequencing titration option; see the *GS FLX Titanium General DNA Library Preparation Method Manual*).

The application includes graphic interfaces to view:

- upper-level information about the Run
- the raw images from the Run
- the locations and status of all identified active wells, well density information for raw wells and keypass wells, and CAFIE correction data
- the raw or subtracted flowgram for selected position(s) on the images; or fully processed flowgrams for all the data-generating wells detected during the Run
- raw signal statistics for the sample library or the Control DNA reads
- read length and quality statistics for the sample library or the Control DNA reads
- consensus flowgrams and accuracy metrics for the Control DNA sequence reads
- statistics on the results of the quality filtering algorithms

The GS Run Browser application is written entirely in Java, and requires Java version 1.5. Java 1.5 is included as part of the standard software package on the Linux-based computer onboard the Genome Sequencer FLX Instrument, as well as of the off-instrument software package to be installed on your DataRig(s) or clusters.

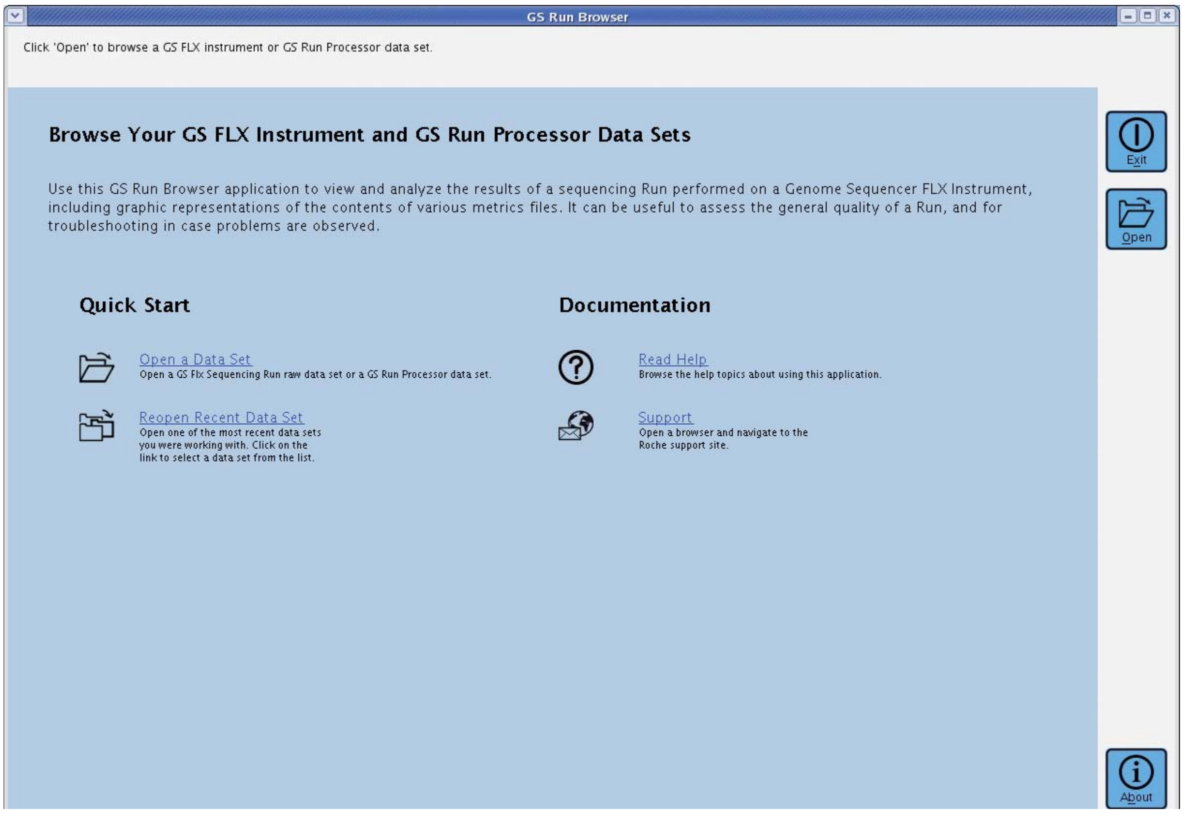
8.1.1 Launching the GS Run Browser Application

The GS Run Browser application consists of a single command, whose command line structure is the following:

```
gsRunBrowser [dir]
```

This will open the GS Run Browser splash screen which will persist for a few seconds (not shown), and the GS Run Browser main window. If the (optional) path to a valid data set is given on the command line, `[dir]`, that data set will be opened directly (see below). Otherwise, an entry window will open (Figure 8–1), ready to load a sequencing Run. This entry window comprises:

- a brief description of the purpose of the program;
- a menu area with two "Text Links" for opening data sets, and two "Text Links" for help and support;
- a vertical toolbar with four main function buttons, on the right-hand edge; and
- a status area along its top edge.



Click 'Open' to browse a GS FLX instrument or GS Run Processor data set.

Browse Your GS FLX Instrument and GS Run Processor Data Sets

Use this GS Run Browser application to view and analyze the results of a sequencing Run performed on a Genome Sequencer FLX Instrument, including graphic representations of the contents of various metrics files. It can be useful to assess the general quality of a Run, and for troubleshooting in case problems are observed.

Quick Start

Open a Data Set
Open a GS Flx Sequencing Run raw data set or a GS Run Processor data set.

Reopen Recent Data Set
Open one of the most recent data sets you were working with. Click on the link to select a data set from the list.

Documentation

Read Help
Browse the help topics about using this application.

Support
Open a browser and navigate to the Roche support site.

Figure 8–1: The GS Run Browser main window, just after launching the application, when no data set is specified on the command line

Click on the "Open" button or on the "Open a Data Set" Text Link, and the "Open a Data Set" dialog window will open, prompting you to select a data set to display (Figure 8–2). This can be an 'R_' (Run) or a 'D_' (Data Processing) directory of the sequencing Run of interest (see section 1.3 for more details on the folder structure of sequencing Run results). The Open button of the Open a Data Set is enabled only when a valid data set has been selected. Data sets that are valid for loading are identified with the following icons:

- R_ directory: this will load only the general Run information and the captured images
- D_ directory: This will load the general Run information, the captured images, and all the data processing results

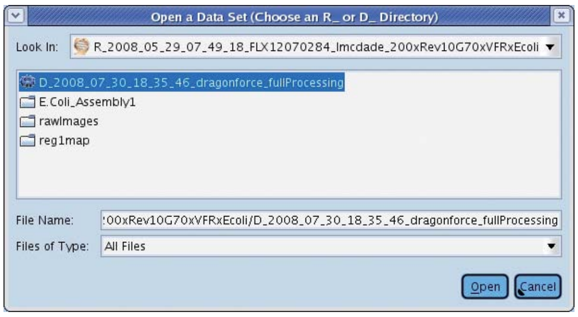
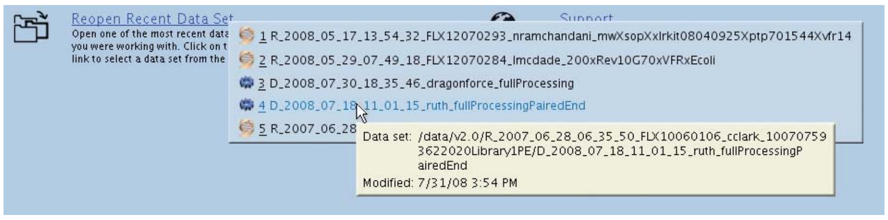

Open a Data Set (Choose an R_ or D_ Directory)
Look In:
R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli
D_2008_07_30_18_35_46_dragonforce_fullProcessing
E.Coli_Assembly1
rawImages
reg1map
File Name:
:00xRev10G70xVFRxEcoli/D_2008_07_30_18_35_46_dragonforce_fullProcessing
Files of Type:
All Files
Open
Cancel

Figure 8–2: The "Open a Data Set" dialog window, used to select the dataset to display in the GS Run Browser application

A progress bar will appear at the upper-right corner of the GS Run Processor window while the data set is being loaded, showing the step of the loading. If the data set you select fails to open, an error message window will appear (not shown). You can click OK in the error window, and then select a valid data set. If the data set you selected is indeed the one you want to open in the GS Run Browser, and it is not recognized as a valid data set, contact your Roche representative; have the specific content of the error message available if you call for help, as it may help troubleshooting the error.

Finally, if you want to open a data set that was viewed recently, click the "Reopen Recent Data Set" Text Link, or click the Open button while pressing the Shift key: a list of data sets recently opened in the GS Run Browser will appear in a pop up menu (Figure 8–3); then, select the data set of interest.


Reopen Recent Data Set
Open one of the most recent data
you were working with. Click on t
link to select a data set from the
Support
1 R_2008_05_17_13_54_32_FLX12070293_nramchandani_mwXsopXxlrkit08040925Xptp701544Xvfr14
2 R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli
3 D_2008_07_30_18_35_46_dragonforce_fullProcessing
4 D_2008_07_18_11_01_15_ruth_fullProcessingPairedEnd
5 R_2007_06_28

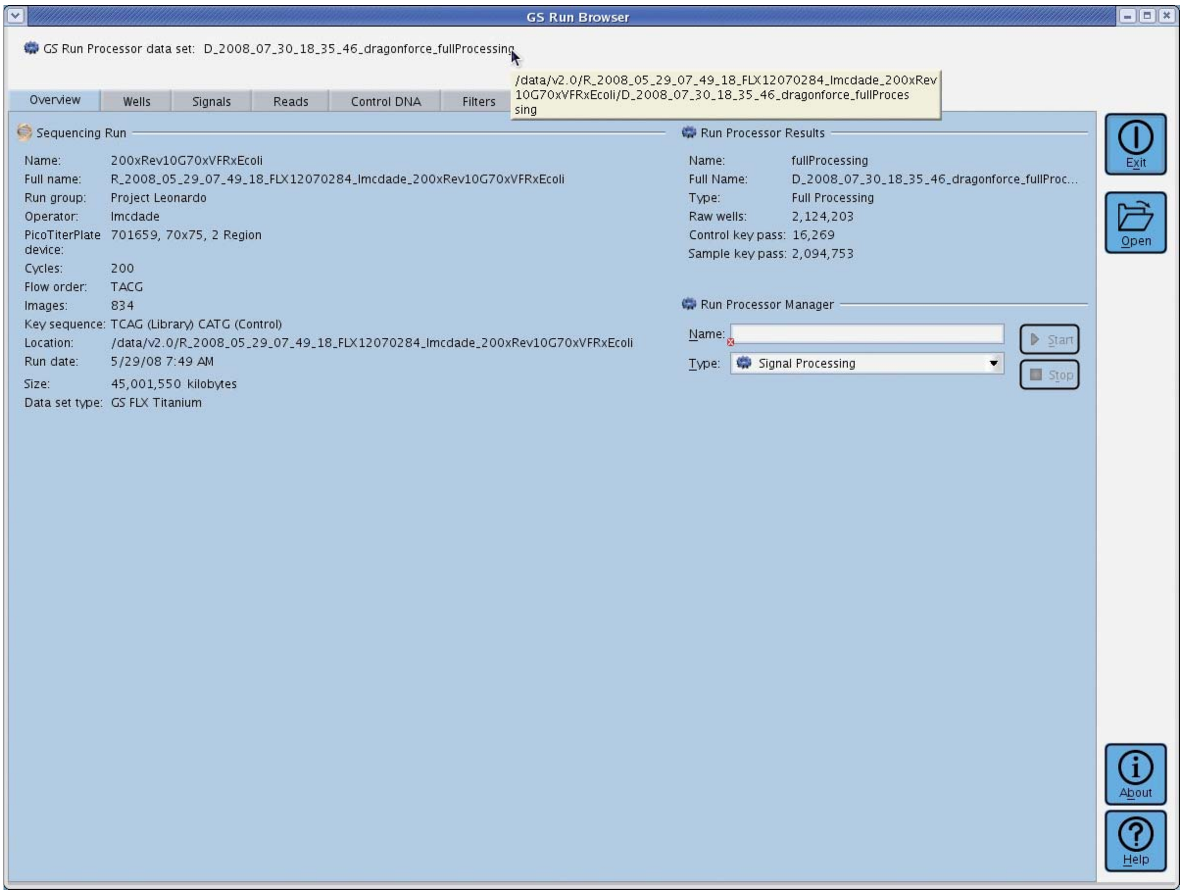
Data set: /data/v2.0/R_2007_06_28_06_35_50_FLX10060106_cclark_10070759 3622020Library1PE/D_2008_07_18_11_01_15_ruth_fullProcessingP airedEnd
Modified: 7/31/08 3:54 PM

Figure 8–3: Pop up list of recent data sets, ready to reopen

8.1.2 GS Run Browser Interface Overview

After a dataset is loaded from a sequencing Run, it will be displayed in the GS Run Browser window (Figure 8–4). The browser comprises three general areas:

- a main tool bar located on the right hand side of the window;
- a status area located at the top of the window, showing the full name of the data set currently displayed, *i.e.* an 'R_' or a 'D_' directory name; this area also displays a loading progress bar with milestone text while the data set is being loaded;
- and a series of tabs to display various aspects of the information contained in the data set.



GS Run Processor data set: D_2008_07_30_18_35_46_dragonforce_fullProcessing

/data/v2.0/R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/D_2008_07_30_18_35_46_dragonforce_fullProcessing

Sequencing Run

Name:	200xRev10G70xVFRxEcoli
Full name:	R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli
Run group:	Project Leonardo
Operator:	lmcdade
PicoTiterPlate device:	701659, 70x75, 2 Region
Cycles:	200
Flow order:	TACG
Images:	834
Key sequence:	TCAG (Library) CATG (Control)
Location:	/data/v2.0/R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli
Run date:	5/29/08 7:49 AM
Size:	45,001,550 kilobytes
Data set type:	GS FLX Titanium

Run Processor Results

Name:	fullProcessing
Full Name:	D_2008_07_30_18_35_46_dragonforce_fullProc...
Type:	Full Processing
Raw wells:	2,124,203
Control key pass:	16,269
Sample key pass:	2,094,753

Run Processor Manager

Name:	
Type:	Signal Processing

Figure 8–4: General view of the GS Run Browser window. Most application screenshots shown in this section are from this *E. coli* library sequencing Run.

8.1.2.1 The Main Buttons

Four main buttons are always available in the tool bar, along the right-hand side of the GS Run Browser window:

- The **Exit** button closes the application.
- The **Open** button allows you to browse your network to find a data set to open. Clicking this button while pressing the Shift key evokes a pop up menu allowing you to choose from a list of data sets recently opened with the GS Run Browser.
- The **About** button opens a dialog window providing some information about the GS Run Browser application (not shown). Click the Close button to close it when you are done.
- The **Help** button will provide online access to help topics, organized by results tab. (Not currently implemented.)

The top of the main GS Run Browser window also always shows the name of the Run or Data Processing directory) for the data set being displayed. The full path to the directory is displayed in a tooltip that appears when you pause the mouse over its name.

8.1.2.2 The Tabs

The GS Run Browser application displays the various aspects of the sequencing Run results in a series of 6 tabs. Clicking on a tab brings the information it contains to the front for viewing, as listed below.

- The **Overview** tab is divided into three sections:
  - The section on the left provides a basic summary of the sequencing Run data: directories, type of PicoTiterPlate device number of nucleotide cycles and of images, *etc.*
  - The section on the top-right shows the GS Run Processor results, if available for the currently open sequencing Run
  - The bottom-right section is the Run Processor Manager interface, which allows the user to launch a new analysis of the currently open sequencing Run.
- The **Wells** tab displays the raw images captured during the sequencing Run, plus overlaid colored circles showing where the data-generating wells identified in the Run are located. Several coloring criteria and well statistics are available on the tab. Clicking on the picture outside any wells can open a raw flowgram view of the location where you clicked, and clicking on a well can open the processed well's flowgram.
- The **Signals** tab provides statistics on raw signal intensity and homopolymer length distributions across all the wells of the PicoTiterPlate device for any individual flow, for either sample library or Control DNA reads.
- The **Reads** tab provides statistics on read length and quality score results, for either sample library or Control DNA reads.
- The **Control DNA** tab displays the accuracy metrics for the Control DNA reads (% match to their reference sequences) and other Control DNA information.
- The **Filter** tab shows the quality filter information for the Run, including the number of wells that failed each filter and the number that passed all filters, for either sample library or Control DNA reads. An option allows the older data set (processed with version 1.1.03 or earlier of the GS data processing software) to be evaluated as a titration experiment.

8.1.2.3 Buttons and Plots

The images, plots and data tables displayed in the various tabs of the GS Run Browser application are scrollable and/or zoomable graphical elements. They share certain common buttons and functions, *e.g.* to perform the scrolling and zooming. When they do appear, these graphic elements have some or all of the following features (see in Figure 8–5, an example for a well flowgram window which has most of these elements):

- Scroll bars for horizontal and/or vertical scrolling (appearing below and to the right of the element, if necessary).
- A column of buttons along the upper left edge of the graphic elements, used for navigation (including various zooming functions) and/or to save snapshot images or text files of the displayed data. For the image area of the Wells tab, this is replaced by a unique set of controls that are overlaid near the upper-left corner of the image.
- "Mousing" functions (pointing, clicking or dragging the mouse, touchpad, pen, *etc.* over the graphical element) to view data values and adjust the zoom level.

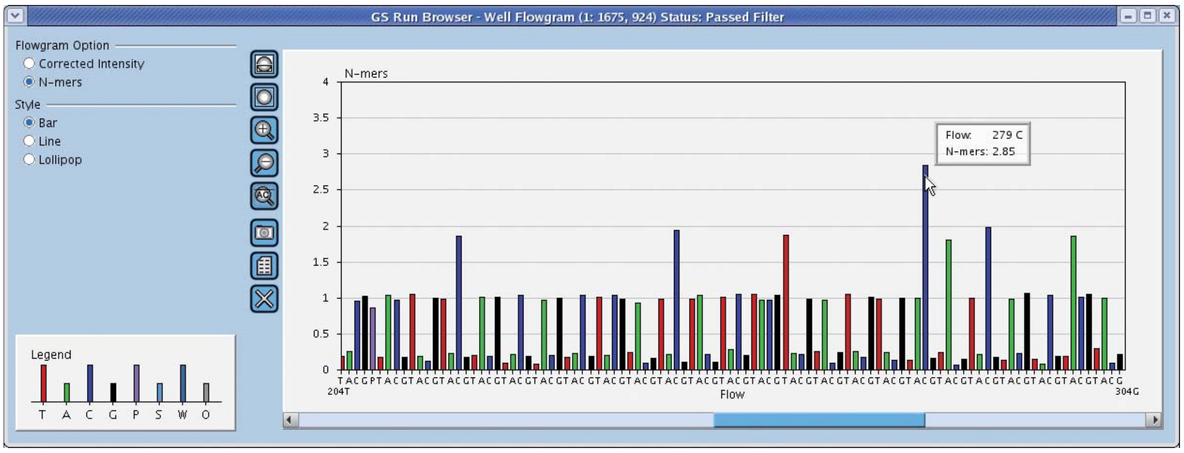

GS Run Browser - Well Flowgram (1: 1675, 924) Status: Passed Filter
Flowgram Option
Corrected Intensity
N-mers
Style
Bar
Line
Lollipop
Legend
T A C G P S W O
N-mers
4
3.5
3
2.5
2
1.5
1
0.5
0
Flow: 279 C
N-mers: 2.85
TACGPTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACG
204T
Flow
304G

Figure 8–5: An example well flowgram window showing many of the common graphical element functions

8.1.2.3.1 Scroll Bars

The scroll bars have the standard functions, and appear when the data in either direction is too large to be shown in the viewable area of the computer screen. In Figure 8–5, for example, only the horizontal scrollbar appears, because the y-axis scale is large enough to show the full height of the flowgram signal bars.

8.1.2.3.2 Navigation Buttons

The buttons appearing to the left of an element have the following general functions, although their specific meaning is in some cases adjusted to the context of the element:

Zoom to fit data – Fit means "zoom all the way out." On plots, scale out to the limits of the data. The y-axis is rounded to an attractive-looking number rather than stopping at the exact data limit.

Zoom to default size – Some plots default to a zoom level different than "fit". For those plots, this button resets the zoom level to the default initially shown.

Zoom in – Zoom in by a factor of 1.5. On image displays, this will zoom in by this factor, centered on the middle of the display. For plots, this button zooms only the y-axis scale (use the **Zoom to labels** and **Freehand zooming** functions described below to zoom the x-axis).

Zoom out – Zoom out by a factor of 1.5. On image displays, this will zoom out by this factor, centered on the middle of the display. For plots, this will zoom only the y-axis scale and, unlike most zoom operations, this will zoom out past the data limits (to allow you to get a better perspective of the data).

Zoom to labels – This button only appears in the flowgram viewers; it zooms the x-axis of the flowgram so that the nucleotide/flow characters can fit below the axis.

Snapshot – Save a snapshot image of the current view to disk. This will open a dialog asking for a location and filename, and then will save a PNG image file at that location. For Summary data tables, this saves the whole tabular area, while for all other displays the saved image contains only the visible region of the element.

Spreadsheet – Save an Excel-compatible, tab-delimited text file of the spreadsheet data for this graphic. This will open a dialog asking for the location and filename to save the file. It then saves the data, along with summary information describing where the data came from, as shown in this file snippet saved from a well flowgram:

```
GS Run Browser - Well Flowgram (1: 1781, 2107) Status: Passed
Filter Corrected Intensity vs. Flow
GS Run Browser, Wed Aug 20 09:56:17 EDT 2008
Run
Directory=/data/flx/R_2007_02_08_17_02_43_build04_
mcorso_100x2575xecoli360k
Processor
Directory=/data/flx/R_2007_02_08_17_02_43_build04_mcorso_100x2575
xecoli360k/D_2008_07_31_11_11_55_enya_signalProcessing

Flow  Corrected Intensity
5 PPI 3,696
8 T   511.5
9 A   112.69
10 C  491.5
11 G  150.12
12 T  119.69
13 A  427.75
...
```

For plots, the spreadsheet file contains just the data currently being plotted. For a tri-flowgram view (see sections 8.3.5 and 8.6.4), the data for all three plots is saved to one file, with some white space between plots. For Summary data tables, the spreadsheet contains the table data. If a plot and a summary present the same data, there may be only one spreadsheet button, attached to whichever has the superset of the data.

Close – Close a flowgram window. This does the same thing as the operating system window close icon (the “x” icon in the upper right corner of the title bar of the window), but may be in a more convenient place.

The image area of the Wells tab has special navigation controls described in section 8.3.3 item 3.g.

8.1.2.3.3 Mousing Functions

When the mouse cursor is located over a graphical element, additional functions can be performed by moving, clicking or dragging the mouse:

Mouse Tracker – Whenever the cursor is located on data shown in an image or plot, the numeric data values for that data is shown in a related Mouse Tracker window that pops up near the location of the cursor. This allows you to see the specific numeric value for any data point (see example in Figure 8–5).

Drag Image – If the displayed image is larger than the viewing frame on screen, the mouse can be used to drag the image within the frame. Left-click on the image and drag the mouse in the direction needed, to view the part of the image that was just out of the frame (like sliding a piece of paper on top of a desk).

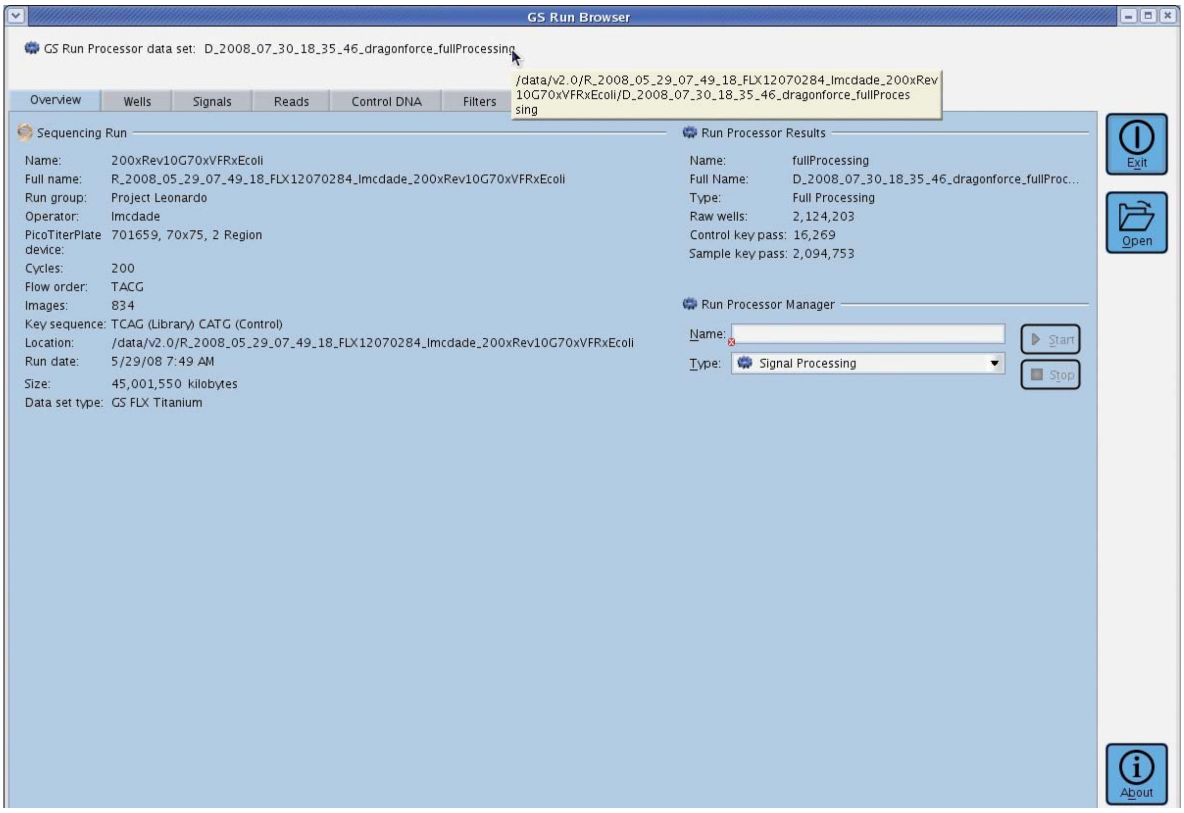
Freehand Zoom In – The mouse can be used to zoom in on specific regions of the image or of a plot. To zoom in on a plot, hold the left mouse button down and drag a box around the area of interest (not shown). Releasing the button zooms to that region. On the image area of the Wells tab, this function is accomplished with the right mouse button rather than the left, because of the “left click and drag” function just described.

Freehand Zoom Out – For plots only, right-clicking the plot will cause the plot to zoom out by a factor of 1.5 in both the X and Y directions, centered on the middle of the current view. This zoom will not zoom farther than the limits of the data.

8.2 The Overview Tab

8.2.1 General Description

The Overview tab contains summary information about the sequencing Run selected (Figure 8–6). It is the default tab shown after a data set has been selected for viewing in the GS Run Browser.



GS Run Processor data set: D_2008_07_30_18_35_46_dragonforce_fullProcessing

/data/v2.0/R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/D_2008_07_30_18_35_46_dragonforce_fullProcessing

Sequencing Run

Name:	200xRev10G70xVFRxEcoli
Full name:	R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli
Run group:	Project Leonardo
Operator:	lmcdade
PicoTiterPlate device:	701659, 70x75, 2 Region
Cycles:	200
Flow order:	TACG
Images:	834
Key sequence:	TCAG (Library) CATG (Control)
Location:	/data/v2.0/R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli
Run date:	5/29/08 7:49 AM
Size:	45,001,550 kilobytes
Data set type:	GS FLX Titanium

Run Processor Results

Name:	fullProcessing
Full Name:	D_2008_07_30_18_35_46_dragonforce_fullProc...
Type:	Full Processing
Raw wells:	2,124,203
Control key pass:	16,269
Sample key pass:	2,094,753

Run Processor Manager

Name:	
Type:	Signal Processing

Figure 8–6: The GS Run Browser application's Overview tab after loading a dataset

8.2.2 Source of the Data

The Overview tab is always created when a Run or a Data Processing directory is opened. The particular data fields will show the values from the sequencing Run if their source data is as follows (see section 1.3).

- If you open an R_ directory, the data always comes from the dataRunParams.parse file present in it; the Run Processor Results area of the Overview tab will contain no data and will be grayed out. (Also, the Wells tab will display only the images, and all the other tabs will be unavailable.)
- If you open a D_ directory, then
  - For sequencing Runs processed with the GS Run Processor version 2.0.00 or later, the information about the sequencing Run (left side of the tab) and about the processed data set (top-right of the tab) comes from the region.cwf files.
  - For sequencing Runs processed with software version 1.1.03 or earlier:
    - the information about the sequencing Run (left side of the tab), comes from the following files:
      - dataRunParams.parse
      - imageLog.parse
    - the information about the processed data set (top-right of the tab), comes from the files present in the D_ directory.

8.2.3 Features and Functionalities

The Overview tab contains the following areas, with the information and functionalities described (see Figure 8–6):

1 The **Sequencing Run** section shows the following basic information about the sequencing Run whose results are currently displayed:

a. Name: the short, unique name of the sequencing Run, given in the Run Wizard at the time of Run set up
b. Full name: the full name of the Run directory (starting with 'R_')
c. Run group: the Run group assigned to the sequencing Run, in the Run Wizard at the time of Run set up
d. Operator: the Operator who was logged in at the time the sequencing Run was started
e. PicoTiterPlate device: the size and the number of loading regions on the PicoTiter-Plate device used for the sequencing Run
f. Cycles: the number of nucleotide flow cycles executed by the sequencing Run script
g. Flow order: the order of the nucleotides in each cycle
h. Images: the number of images captured during the sequencing Run
i. Key sequence: the key sequences for the Sample and, if applicable, Control DNA reads used for the sequencing Run
j. Location: full path to the location of the 'R_' directory for the sequencing Run
k. Run date: date and time at which the sequencing Run was started
l. Size: size of the 'R_' directory for the sequencing Run, including all sub-directories and their contents, in kilobytes
m. Data set type: either GS20, GS FLX Standard or GS FLX Titanium. This is normally a static field, but when the type of a data set cannot be determined (GS20 or GS FLX standard), this is a drop down menu, allowing the user to specify the dataset type; the flow labeling in the Wells tab and the Location flowgrams (and Well flowgrams, if *region*.wells files are absent) will be adjusted accordingly.

2 The **Run Processor Results** section shows the following information about the processed results of the data set currently displayed (only for D_ directories):

a. Name: the short, unique name of the data processing data set, *e.g.* "fullProcessing", or any name entered in the "Name" field of the Run Processor Manager (see below)
b. Full name: the full name of the Data Processing directory (starting with 'D_')
c. Type: the type of data processing, *e.g.* Full Processing, Full Processing for Amplicons, *etc.*
d. Raw wells: the total number of raw wells found in the data set
e. Control key pass: the number of key pass wells with the Control DNA key, found in the data set
f. Sample key pass: the number of key pass wells with the Sample key, found in the data set

3 The **Run Processor Manager** section allows you to re-process the data set currently displayed. If the GS Run Processor application is not currently available, the GS Processor Manager is disabled (grayed out), and the statement "No Run Processor Manager available" appears. This may happen if the Run Processor Manager is not currently running on the same machine where GS Run Browser was started. The Run Processor Manager has the following features:

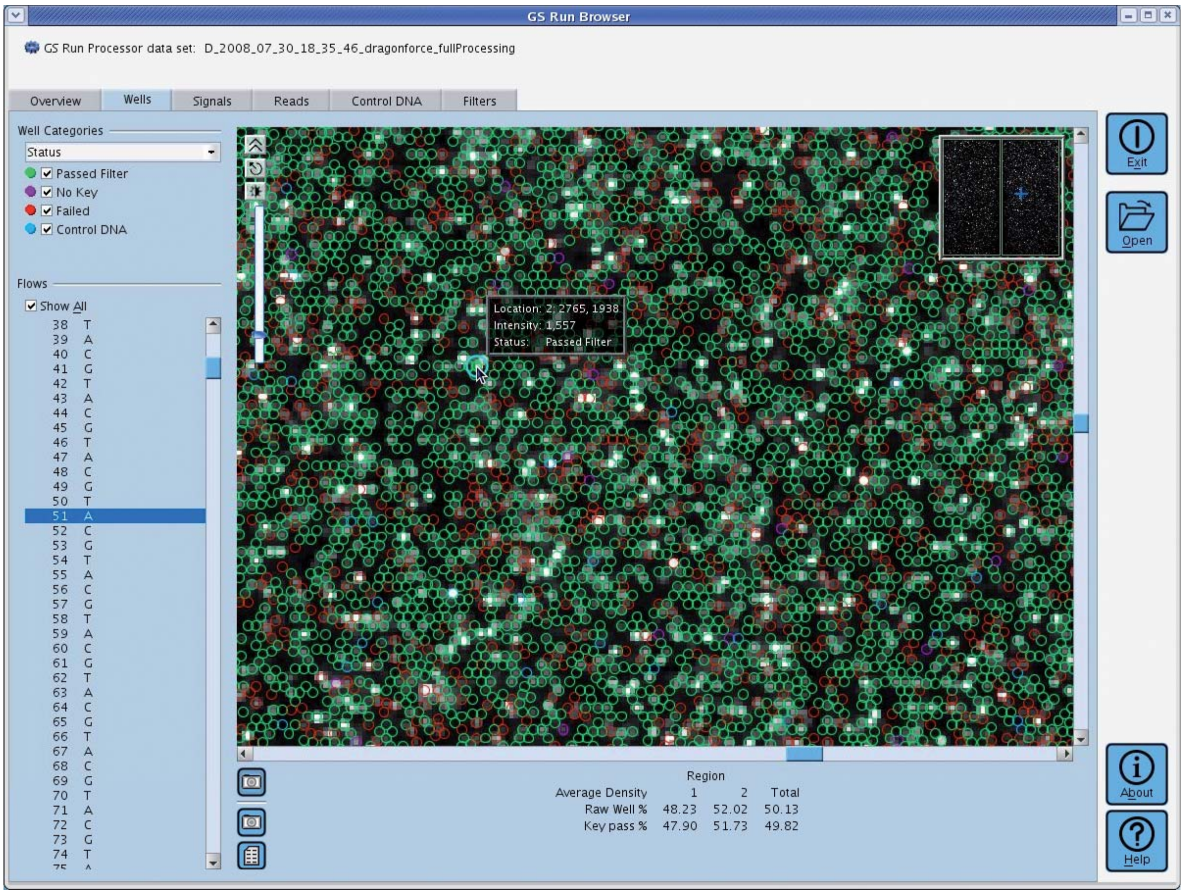
a. Name field: used to enter a unique name for the re-processed data set (will be appended to the 'D_' directory name; see section 1.3); shows the name of the re-processed data set when re-processing is in progress
b. Type field: the data processing type to be used for re-processing (see section 1.2) ; shows the type of the re-processed data set when re-processing is in progress
c. Start button: click to start re-processing of the data set. Disabled (grayed out) if the name field is blank or if re-processing of a data set is currently in progress
d. Stop button: click to stop the re-processing of a data set currently under way. Disabled (grayed out) if re-processing of a data set is *not* currently in progress
e. Statement or messages area: various status and error message reported by the Run Processor Manager are displayed in the area located under the Type field

8.3 The Wells Tab

8.3.1 General Description

The Wells tab (Figure 8–7) displays information about the wells identified during data processing; this information is overlaid on the raw images if they are available. The main display features include the following:

- ▶ The wells are shown as colored circles, whereby the color scheme can be used to indicate a number of read type and quality attributes, as described below (section 8.3.3).
- ▶ The fully processed well flowgrams generated by the data processing software can be accessed.
- ▶ Additional well density statistics are displayed for each region of the PicoTiterPlate device and for the whole PicoTiterPlate device.



GS Run Processor data set: D_2008_07_30_18_35_46_dragonforce_fullProcessing

Location: 2: 2765, 1938
Intensity: 1,557
Status: Passed Filter

	Region		
Average Density	1	2	Total
Raw Well %	48.23	52.02	50.13
Key pass %	47.90	51.73	49.82

Figure 8–7: The GS Run Browser application's Wells tab showing the wells identified by the GS Run Processor application, displayed per the well status, with all statuses shown. These are overlaid on the image taken at step 51 of this sequencing Run, an "A" nucleotide flow. Circles that do not appear to contain any signal simply represent wells that did not extend an "A" at this particular flow. Details on the various areas of the tab shown on this figure are provided below.

8.3.2 Source of the Data

For the Wells tab data to be displayed, the *region*.cwf file(s) [or, for data sets processed with software v. 1.1.03 or earlier, the *region*.wells file(s)], located in the "regions" sub-folder of the "D_" data processing folder of the sequencing Run, must exist (see section 1.3). If no *region*.cwf (or *region*.wells) files exist for the Run, the Wells tab will only display the images and the Flows selector, if available.

For the images to be displayed, there must be a rawImages directory (containing at least one .pif file) for the sequencing Run being viewed (see section 1.3). Ideally, there should also be an imageLog.parse file; this file tells the GS Run Browser what reagent was flowed at each step, and allows the Flows selector to display what nucleotide (or wash, or other reagent) was flowed with which image.

Datasets from past sequencing Runs may or may not contain the images, since images are often removed or archived to save disk space. All the tabs in the GS Run Browser will still operate even if no images exist.

8.3.3 Features and Functionalities

The Wells tab contains the areas listed below, with the functionalities described (see Figure 8–7):

1 The **name** (and path, via the tooltip) **of the Run or Data Processing directory** and the **tabs** are common to the GS Run Browser application (see section 8.1.2).

2 The **Options** area (on the left area of the tab):

a. **Well Categories**: Drop down menu used to select the well display category. Each category has its own color chart (see item 2.b, below) to identify the specific attribute of each well, within the category. If the data from one or more regions of the Pico-TiterPlate device is missing, "(Data unavailable)" will appear in the Well Categories drop down menu.

I. **Status**: This colors the wells according to their quality and/or type (Library read vs. Control DNA read).

01. **Passed Filter**: Library read (key is TCAG) that passed all quality filters (may have incurred trimming)

02. **No Key**: Identified as a well (generates signal), but not one with recognizable data

03. **Failed**: Library read (key is TCAG) that failed any of the quality filters. Note that the GS Run Browser does not make pass/fail decisions; it only reads results provided by the quality filtering algorithms of the GS Run Processor application.

04. **Control DNA**: Control DNA read (key is ATGC or CATG, for GS FLX standard and GS FLX Titanium chemistries, respectively), whether or not it passed all quality filters

II. **Control DNA**: Similar to the "Status" category except that it uses separate bright colors for the reads matching each specific Control DNA sequence. In this category, the attribute "CATG – control" (or, for the GS FLX standard chemistry, "ATGC – control") will reveal wells where the appropriate Control DNA key was detected, but the sequence of the read does not match any of the known Control DNA fragments.

III. **Raw Density**: A measure of local well crowding, calculated individually for all identified wells in which a signal is detected.

IV. **Key Pass Density**: Another measure of local well crowding, calculated individually for wells whose initial signals match a known key, *i.e.* excluding the wells with a **No Key** status. For data sets processed with the software version 2.0.00 or later, this option will appear only if the signal processing step of Data Processing was carried out. For versions 1.1.03 or earlier, this will normally show (requires the *region*.wells files in the data set).

V. **Carry Forward**: Level of CAFIE correction applied to each well, due to the detection of carry forward events (see description of these terms in the Glossary of the *Genome Sequencer FLX Operator's Manual*). This option will appear only if the signal processing step of Data Processing was carried out.

VI. **Incomplete Extension**: Level of CAFIE correction applied to each well, due to the detection of incomplete extension events (see description of these terms in the Glossary of the *Genome Sequencer FLX Operator's Manual*). This option will appear only if the signal processing step of Data Processing was carried out.

▸▸▸

2

b. **Color chart**: Located just below the Well categories drop down menu (item 2.a, above), this area allows the user to select the range of well values to display in the main image area, for the category selected.
   I. Checking the box for an attribute (or numerical range) in the list causes wells matching that attribute (or range) to be displayed on the image. If an item in that list is not checked, the corresponding wells are not displayed. Shift-clicking deselects all attributes; or, if all are already deselected, it selects them all. Control-clicking deselects all attributes except the one clicked; or, if you control-click on an attribute that is currently the only one selected, it selects all attributes except the one clicked.
   II. For the categories whose attributes are continuous variables, the color scheme uses the "color wheel" order for the gradation of value ranges.
   III. For the attributes that denote "goodness" or "badness" of the wells, the "good" values are indicated by green hues, and the "bad" values by red hues, when possible. For example, the most desirable values (wells marked green) for well density near the upper range values (above that, turns bluish), but for CAFIE correction the best wells have minimum values.
   IV. The GS Run Browser "remembers" attribute selection for each category, even if you navigate to another tab and then return to the Wells tab.

c. **Flows**: Located below the color chart, the Flows selector allows the user to choose an image to underlay the well circles, in the main image area.
   I. If images are available in the Run dataset, this selector is displayed; it lists the steps of the sequencing Run where an image was taken by the instrument camera. If any image is unavailable in the data set, it is grayed out in the Flows list.
   II. The Flow tags have the following meanings:
      01. SUB: Substrate buffer (background)
      02. WASH: Buffer wash flow (GS 20 and GS FLX standard chemistries only)
      03. Apyrase##: flow used for Apyrase pulse calibration (GS FLX Titanium chemistry only)
      04. PPI: Flow of PPi for the GS 20 and GS FLX standard chemistries; or ATP for the GS FLX Titanium chemistry (all wells)
      05. T, A, C, G: Nucleotide flow
   III. Clicking on a step displays the corresponding image in the main image area (and overview image area).
   IV. After clicking in the Flows selector list, you can use the keyboard arrow keys to navigate quickly up and down the list of images.
   V. When the Show All box is checked, all the images captured during the sequencing Run are listed. If the box is not checked, only the images captured during nucleotide or PPi / ATP flows (*i.e.* excluding SUB and Apyrase flows) are listed.
   VI. The wells themselves exist during all flows, so this selector has no effect on the display and coloring of the well circles. If the background image is a distraction, you can disable the images by deselecting "Show image" in the contextual menu that appears if you right-click on the image (item 3.f.VIII, below).
   VI. If the imageLog.parse file exists in the Run directory, the complete list of steps and reagents flowed during the sequencing Run is shown in the selector. If the file does not exist, a generic list of steps is created assuming the standard "TACG" flow order, and values in the "Flow" column are followed by a "?" to indicate that this is a guess about the reagent flowed.
   VII. If the imageLog.parse file exists in the Run directory, the complete list of steps and reagents flowed during the sequencing Run is shown in the selector. If the file does not exist, neither the images nor this list will be displayed.

▸▸▸

3 **Image area:**

a. This area provides a zoomable, scrollable view of the entire PicoTiterPlate device area, wereby well data is overlaid on the camera image, if available. Overall, the image area comprises four layers, further described below:
   I. The camera image selected, if available; the displayed image is corrected to emphasize the well intensities and deemphasize the background intensities, for better visualization.
   II. Green rectangles delimiting the bead loading regions of the sequencing Run; the top of each region is labeled with the region name.
   III. Wells, per the options selected.
   IV. Display control buttons and overview image

b. Settings in the **Options** area (see item 2, above) affect this display, such that colored circles are displayed at the x,y location of the wells identified during data processing; the **Well Categories** option and the selected range of attribute values (items 2.a and 2.b, above) determine the specific colors displayed and which wells to show, overlaid on the image from the selected **Flow** (item 2.c), if available.

c. Wells remain in the same locations across all flows of a sequencing Run, irrespective of what image is selected for display in the Flow selector (or of whether images are available at all in the dataset).

d. Pausing or moving the mouse slowly over the main image area has the following effects:
   I. The Mouse Tracker box (section 8.1.2.3.3) appears near the pointer, providing information about the image pixel under the pointer and the closest well. This includes the location under the cursor (region number and x,y coordinates), the corresponding pixel signal intensity value (for the flow selected, if images are available), and the attribute value for the closest (highlighted) well, in the category currently selected in the options. Note that setting the image brightness using the White Threshold slider (see item 3.g.IV, below) does not change the intensity value of the pixel, only the brightness at which it is drawn on the screen.
   II. The pointer position is marked by a blue "+" on the overview image area (see item 3.g.VI, below).
   III. The well closest to the pointer is highlighted.

e. Clicking on the image can have the following effects:
   I. A simple left-click selects the closest well for further action. A selected well is marked by an additional light blue circle around it.
   II. A double-click also applies to the closest well. This constructs a **Well Flowgram** (for wells that contain a library bead; TCAG key) or a **Tri-Flowgram** (for wells that contain a Control DNA bead, ATGC or CATG key, for GS FLX standard and GS FLX Titanium chemistries, respectively), and opens the flowgram viewer to display it (see sections 8.3.4 and 8.3.5 for a full description of well flowgrams and well tri-flowgrams, respectively).
   III. A right-click opens the contextual menu described in item 3.f, below.
   IV. A left-click-and-drag slides the image in the direction of the drag, revealing the part of the image that was just out of view (see section 8.1.2.3.3).
   V. A right-click and drag zooms the image to the area circumscribed by the dragging action (see section 8.1.2.3.3).
   VI. If a mouse wheel is available, rolling forward zooms in and rolling backward zooms out the image. The Magnification Slider control (see below) moves along with the mouse wheeling action.

▶▶▶

3

f. Right-clicking on the main image area evokes the contextual menu shown in Figure 8–8.

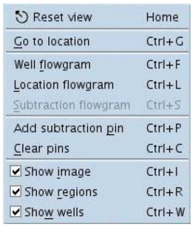


Figure 8–8: Contextual menu evoked by right-clicking the image area of the Wells tab

The contextual menu of the image area offers the following actions:

I. **Reset view**: resets the image to its default size and location, in the top-left corner position. The accelerator key is *Home*.

II. **Go to location**: opens a data entry field just above the image area allowing you to specify a location on the image, for immediate navigation when you press the "Enter" key (Figure 8–9). Valid entries and their meanings are as follows:

01. a single positive integer scrolls the image to the top of the region number specified (*e.g.* 1–16).

02. two positive integers equal to of less than the number of the image pixels scroll the image to that x and y location. The two values can be separated with characters such as a space, comma, period or a slash.

03. a well ID or accession number scrolls the image to that location and selects the well

If the entry is not valid, an icon showing an "x" in a red circle appears at the lower-left corner of the field, and the image location does not change. The accelerator key is *Ctrl-G.*

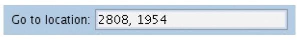

Go to location: 2808, 1954

Figure 8–9: The "Go to location" field, for the Image area of the Wells tab.

III. **Well flowgram**: opens the "Well flowgram" for the closest well. This will be a regular flowgram if the status of the "Passed filter" (or "No key", or "Failed"), and a tri-flowgram for "Control DNA" wells. See sections 8.3.4 and 8.3.5 for a full description of Well flowgrams and Control DNA Tri-flowgrams, respectively. The accelerator key is *Ctrl-F*.

IV. **Location flowgram**: opens the "Location flowgram" for the image pixel on which you clicked. See section 8.3.6 for a full description of Location flowgrams. The accelerator key is *Ctrl-L*.

V. **Subtraction flowgram**: this action requires that a pair of subtraction pins (see below) be first added on the image. Then, selecting this menu item opens the "Subtraction flowgram" (section 8.3.7) for the pair of image pixels selected by the pin, that is, for each flow of the sequencing Run, the difference between the signals at the two ends of the pin. The accelerator key is *Ctrl-S*.

VI. **Add subtraction pin**: this action deposits a subtraction pin on the image (Figure 8–10). A subtraction pin is a visual indicator showing two image locations that can used for calculating a subtraction flowgram. A small circle ciscum-scribes the area that will be used for the calculation; a pentagon and a square around each of the location serve as "handles" that can be dragged to position the circles at the locations of interest; a bar between the two squares shows which two locations are associated with other. The direction of the subtraction is: "circle in the pentagon" (pin head) minus "circle in the square" (pin tail). You can add multiple subtraction pins on an image; the "active" one, *i.e.* on which an eventual "Subtraction flowgram" command would apply, is shown in blue; all the others are yellow. The accelerator key is *Ctrl-P*.

VII. **Clear pins**: removes all the subtraction pins currently in view. The accelerator key is *Ctrl-C*.

VIII. **Show image**, **Show regions**, and **Show wells** check boxes: control whether the corresponding layers of the image area are displayed (checked) or not (unchecked).

▶▶▶

3

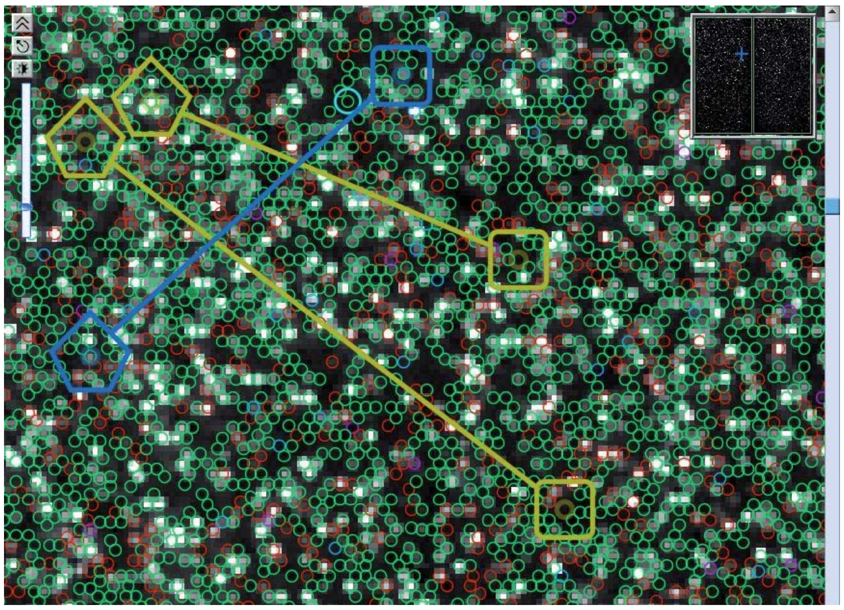


Figure 8–10: The image area of the Wells tab, with three subtraction pins. The one on which an eventual "Subtraction flowgram" command would apply is in blue. The direction of the subtraction is: "circle in the pentagon" (pin head) minus "circle in the square" (pin tail).

g. The image area of the Wells tab has the following special navigation controls:

I. **Collapse Icons** – Collapse the icons for the other navigation controls of the image area, on the Wells tab.

II. **Expand Icons** – Expand the icons for the other navigation controls of the image area, on the Wells tab.

III. **Reset Image** – Reset the view of the image to the default top-left location and default magnification.

IV. **Adjust White Threshold** – Adjust the upper threshold for white values in the image. Clicking this button elicits a slider that can be used to set the white threshold (Figure 8–11). The range is from 0 to 2500 and the default setting is 1000. Clicking the blue button to the right of the slider resets the threshold to its default value. The white threshold value selected applies to all images.

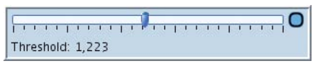


Figure 8–11: The Adjust White Threshold slider

V. **Magnification Slider** – Set the magnification of the image. The higher you go on the slider, the higher the magnification (zoom in).

VI. **Overview image**: shows a small representation of the entire PicoTiterPlate device, with the image, if available, but without the wells, as an inset near the upper-right corner of the image area. A blue "+" shows the current location of the mouse pointer, in the main image. Clicking (left or right) on the overview image scrolls the main image directly to the pixel on which you clicked.

▶▶▶

4 **Average well density summary**:

a. This area provides the statistical averages for raw well density and keypass well density, calculated for each region and for the entire PicoTiterPlate device.

b. For data sets processed with the GS Run Processor version 2.0.00 or later, these values are taken from the *region*.cwf files, if available. For older processed data sets, they are calculated by the GS Run Browser as the wells are loaded. For each well, the application counts the number of other wells in a standard size area around the well's center; the raw well density calculation includes all wells, whereas the key pass density calculation omits wells that did not key pass (had a **No Key** status value). Note that the GS Run Processor and the GS Run Browser use different algorithms to perform these calculations, and will not yield identical well density values, but regions of fairly uniform density will have close results.

c. An invisible divider is present between the image area and the well density summary table, that can be dragged to adjust the part of the window allotted to these two areas.

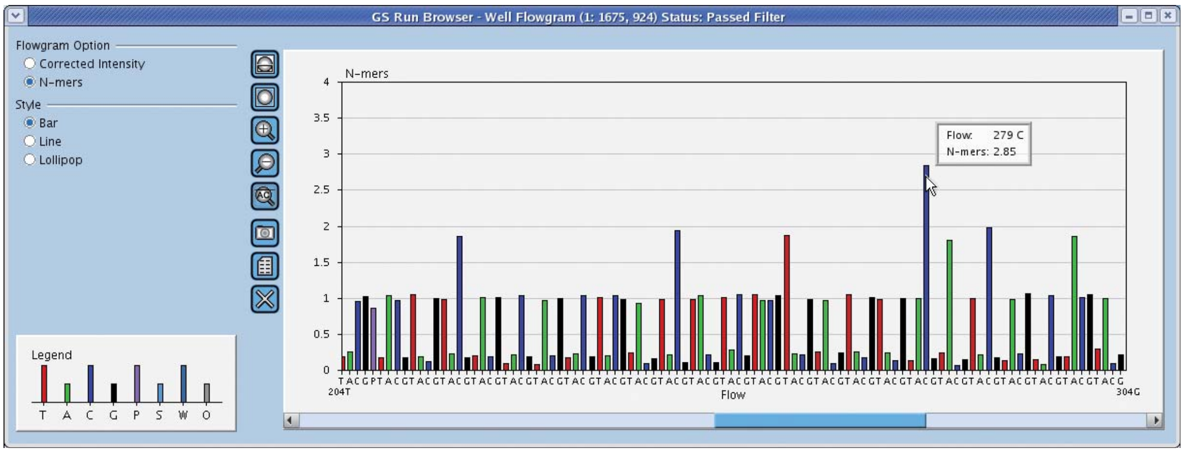
5 **Standard navigation buttons**:

a. The three standard navigation buttons on the Wells tab have common functions: allowing you to save a snapshot image of the main image area or of the average well density summary table; or a tab-delimited file of the summary table. See section 8.1.2.3 for a description of the button functions.

8.3.4 Well Flowgrams (for Wells Generating Library Reads)

8.3.4.1 General Description

Double-clicking the mouse in the main image area brings up the closest well's processed flowgram, which was generated by the GS Run Processor application and used to determine the basecalled sequence for that well. (In this flowgram view, however, all flows are shown rather than only the flows generating the trimmed sequence). For wells producing library reads (start with the "TCAG" key), a well flowgram viewer opens, showing the flowgram values (Figure 8–12). The flowgram viewer for Control DNA wells is described in section 8.3.5.


GS Run Browser - Well Flowgram (1: 1675, 924) Status: Passed Filter

Flowgram Option: Corrected Intensity [✓ N-mers]

Style: [✓ Bar] Line Lollipop

Flow: 279 C
N-mers: 2.85

N-mers
4
3.5
3
2.5
2
1.5
1
0.5
0
T A C G P T A C G
204T
Flow
304G
Legend
T A C G P S W O

Figure 8–12: Flowgram of a library DNA well at (region 1, position 1675, 924), a well that passed all filters, uses the bar style of display, and default y-axis settings for number of bases extended (N-mer). The mouse pointer is over the data point of flow 279, a "C" nucleotide flow with a 2.85 N-mer signal (as shown in the mouse tracker area).

Only one well flowgram window (or Control DNA well tri-flowgram window; see section 8.3.5) can be open at a time; selecting the well flowgram action for another well on the main image area will replace any existing data in the well flowgram window. However, you can have both a well flowgram window and a location or a subtraction flowgram window (see sections 8.3.6 and 8.3.7) open together.

8.3.4.2 Features and Functionalities

The well flowgram window has the following areas, with the functionalities described (see Figure 8–12):

1	The **title bar** of the window identifies the type of flowgram window (Well Flowgram), the PicoTiterPlate device region and the x,y location of the well whose flowgram is being displayed, and the well's attribute value for the currently selected category.
2	The **Options** area: a. **Flowgram option**: Choice of the type of flowgram signal to display: I. **Corrected Intensity** – Display the processed ("corrected") signal intensity as computed by the GS Run Processor application. II. **N-mers** – Display each signal as an estimate of the number of bases extended at each flow (*i.e.*, the homopolymer length). b. Choice of 3 plot **Styles**: Bars, Lines, or Lollipop. In all cases, the reagent flowed in each step is color-coded, per the legend. The lines and lollipop styles are narrower than the bar style, and will allow you to view more of your Run without scrolling.
3	**Navigation buttons**: a. All the buttons in this window have common functions: setting and adjusting the zoom level of the plot display or allowing you to save the data as a text file or snapshot image. See section 8.1.2.3 for a description of the button functions.
4	**Plot display**: a. The plot has the common functions, found in any plot across the GS Run Browser tabs, for scrolling and zooming the plot. See section 8.1.2.3 for a description of the plot functions. b. The default y-axis scale is set on the fly to values appropriate for visualizing the range of signals generated for the well flowgram displayed. The buttons can then be used to adjust this zoom level. c. When the mouse pointer is over the plot, the mouse tracker shows the flow number, the reagent flowed, and the count/intensity for the flow under the pointer.
5	**Signal legend**: a. This area shows the colors used to display the flowgram signals on the plot; each reagent is shown in a different color.

■

For data sets processed with the software version 1.1.03 or earlier, the nucleotide labels of the x axis of the well flowgrams come from the imageLog.parse file. If this file is absent (*e.g.* if the D_ directory has been separated from its parent R_ directory), the software will attempt to infer what the flows were. If incorrect, this can be overridden by specifying the proper data set type using the drop down menu on the Overview tab (section 8.2.3).

8.3.5 Well Tri-Flowgrams (for Wells Generating Control DNA Reads)

8.3.5.1 General Description

When the selected well produces a Control DNA read (starts with the ATGC key for the GS FLX standard chemistry, or the CATG key for the GS FLX Titanium chemistry), a "tri-flowgram" viewer opens (Figure 8–13). Tri-flowgrams allow the well's flowgram to be compared to an idealized flowgram of the Control DNA's reference sequence, or to be compared to the consensus flowgram generated from all the wells in the Run that contain this Control DNA sequence.

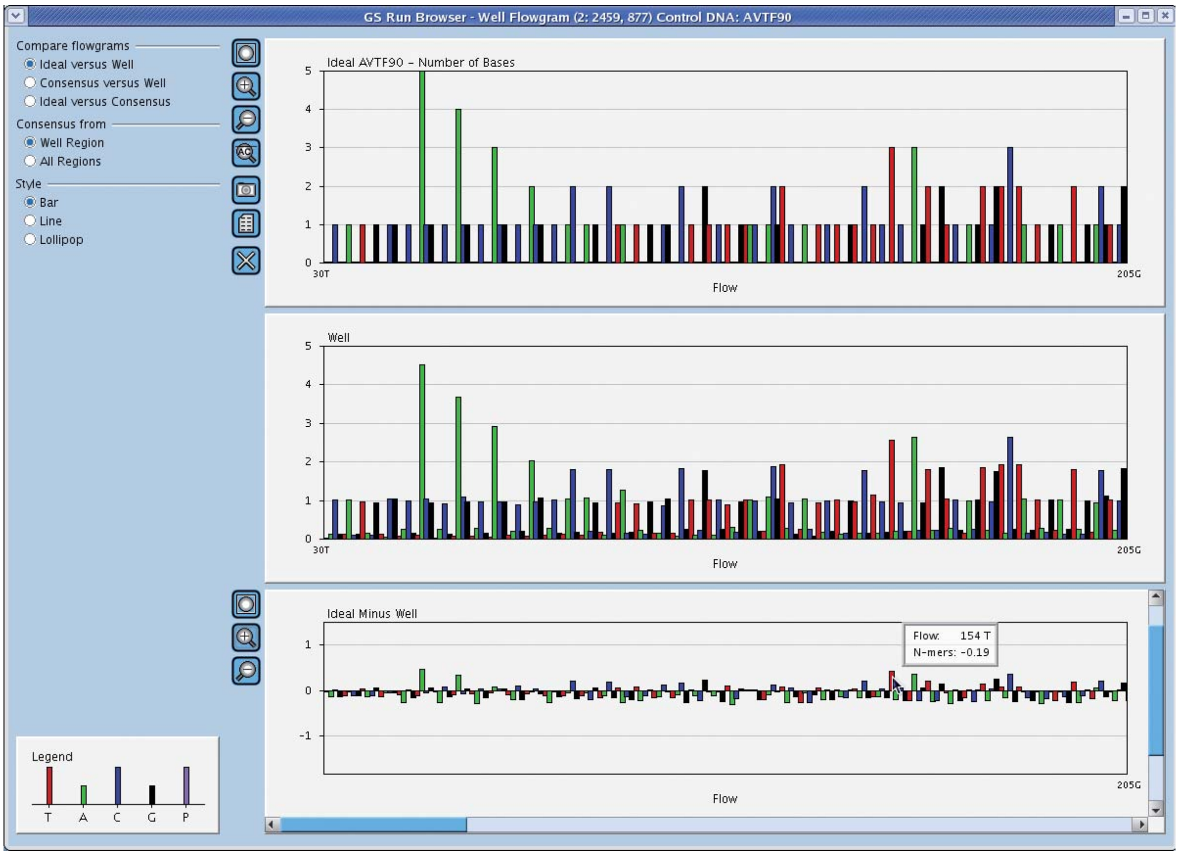


Figure 8–13: The tri-flowgram view of a Control DNA well (sequence AVTF90), at (region 2, position 2459, 877)

8.3.5.2 Features and Functionalities

The tri-flowgram viewer contains two flowgram plots, plus a "difference" flowgram showing the flow-by-flow difference between the top two flowgrams. This flowgram viewer has functionalities similar to the standard flowgram viewer, but with the following differences:

1 Three flowgram plots are calculated for this window:

a. An "ideal" flowgram constructed from theoretical values for the (known) sequence of the Control DNA fragment identified in the selected well. This is generated by taking the known nucleotide reference sequence and converting each homopolymer stretch into a corresponding signal (so, for example, "AAA" becomes a signal of 3.0 in the next A flow).

b. A "consensus" flowgram calculated as a flow-by-flow average of all the wells that contained the same Control DNA sequence as the selected well.

c. The flowgram of the selected well.

2 The **Options** area offers the following choices (specific to the tri-flowgram viewer):

a. **Compare flowgrams**:

I. **Ideal versus Well** – Display the idealized flowgram in the top plot and the selected well's flowgram in the middle; the bottom flowgram shows the difference, flow-by-flow.

II. **Consensus versus Well** – Display the consensus flowgram for the selected well's Control DNA sequence in the top plot and the selected well's flowgram in the middle; the bottom flowgram shows the difference, flow-by-flow.

III. **Ideal versus Consensus** – Display the ideal flowgram in the top plot and the consensus flowgram in the middle; the bottom flowgram shows the difference, flow-by-flow.

b. **Consensus from**:

I. **Well region** – Average the signals of the reads (for this Control DNA sequence) only from the same region as the selected well.

II. **All regions** – Average the signals of all the reads (for this Control DNA sequence) across the whole PicoTiterPlate device.

c. Only the **N-mers** flowgram signals are shown, because the **Corrected Intensity** signals are not relevant in this type of comparison.

3 The **Navigation buttons** and **Plot display** have the following special functionalities (compared to the standard well flowgram view):

a. The scrolling and zooming of the three plots are tied together. All three plots scroll and zoom together along the x-axis, and the top two plots scroll and zoom together along the y-axis (the bottom "difference" plot scrolls and zooms separately along the y-axis).

b. Since it zooms separately, the bottom plot has its own zoom buttons.

c. The horizontal bars between the plots allow for resizing the heights of the three plots.

d. The text file and snapshot image buttons will save a file containing the data or view for all three plots.

■

8.3.6 Location Flowgrams

8.3.6.1 General Description

Selecting the Location flowgram action from the right-click menu from anywhere in the main image area brings up a “raw” flowgram window for that location (Figure 8–14). The flowgram is constructed by computing, for each image in the sequencing Run the average raw (non-corrected) signal intensity for the 9 pixels surrounding the selected location, and plotting these averages against the succession of reagent flows. (Note that this calculation does not give any consideration to the notion of “wells”.)

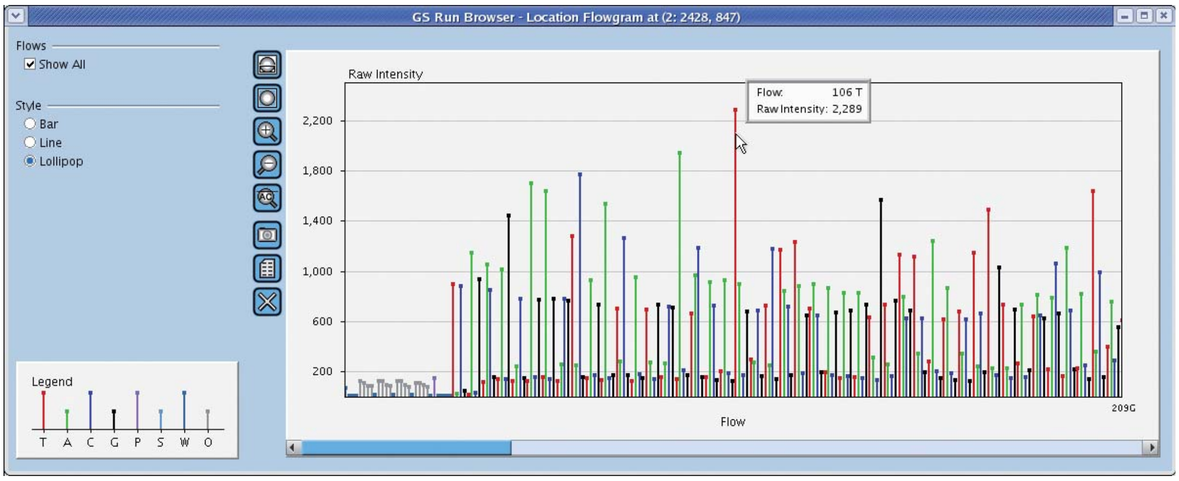


Figure 8–14: The Location flowgram of image pixel (region 2, position 2428, 847). It is shown with the default y-axis settings, the lollipop style is chosen, and all flows are displayed. The mouse pointer is over the data point of flow 106, a “T” nucleotide flow with a fairly high raw signal, as shown in the mouse tracker.

Only one Location flowgram window (or subtraction flowgram window; see section 8.3.7) can be open at a time. Selecting the Location flowgram or Subtraction flowgram action for another location on the main image display will replace any existing data in a Location flowgram window. However, you can have both a Location/Subtraction flowgram window and a Well/Tri-flowgram flowgram window (see sections 8.3.4 and 8.3.5) open together.

8.3.6.2 Features and Functionalities

The Location flowgram window is very similar to the Well flowgram window (section 8.3.4). The only four differences are:

- The window title bar identifies it as containing a Location flowgram.
- The flowgram plot displays raw signal intensities instead of corrected intensities.
- There isn’t an option to display N-mers since these cannot be computed from non-normalized (corrected) data.
- A **Show all** check box allows you to display (checked) or hide (unchecked) the flows that are neither nucleotides nor PPi/ATP.

8.3.7 Subtraction Raw Flowgrams

8.3.7.1 General Description

In addition to single location raw flowgrams, the GS Run Browser can generate a Subtraction (raw) flowgram from any two locations on the main image (Figure 8–15). Subtraction flowgrams are produced by right-clicking on a subtraction pin and selecting the Subtraction flowgram action, as described in section 8.3.3, step 3.f. This is an advanced diagnostic function that can be used either:

- to generate a raw flowgram that more closely matches a presumptive well's processed flowgram by subtracting the local background: if the signal intensity at the pin tail (square) location is lower than a set threshold (450 counts for "A" flows and 150 counts for any other flow except PPi), the software will compute this simple local background subtraction; or
- to compare the flowgrams from two well locations (or any two locations): if the signal intensity at the pin tail is above the threshold, the software will assume that you are comparing two wells. In this case, since different wells have different light-generating efficiencies, the calculation first normalizes the raw flowgrams at both locations by the average pixel intensity of their respective "first PPI" signals, before doing the subtraction. This type of calculation is also always applied if the Subtraction flowgram is requested from a PPi image, irrespective of the pin tail signal.

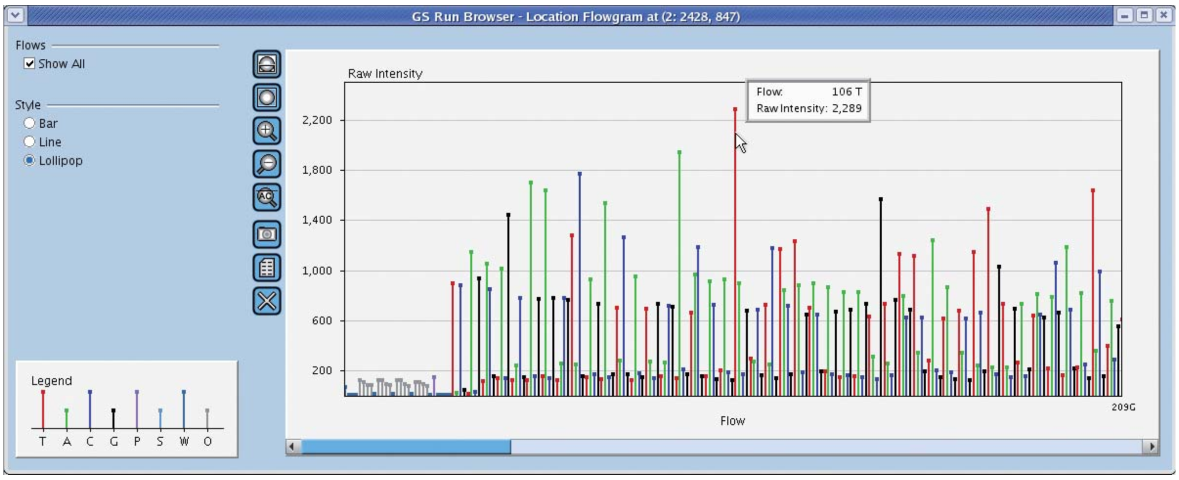


Figure 8–15: The Subtraction flowgram of image pixels (region 2, position 2420, 838) minus (region 2, position 2452, 841). It is shown with a scale-to-fit y-axis, the lines style is chosen, and the wash steps are not displayed. The mouse pointer is over the data point of flow 154, a "T" nucleotide flow that had substantially more signal in the first pixel than in the second, as shown in the mouse tracker.

8.3.7.2 Features and Functionalities

A Subtraction flowgram window is identical to a Location flowgram window (section 8.3.6), except that:

- The window title bar identifies it as containing a Subtraction flowgram, and references two pixel locations rather than one.
- the y-axis may contain negative values, since the subtraction may yield "signals" below zero; this is akin to the "difference plot" of a tri-flowgram window (section 8.3.5).

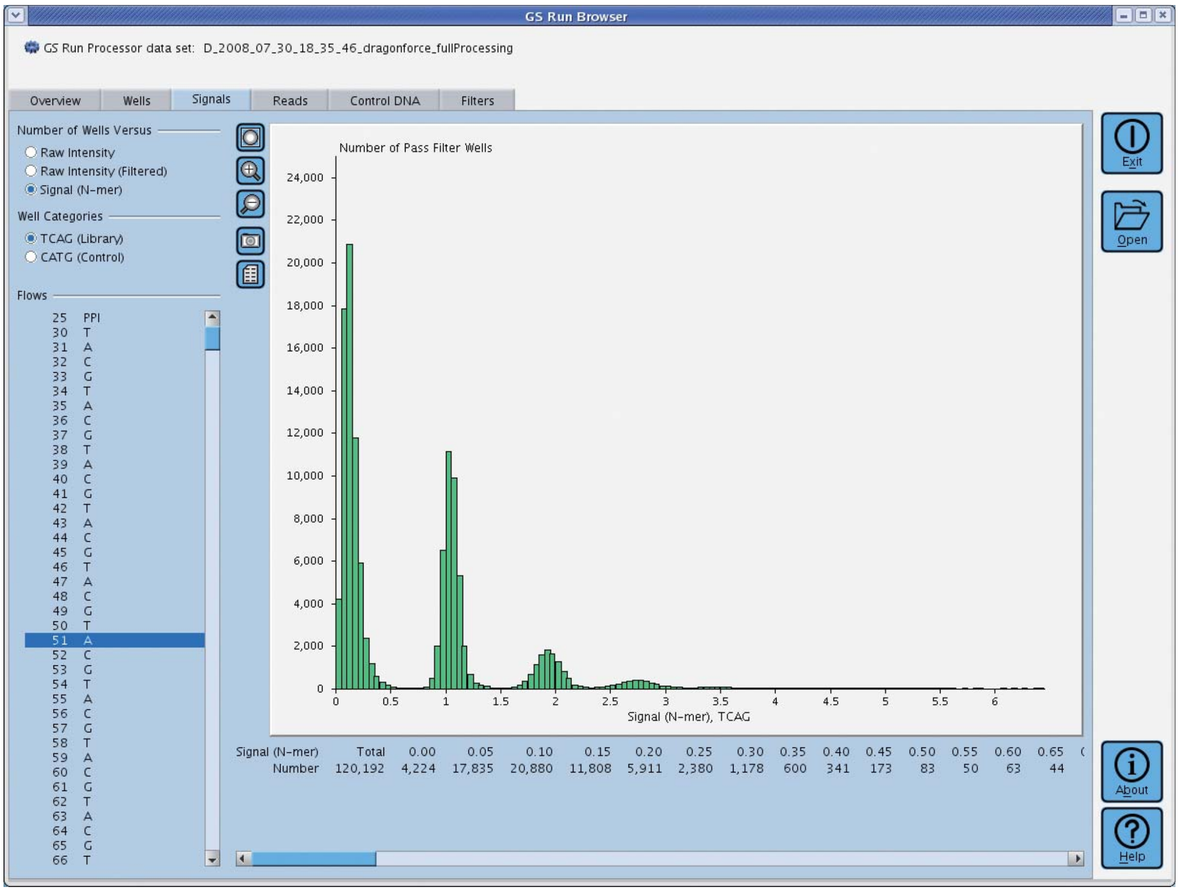
8.4 The Signals Tab

8.4.1 General Description

The Signals tab (Figure 8–16) provides statistics on the distribution of the signals recorded during a Run, and how they were interpreted by the data processing software as number of bases. The tab displays either library well signals (sequencing key TCAG) or Control DNA well signals (sequencing key ATGC or CATG, for GS FLX standard and GS FLX Titanium chemistries, respectively), in one of the following three modes:

- **Raw Intensity**: the signal intensities as found in the *region*.cwf (or *region*.wells) files, using all the key passed wells for the library or Control DNA wells.
- **Raw Intensity (Filtered)**: the same signal intensity values, but using only the wells that both key passed and passed all quality filters.
- **Signals (N-mer)**: the basecalling estimates for the signal intensities, where the Raw Intensity (Filtered) signal values are converted into the estimated numbers of nucleotides that were incorporated in each well during the flow selected (for filter pass wells only).

The information displayed on this tab is not normally used for the evaluation of a sequencing Run, but it can prove useful as an advanced diagnostic tool to troubleshoot problems.



GS Run Processor data set: D_2008_07_30_18_35_46_dragonforce_fullProcessing

Number of Wells Versus
- ( ) Raw Intensity
- ( ) Raw Intensity (Filtered)
- (●) Signal (N-mer)

Well Categories
- (●) TCAG (Library)
- ( ) CATG (Control)

Flows

25	PPI
30	T
31	A
32	C
33	G
34	T
35	A
36	C
37	G
38	T
39	A
40	C
41	G
42	T
43	A
44	C
45	G
46	T
47	A
48	C
49	G
50	T
51	A
52	C
53	G
54	T
55	A
56	C
57	G
58	T
59	A
60	C
61	G
62	T
63	A
64	C
65	G
66	T

Signal (N-mer)	Total	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65
Number	120,192	4,224	17,835	20,880	11,808	5,911	2,380	1,178	600	341	173	83	50	63	44

Figure 8–16: The GS Run Browser application's Signals tab. The plot shows the number of wells that produce final (passed filter) library reads and those reads interpreted by the data processing software as the extension of a given number of nucleotides; this data was collected during flow 51 of the sequencing Run, an "A" nucleotide flow. Details on the various areas of the tab shown on this figure are provided below.

8.4.2 Source of the Data

For the Signals tab data to be displayed, the *region*.cwf (or *region*.wells) file(s) located in the "regions" sub-folder of the "D_" data analysis folder of the sequencing Run must exist (see section 1.3). If no *region*.cwf (or *region*.wells) files exist for the Run, the Signals tab will be unavailable.

8.4.3 Features and Functionalities

The Signals tab contains the following areas, with the functionalities described (see Figure 8–16).

1. The **name** (and path, via the tooltip) **of the Run or Data Processing directories** and the **tabs** are common to the GS Run Browser application (see section 8.1.2).

2. The **Options** area (on the top left area of the tab) gives a choice of 6 plot types (each data set is available *for each flow*), broken down as follows:
   a. **Number of Wells Versus**: For the well type selected (see item 2.b, below), display the number of wells (y-axis of the plots) as a function of (x-axis of the plots):
      I. **Raw Intensity** for all key pass wells of the type chosen (Library or Control DNA), at the flow picked in the Flow selector.
      II. **Raw Intensity (Filtered)** for only filter pass (always also key pass) wells of the type chosen (Library or Control DNA), at the flow picked in the Flow selector.
      III. **Signal (N-mer)** equivalent of the intensity values from the **Raw Intensity (Filtered)** signals (*i.e.* "homopolymer length", as estimated by the basecaller), for filter pass wells of the type chosen (Library or Control DNA), at the flow picked in the Flow selector. This is the default view (for library wells); unless you find a problem with the Signal (N-mer) plot, it is unlikely that the Raw Intensity plots will be needed.
   b. **Well Categories**: For the type of output selected (see item 2.a, above), display the data for the wells with the following key:
      I. **TCAG (library)** – Use the key passed wells identified as library wells.
      II. **CATG (control)**, or **ATGC (control)** – Use the key passed wells identified as Control DNA wells. CATG is the Control DNA key used with the GS FLX Titanium chemistry, and ATGC is the Control DNA key used with the GS FLX standard chemistry.
   c. **Flows:** Located below the other options' radio buttons, the Flows selector allows the user to choose the flow from the sequencing Run whose signal statistics you want to display.
      I. It parallels the Flows selector of the Wells tab (section 8.3.3, step 2.c). The main difference is that only the nucleotide and PPi flows are listed: signal statistics for wash flows would be of no interest, so there is no "Show all" check box.
      II. As mentioned above, after clicking in the Flows selector, you can use the keyboard arrow keys to navigate quickly up and down this list. This feature is especially useful on this tab.

3. **Navigation buttons**:
   a. All the buttons on this tab have common functions: setting and adjusting the zoom level of the main image display or allowing you to save the data as a text file or snapshot image. See section 8.1.2.3 for a description of the button functions.

▶▶▶

4 **Plot display**:

a. The plot has the common functions, found in any plot across the GS Run Browser tabs, for scrolling and zooming the plot. See section 8.1.2.3 for a description of the plot functions.
b. The default scales for the x and y axes have been set to fixed values that typically provide a useful view of the data. After first clicking the Flow selector, you can use the keyboard arrow keys to quickly scan from flow to flow; this constitutes a very useful diagnostic tool for evaluating the signal distributions.
c. The bar widths (*i.e.* histogram "bin sizes") are 100.0 for Raw Intensity plots, and 0.05 for Signal (N-mer) plots. These bar widths cannot be adjusted.
d. When the mouse pointer is over the plot, the usual mouse tracker shows the x-axis signal value and the number of wells having that signal value, in the selected flow.

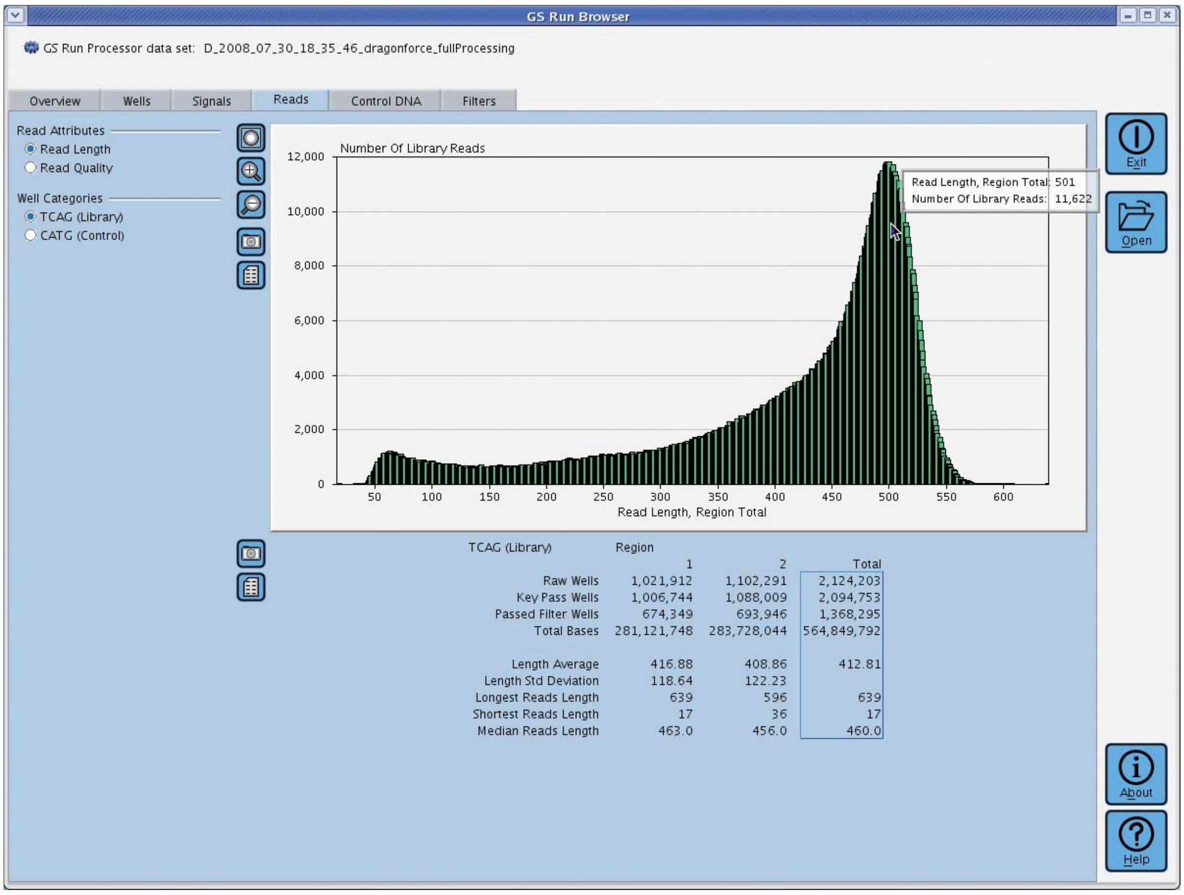
5 **Summary**:

a. This displays a text version of the data shown in the plot, listing the height of each bar in the plot, with zero height bars omitted.
b. An invisible divider is present between the image area and the well density summary table, that can be dragged to adjust the part of the window allotted to these two areas.

8.5 The Reads Tab

8.5.1 General Description

The Reads tab (Figure 8–17) provides statistics on the read length and read quality observed during the sequencing Run; it can show these values for either the sample library reads (sequencing key TCAG) or the Control DNA reads (sequencing key ATGC or CATG, for GS FLX standard and GS FLX Titanium chemistries, respectively). It displays a plot of the read length or quality data either for a selected PicoTiterPlate device region or across the whole PTP device, and a table of region-by-region and whole PTP device statistics of related data.



TCAG (Library)	Region 1	2	Total
Raw Wells	1,021,912	1,102,291	2,124,203
Key Pass Wells	1,006,744	1,088,009	2,094,753
Passed Filter Wells	674,349	693,946	1,368,295
Total Bases	281,121,748	283,728,044	564,849,792
Length Average	416.88	408.86	412.81
Length Std Deviation	118.64	122.23	
Longest Reads Length	639	596	639
Shortest Reads Length	17	36	17
Median Reads Length	463.0	456.0	460.0

Figure 8–17: The GS Run Browser application's Reads tab, showing the number of wells producing final library reads over the entire PTP device, and whose reads were determined by the data processing software as being of any given read length.

8.5.2 Source of the Data

For the Reads tab data to be displayed, the *region*.cwf files (or, for the GS FLX standard chemistry, files 454QualityFilterMetrics.txt and 454BaseCallerMetrics.txt) must be present in the data processing directory of the selected Run (see section 1.3).

8.5.3 Features and Functionalities

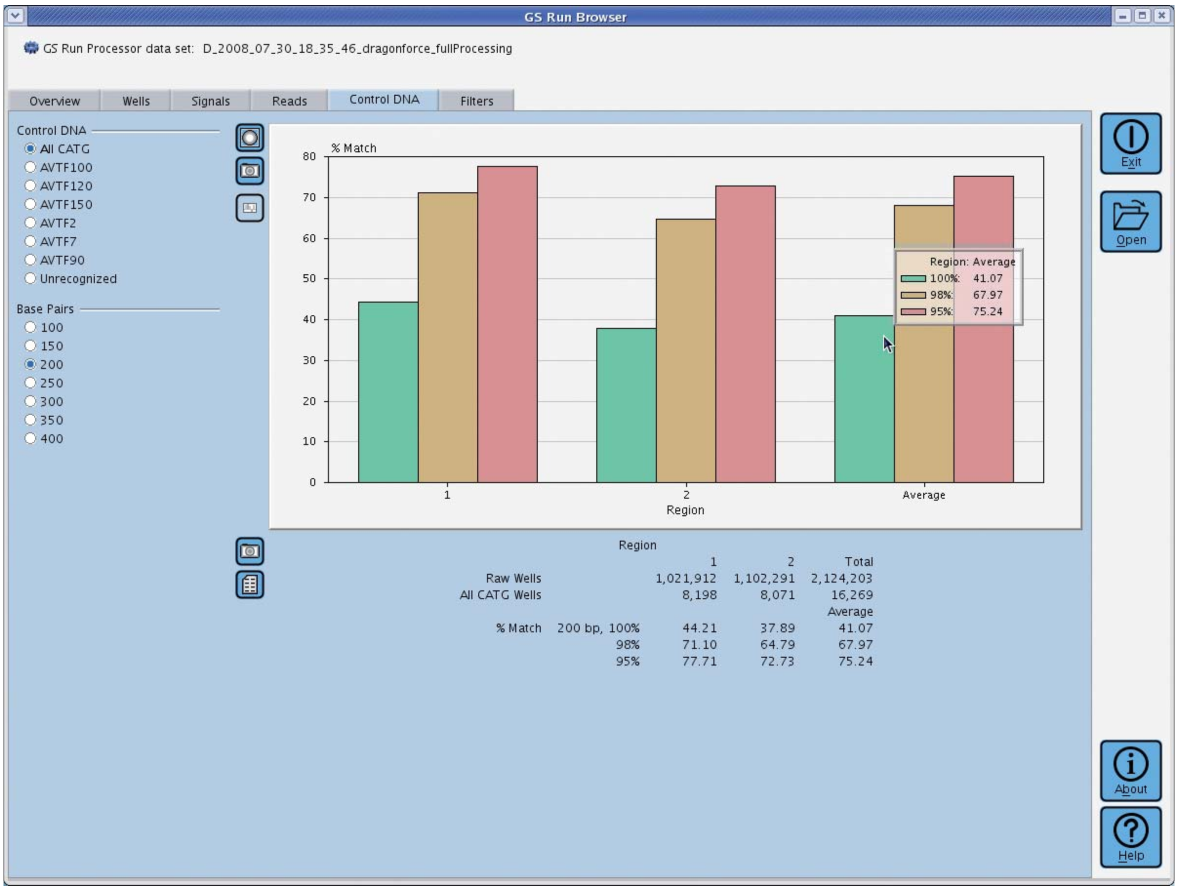
The Reads tab contains the following areas, with the functionalities described (see Figure 8–17):

1. The **name** (and path, via the tooltip) **of the Run or Data Analysis directory** and the **tabs** are common to the GS Run Browser application (see section 8.1.2).
2. The **Options** area (on the top left area of the tab):
   a. **Read Attributes**: choice of data type to display:
      I. **Read length**, or
      II. **Quality**
   b. **Well Categories**: choice of reads to display (only pass filter reads):
      I. **TCAG (library)**, or
      II. **CATG (control),** or **ATGC (control).** CATG is the Control DNA key used with the GS FLX Titanium chemistry, and ATGC is the Control DNA key used with the GS FLX standard chemistry.
3. **Navigation buttons**:
   a. All the buttons on this tab have common functions, setting and adjusting the zoom level of the plot display or allowing you to save the data as a text file or snapshot image. See section 8.1.2.3 for a description of the button functions.
4. **Plot display**:
   a. Graphic display of the data, as selected in the Options area (item 2, above) for a single region or for the whole PTP device, as selected on the Summary (item 4.e, below).
   b. Each selection on the options area refreshes the plot (and Summary) with a different set of data, so there are four possible kinds of plot displays to choose from.
   c. For read length, the "count" (y-axis) means the number of reads that had this length. For read quality, the "count" (y-axis) means the number of bases that had this quality score (Phred equivalent; see section 3.2.2.2).
   d. The plot has the common functions, found in any plot across the GS Run Browser tabs, for scrolling and zooming the plot. See section 8.1.2.3 for a description of the plot functions.
   e. When the mouse pointer is over the plot, the usual mouse tracker shows the x-axis signal value and the number of reads having that attribute value, in the selected region or entire PTP device.
5. **Summary**:
   a. This shows the same data as in the plot display (choice of 4 kinds of sets of data), for each of the regions and for the entire PTP device, plus additional intermediate data.
   b. A blue rectangle identifies the set of data (single region or "Total" column) currently displayed on the plot. When you move the mouse over the data in another column, the highlight moves to this column; left-clicking the mouse selects the highlighted column (blue rectangle) for display in the plot.

8.6 The Control DNA Tab

8.6.1 General Description

The Control DNA tab (Figure 8–18) displays accuracy results for the Control DNA Beads that were spiked in the sequencing reaction, if available [see the *GS FLX (Titanium) Sequencing Method Manual* for details about this procedure]. Specifically, the Control DNA tab reports the percentage of Control DNA reads that match their reference sequence at 95%, 98%, and 100% accuracy, in each region of the PicoTiterPlate device. It also allows you to view the “consensus flowgram” for the reads from each Control DNA sequence. (A consensus flowgram is the flowgram constructed by averaging, for each nucleotide flow, the signals of the reads that matched the reference sequence, for the selected region or the entire PicoTiterPlate device.)



GS Run Processor data set: D_2008_07_30_18_35_46_dragonforce_fullProcessing

Control DNA
- [✓ All CATG]
- AVTF100
- AVTF120
- AVTF150
- AVTF2
- AVTF7
- AVTF90
- Unrecognized

Base Pairs
- 100
- 150
- [✓ 200]
- 250
- 300
- 350
- 400

```
                                Region
                                     1          2       Total
      Raw Wells              1,021,912  1,102,291   2,124,203
  All CATG Wells                 8,198      8,071      16,269
                                                      Average
        % Match  200 bp, 100%    44.21      37.89       41.07
                          98%    71.10      64.79       67.97
                          95%    77.71      72.73       75.24
```

Figure 8–18: The GS Run Browser application’s Control DNA tab, showing the % match of the first 200 bases of all Control DNA reads to their respective reference sequences, for each region of the PicoTiterPlate device and as an aggregate average. This is presented both graphically and numerically.

8.6.2 Source of the Data

For the Control DNA tab data to be displayed, the *region*.cwf files (or, for the GS FLX standard chemistry, files 454RunTimeMetricsAll.txt and 454QualityFilterMetrics.txt) must be present in the Analysis sub-directory of the sequencing Run (see section 1.3). If the *region*.wells files also exist (GS FLX standard chemistry only), the consensus flowgrams can be displayed.

8.6.3 Features and Functionalities

The Control DNA tab contains the following areas, with the functionalities described (see Figure 8–18):

1 The **name** (and path, via the tooltip) **of the Run or Data Analysis directory** and the **tabs** are common to the GS Run Browser application (see section 8.1.2).

2 The **Options** area (on the top left area of the tab):

a. **Control DNA**: choice of data to display:
   I. All Control DNA sequences
   II. Any of the 6 specific Control DNA sequences (different set for the GS FLX Titanium chemistry or the GS FLX standard chemistry)
   III. Unrecognized reads, *i.e.* reads which begin with the Control DNA sequencing key (CATG for the GS FLX Titanium chemistry, and ATGC for the GS FLX standard chemistry) but do not match any of the corresponding Control DNA reference sequences

b. **Base pairs**: choice of length over which to calculate match, *i.e.* show % match from the first base after the key up to this base in the reads. Only the options relevant to the read length of the data set (number of nucleotide cycles included in the sequencing Run script) are displayed.

c. A button to display the consensus flowgrams for each of the specific Control DNA sequences (as selected) in a **tri-flowgram viewer** (Figure 8–19). See section 8.6.4 for a complete description of the Control DNA consensus flowgram view. Note that this button is available only if the current selection is one of the Control DNA sequences; and grayed out if "all" or "unrecognized" are selected.

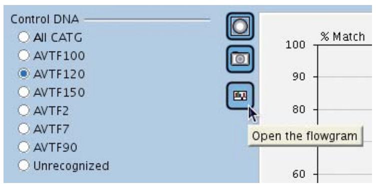


Figure 8–19: The "Display Consensus Flowgram" button of the Control DNA tab, available when one of the Control DNA sequences is selected

3 **Navigation buttons**:

a. Except for the "Display Consensus Flowgram" button described above, all the buttons on this tab have common functions, which allows you to save the data as a text file or snapshot image. See section 8.1.2.3 for a description of the button functions.

4 **Plot display**:

a. Graphic display of the % match data for all key passed Control DNA reads per the options selected (item 2, above), for the individual regions of the PicoTiterPlate device and as an aggregate average.

b. This data is presented at 3 levels of accuracy:
   I. **100%**: The percent of key passed Control DNA reads that *exactly* matched the first number of nucleotides (per the "base pairs" selection) of the corresponding known reference sequence. Zero base-calling differences in these first bases of the sequence.
   II. **98%**: Percent Control DNA reads that had no more than 2% base calling differences (insertion or deletion) compared to their known reference sequence, in their first number of nucleotides (per the "base pairs" selection).
   III. **95%**: Percent Control DNA reads that had no more than 5% base calling differences (insertion or deletion) compared to their known reference sequence, in their first number of nucleotides (per the "base pairs" selection).

▸▸▸

4

c. The plot has the common functions, found in any plot across the GS Run Browser tabs, for scrolling and zooming the plot. See section 8.1.2.3 for a description of the plot functions.

d. When the mouse pointer is over the plot, the usual mouse tracker shows the x-axis value, and the % of reads among the Control DNA sequences selected that matched their respective reference sequences at 100, 98, and 95% accuracy over the selected length, in the region under the pointer or averaged over the entire PTP device.

5

Summary:

a. This area displays the number of Raw Wells and Control DNA reads for the Control DNA species specified; it also displays the percentage matches between these reads and their reference sequence at the read length specified, for each region of the PicoTiterPlate device and as an aggregate average.

8.6.4 Control DNA Consensus Flowgrams

8.6.4.1 General Description

Clicking the "Display Consensus Flowgram" button (in the Options area of the Control DNA tab), when one of the Control DNA sequences is selected, brings up the consensus flowgram for that sequence (Figure 8–20). A consensus flowgram is the flowgram constructed by averaging, for each nucleotide flow, the read flowgram signals of the reads identified as that reference sequence. This is presented as part of a tri-flowgram, along with the "ideal" flowgram for that Control DNA sequence and the "difference" flowgram, in a manner similar to the tri-flowgram of a Control DNA well, described before (section 8.3.5).

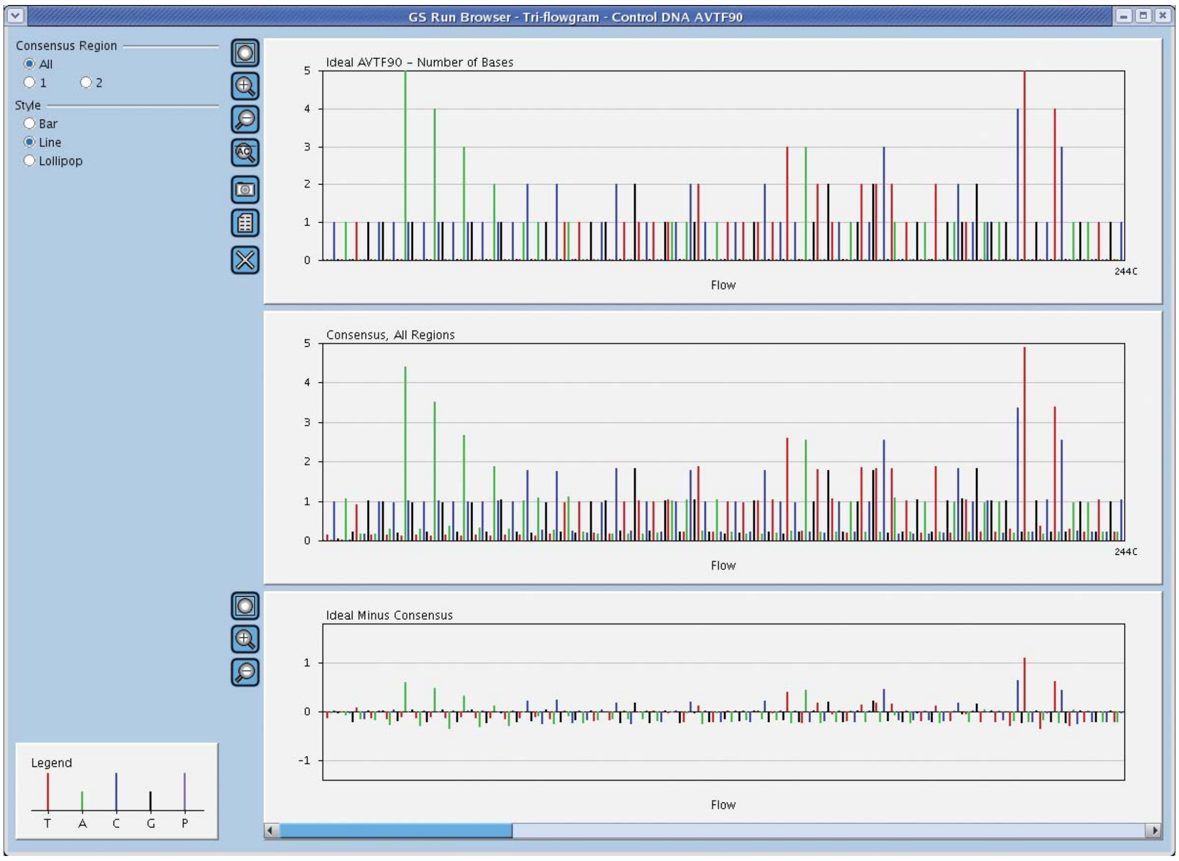


Figure 8–20: The tri-flowgram view of the AVTF90 Control DNA sequence, showing the consensus flowgram for both regions of the PicoTiterPlate device

8.6.4.2 Features and Functionalities

The tri-flowgram view of a Control DNA consensus sequence is similar to that of an individual Control DNA read (see section 8.3.5); it has the following areas, with the functionalities described:

1. The **title bar** of the window displays the name of the Control DNA sequence whose tri-flowgram is displayed in the window.

2. The **Options** area provides the following choices:
   a. **Consensus Region**: choice of what PicoTiterPlate device **region** data to display:
      I. **All** – Show the consensus flowgram formed by averaging all the read flowgrams across the entire PicoTiterPlate device, for the Control DNA sequence specified.
      II. individual region – Show the consensus flowgram of the reads from an individual region of the PicoTiterPlate device for the Control DNA sequence specified.

   b. **Style**: choice of Bars, Lines, or Lollipop plot styles. In all cases, the reagent flowed in each step is color-coded, per the legend. The lines and lollipop styles are narrower than the bar style, and will allow you to view more of your Run without scrolling.

3. **Navigation buttons**:
   a. All the buttons in this window have common functions: setting and adjusting the zoom level of the plot display or allowing you to save the data as a text file or snapshot image. See section 8.1.2.3 for a description of the button functions.

   b. The text file and snapshot image buttons will save a file containing the data or view for all three plots.

   c. The bottom plot contains separate zooming buttons because the scrolling and zooming of the plots are tied together except for the y-axis of the bottom plot (see below).

4. **Plot display**:
   a. The top plot displays the idealized flowgram for the Control DNA sequence selected, generated by taking the known nucleotide reference sequence and converting each homopolymer stretch into a corresponding signal (so, for example, “AAA” becomes a signal of 3.0 in the next A flow). The middle plot displays the consensus flowgram calculated as a flow-by-flow average of all the reads matching this Control DNA sequence in the region(s) selected, and the bottom plot displays the flow-by-flow differences between the top two plots.

   b. The plots have the common functions, found in any plot across the GS Run Browser tabs, for scrolling and zooming. See section 8.1.2.3 for a description of the plot functions.

   c. However, the scrolling and zooming of the three plots are tied together. All three plots scroll and zoom together along the x-axis, and the top two plots scroll and zoom together along the y-axis as well (the bottom “difference” plot scrolls and zooms separately along the y-axis).

   d. Since it zooms separately, the bottom plot has its own zoom buttons.

   e. The horizontal bars between the plots allow for adjusting the heights of the three plots.

   f. When the mouse pointer is over the plot, the mouse tracker shows the flow number, the reagent flowed and the N-mers count for the flow under the p.

5. **Signal legend**:
   a. This area shows the colors used to display the flowgram signals on the plot; each reagent is shown in a different color.

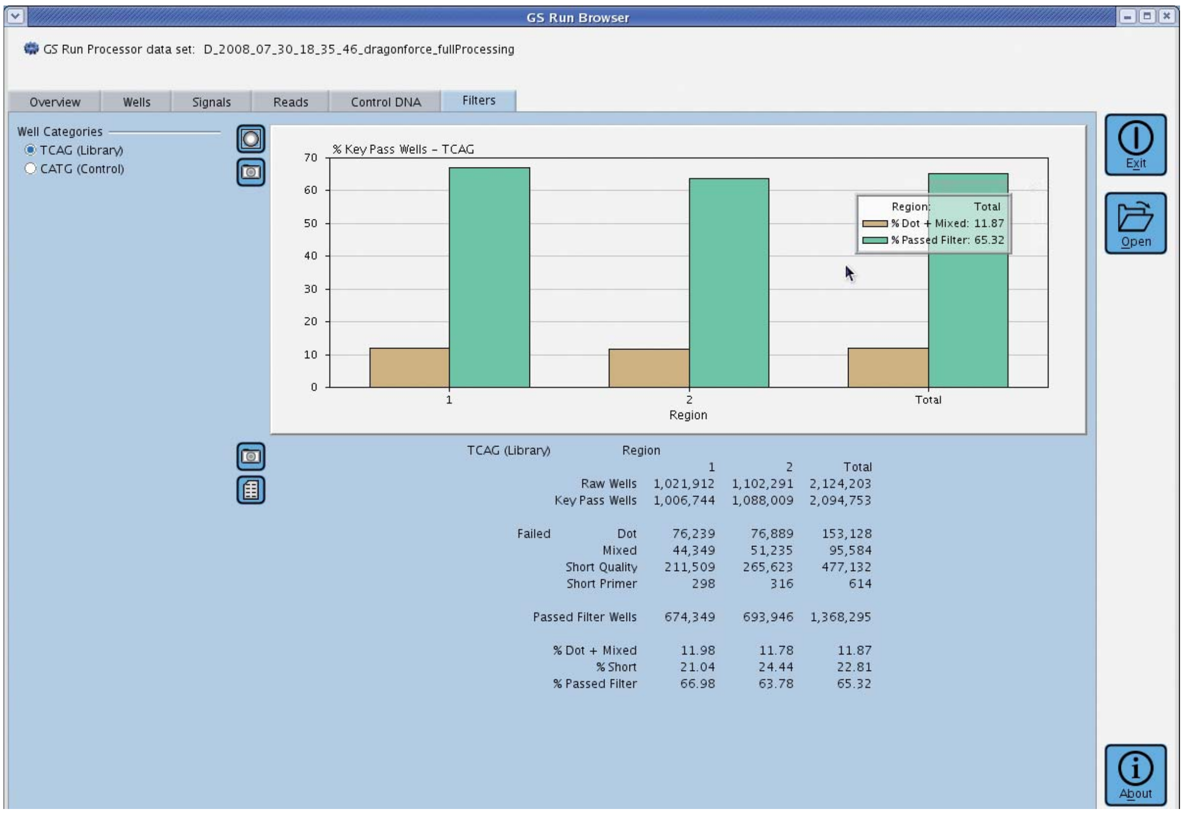
■

8.7 The Filters Tab

8.7.1 General Description

The Filters tab (Figure 8–21) provides statistics on the read quality filters of the GS Run Processor application used to analyze a sequencing Run. The statistics are displayed, per PicoTiterPlate region and in aggregate, for either the sample library reads (sequencing key TCAG) or the Control DNA reads (sequencing key ATGC or CATG, for GS FLX standard and GS FLX Titanium chemistries, respectively). The filtering statistics displayed are: **Key Pass, Dot, Mixed, Short Quality, and Short Primer**.

In addition, for sequencing Runs carried out using the GS 20 or GS FLX standard chemistries, the tab can compute an "Expected bead recovery" figure (also known as "Predicted emPCR bead yield") for each region of a multi-lane gasket ("small regions" format). This figure, assuming these data belong to a titration experiment, can help you determine the optimal amount of input DNA to use for future emPCR amplifications/ sequencing Runs with this library (for more details on titration experiments, see the *GS FLX Shotgun DNA Library Preparation Method Manual*). This feature DOES NOT APPLY to sequencing Runs carried out using the GS FLX Titanium chemistry.



GS Run Processor data set: D_2008_07_30_18_35_46_dragonforce_fullProcessing

TCAG (Library)

		Region 1	2	Total
	Raw Wells	1,021,912	1,102,291	2,124,203
	Key Pass Wells	1,006,744	1,088,009	2,094,753
Failed	Dot	76,239	76,889	153,128
	Mixed	44,349	51,235	95,584
	Short Quality	211,509	265,623	477,132
	Short Primer	298	316	614
	Passed Filter Wells	674,349	693,946	1,368,295
	% Dot + Mixed	11.98	11.78	11.87
	% Short	21.04	24.44	22.81
	% Passed Filter	66.98	63.78	65.32

Figure 8–21: The GS Run Browser application's Filters tab showing the results of the various quality filters for the sample library wells

8.7.2 Source of the Data

For the Filter tab data to be displayed, the *region*.cwf files (or, for the GS FLX standard chemistry, the 454QualityFilterMetrics.txt file) must be present in the data processing directory of the selected Run (see section 1.3).

8.7.3 Features and Functionalities

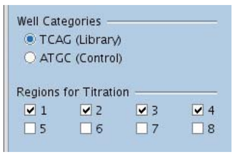
The Filter tab contains the following areas, with the functionalities described (see Figure 8–21):

1 The **name** (and path, via the tooltip) **of the Run or Data Analysis directory** and the **tabs** are common to the GS Run Browser application (see section 8.1.2).

2 The **Options** area (on the top left area of the tab):

a. **Well Categories**: choice of reads to display:
   I. **TCAG (library)**, or
   II. **CATG (control)**, or **ATGC (control)**. CATG is the Control DNA key used with the GS FLX Titanium chemistry, and ATGC is the Control DNA key used with the GS FLX standard chemistry.

b. **Regions for Titration**: Only for sequencing Runs carried out with the GS 20 or the GS FLX standard chemistry kits, choice of whether to evaluate the data as a titration and if so, choice of which regions to include in such evaluation (Figure 8–22; see also item 5.b, below). This is not applicable (grayed out) if the Control DNA reads are selected; and is not displayed at all for sequencing Runs carried out with a GS FLX Titanium series sequencing kit.


Well Categories
[✓ TCAG (Library)]
ATGC (Control)

Regions for Titration
[✓ 1] [✓ 2] [✓ 3] [✓ 4]
5 6 7 8

Figure 8–22: Regions for Titration selector, available only for sequencing Runs carried out with the GS 20 or the GS FLX standard chemistry kits, when data for the library wells category is selected.

3 **Navigation buttons**:

a. All the buttons on this window have common functions:
   setting and adjusting the zoom level of the plot display or allowing you to save the data as a text file or snapshot image. See section 8.1.2.3 for a description of the button functions.

4 **Plot display**:

a. The plot shows the number of reads that passed all filters and the number of reads that failed either the dot or the mixed filter (or both); these data are calculated as a percentage of key pass reads, for the sequencing key selected.

b. The plot has the common functions, found in any plot across the GS Run Browser tabs, for scrolling and zooming the plot. See section 8.1.2.3 for a description of the plot functions.

c. When the mouse pointer is over the plot, the usual mouse tracker shows the x-axis value, and the % of reads in the selected well category that passed all filters and that failed the dot and/or mixed filters, in the region under the pointer or averaged over the entire PTP device.

►►►

5 **Summary**:

a. The Summary area shows all filter data, per PicoTiterPlate region and in aggregate, including:

  I. Number of **Raw Wells**

  II. Then, for the key selected, the **number of reads** that:

    01. passed key

    02. failed each filter

    03. passed all filters

  III. … and, for the key selected, the **percentage of keypass reads** that:

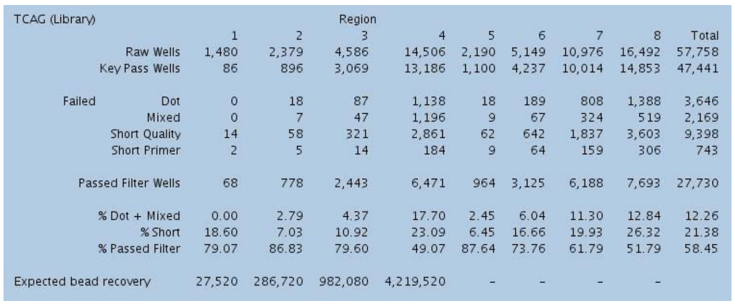
    01. failed either or both the "dot" or the "mixed" filters (combined)

    02. failed either or both of the "short" filters (combined)

    03. passed all filters

b. Only for sequencing Runs carried out with the GS 20 or the GS FLX standard chemistry kits, if any region(s) are selected to be Evaluated as a titration (see item 2.b, above), the **Expected bead recovery** line is added at the bottom of the Summary (Figure 8–23, below). This information makes sense only for the library reads and only if the experiment was indeed a titration, performed with the GS 20 or the GS FLX standard chemistry, as described in the *GS FLX Shotgun DNA Library Preparation Method Manual*.

  I. An "Expected bead recovery" value (also known as "Predicted emPCR bead yield") is calculated for each selected lane and provides the number of enriched beads (that will produce key pass reads) one might expect to get from a full emPCR kit (16 reactions) with this library, assuming the performance observed in this lane. See the *GS FLX Shotgun DNA Library Preparation Method Manual*, section 3.7.3.3, for a description of how it is calculated and how to use it to determine the optimal emPCR amplification conditions (DNA concentration) for your library.



TCAG (Library)				Region						
		1	2	3	4	5	6	7	8	Total
	Raw Wells	1,480	2,379	4,586	14,506	2,190	5,149	10,976	16,492	57,758
	Key Pass Wells	86	896	3,069	13,186	1,100	4,237	10,014	14,853	47,441
Failed	Dot	0	18	87	1,138	18	189	808	1,388	3,646
	Mixed	0	7	47	1,196	9	67	324	519	2,169
	Short Quality	14	58	321	2,861	62	642	1,837	3,603	9,398
	Short Primer	2	5	14	184	9	64	159	306	743
	Passed Filter Wells	68	778	2,443	6,471	964	3,125	6,188	7,693	27,730
	% Dot + Mixed	0.00	2.79	4.37	17.70	2.45	6.04	11.30	12.84	12.26
	% Short	18.60	7.03	10.92	23.09	6.45	16.66	19.93	26.32	21.38
	% Passed Filter	79.07	86.83	79.60	49.07	87.64	73.76	61.79	51.79	58.45
Expected bead recovery		27,520	286,720	982,080	4,219,520	-	-	-	-	

Figure 8–23: Summary area from the Filters tab, showing lanes 1 through 4 evaluated as a titration Run, with corresponding entries in the "Expected bead recovery" row. In this example, lane 3 provides the best conditions. Region 2 might appear to be best based on quality criteria (more % Passed Filter and less % Dot + Mixed and % Short), but the Expected bead recovery data shows that it would provide far too few beads. Conversely, the conditions of lane 4 would generate a lot more beads, but their quality would be too low (less than 50% Passed filter and much higher % Dot + Mixed and % Short). This type of assessment DOES NOT APPLY to sequencing Runs carried out using the GS FLX Titanium chemistry.

Data Analysis (Amplicon Variant Analysis)

D

Part D: Data Analysis (Amplicon Variant Analysis)

9. GS Amplicon Variant Analyzer Application 265

9.1 Introduction to the GS Amplicon Variant Analyzer Application 266
9.1.1 Definitions 266
9.1.1.1 Project 266
9.1.1.2 Reference Sequence 266
9.1.1.3 Amplicon and Target 267
9.1.1.4 Read Data Set and Read Group 267
9.1.1.5 Variant 268
9.1.1.6 Sample 268
9.1.1.7 MID and MID Group 269
9.1.1.8 Multiplexer 270
9.1.2 Launching the GS Amplicon Variant Analyzer Application 271
9.1.3 GS Amplicon Variant Analyzer Application Interface Overview 272
9.1.3.1 Main Buttons 272
9.1.3.2 The Tabs and Sub-Tabs 274
9.1.3.3 Buttons and Plots 276
9.1.3.3.1 Scroll Bars 277
9.1.3.3.2 Navigation Buttons 277
9.1.3.3.3 Mousing Functions 278
9.1.3.3.4 Progress bars 279
9.1.3.3.5 Special Action Buttons 279
9.1.3.4 File Browsing in Linux 279
9.2 The Overview Tab 280
9.3 The Project Tab 281
9.3.1 The "Project Tree" Sub-Tabs 283
9.3.1.1 The References Tree 286
9.3.1.2 The Read Data Tree 287
9.3.1.3 The Samples Tree 289
9.3.1.4 The MIDs Tree 290
9.3.2 The "Definition Table" Sub-Tabs 291
9.3.2.1 The References Definition Table 296
9.3.2.1.1 To Enter or Edit the DNA Sequence of a Reference Sequence 297
9.3.2.2 The Amplicons Definition Table 297
9.3.2.2.1 To Enter or Edit the Reference Sequence to which an Amplicon is associated 298
9.3.2.2.2 To Enter or Edit the Primer Sequences for the Amplicon 299
9.3.2.2.3 To Enter or Edit the Target Start and End Positions 300
9.3.2.3 The Read Data Definition Table 302
9.3.2.3.1 To Edit the Read Group of a Read Data Set 303
9.3.2.3.2 To Edit the "Active" status of a Read Data Set 303
9.3.2.4 The Samples Definition Table 304
9.3.2.5 The Variants Definition Table 305
9.3.2.5.1 To Enter or Edit the Reference Sequence to which a Variant is associated 306
9.3.2.5.2 To Enter or Edit the Pattern of a (Known) Variant 306
9.3.2.5.3 To Edit the Status of a Variant 309
9.3.2.6 The MIDs Definition Table 310
9.3.2.6.1 To Enter or Edit the DNA Sequence of an MID 312
9.3.2.6.2 To Edit the MID Group of an MID 313
9.3.2.7 The Multiplexers Definition Table 314
9.3.2.7.1 To Enter or Edit the Sample Encoding using Multiplexers 315
9.3.2.7.1.1 "Primer 1 MID" and "Primer 2 MID" Encoding 316
9.3.2.7.1.2 "Both" Encoding 316
9.3.2.7.1.3 "Either" Encoding 317
9.3.2.7.2 To Enter or Edit the Primer 1 MIDs and Primer 2 MIDs 317
9.3.2.7.3 To Enter or Edit the Samples Assignment 319
9.3.2.7.3.1 Sample Assignment with "Primer 1 MID" or "Primer 2 MID" Encoding 320
9.3.2.7.3.2 Sample Assignment with "Both" Encoding 321
9.3.2.7.3.3 Sample Assignment with "Either" Encoding 322
9.3.2.7.4 Using Multiplexers for more than one Read Data 325
9.4 The Computations Tab 326

Part D: Data Analysis (Amplicon Variant Analysis), continued

9.5 **The Variants Tab** 330
9.5.1 The Variants Frequency Table 331
9.5.1.1 General organization 331
9.5.1.2 Organizing Data in the Variants Frequency Table 333
9.5.1.2.1 Sort options 334
9.5.1.2.2 Ignore filters 335
9.5.1.2.3 Show filters 335
9.5.1.2.4 Option reversions 335
9.5.1.3 Populating the Global Align Tab from the Variants Tab 336
9.5.1.4 Defining a Haplotype from the Variants Tab 336
9.5.1.5 Editing/Removing Variants from the Variants Tab 338
9.5.1.6 The Mouse Tracker 338
9.5.2 Variant Data Display Controls 339
9.5.2.1 The Alignment Read Type Controls 339
9.5.2.2 The Show Values Controls 340
9.5.2.3 The Min / Max Filters 341
9.5.2.4 The Variant Status Filter 342
9.5.2.5 The Compact Table Checkbox 342
9.5.2.6 The Auto-Detected Variant Load Button 342
9.5.2.7 Variant Discovery Workflow 344
9.6 **The Global Align Tab** 346
9.6.1 Populating the Global Align Tab 347
9.6.2 The Variation Frequency Plot 348
9.6.3 The Multiple Alignment Display 349
9.6.3.1 The Reference Sequence 349
9.6.3.2 The Multi-Alignment 350
9.6.3.3 Special Function Buttons 352
9.6.4 Display Option Tools 354
9.6.4.1 Alignment Data 355
9.6.4.2 Read Type 356
9.6.4.3 Reported Frequency 356
9.6.4.4 Read Orientation 357
9.7 **The Consensus Align Tab** 358
9.7.1 Populating the Consensus Align Tab 358
9.7.2 The Variation Frequency Plot 359
9.7.3 The Multiple Alignment Display 359
9.7.4 Display Option Tools 359
9.8 **The Flowgrams Tab** 360
9.8.1 Populating the Flowgrams Tab 362
9.8.2 The Tri-Flowgram Plot 362
9.8.3 Navigation on the Flowgrams Tab 363

10. **Example Amplicon Project Design and Analysis** **364**

10.1 **Experimental Design** 365
10.2 **Project Setup in the AVA Software** 368
10.2.1 Launching the AVA Application 369
10.2.2 Creating a New Project 370
10.2.3 Defining the Reference Sequence 372
10.2.4 Defining the Amplicons 374
10.2.5 Defining the Sample 376
10.2.6 Defining the Known Variant 377
10.2.7 Importing the Read Data Set 379
10.3 **Analysis of Known Variants** 381
10.3.1 Compute the Project 381
10.3.2 Frequency of Known Variants 382
10.4 **Mining a Project for New Variants** 389

Part D: Data Analysis (Amplicon Variant Analysis), continued

10.5 Important Factors in the Assessment of New Variants 404
10.5.1 Above the Noise 404
10.5.2 Coverage 404
10.5.3 Bidirectional Support 404
10.5.4 Homopolymers 404
10.5.5 Flowgram Evidence 405
10.5.6 Read Length 405
10.6 Other Issues of Special Interest 406
10.6.1 What Does 'Sample' Mean? 406
10.6.2 How Should Your Project Be Organized? 407
10.6.3 Should Amplicons Share a Reference Sequence or Have Individual Ones? 407
10.6.4 When should MIDs be used? 408
10.6.5 What is the purpose of Multiplexers? 408
10.6.5.1 Non-Multiplexer Example 409
10.6.5.2 Multiplexer Example 410
10.6.5.3 Multiplexer Benefits Summary 412

11. GS Amplicon Variant Analyzer Command Line Interface 413

11.1 Purpose of the CLI 413
11.1.1 Data Import 413
11.1.2 Data Export 414
11.1.3 Automating the Triggering of Computations 414
11.1.4 Result Reporting 414
11.2 AVA-CLI Command Language Overview 415
11.2.1 Entities 415
11.2.2 Available Commands 415
11.3 AVA-CLI General Online Help 417
11.3.1 Help 417
11.3.2 General Help 417
11.3.2.1 CommandLine Help 418
11.3.2.2 Parsing Help 419
11.3.2.3 Tabular Commands Help 420
11.3.2.4 Record Names Help 422
11.3.2.5 Abbreviations Help 422
11.3.2.6 File Paths Help 423
11.3.2.7 Multiplexing Help 424
11.4 AVA-CLI Command Usage Statements 425
11.4.1 associate 425
11.4.2 close 428
11.4.3 computation 429
11.4.3.1 computation start 429
11.4.3.2 computation stop 429
11.4.3.3 computation status 429
11.4.3.4 computation loadDetectedVariants 429
11.4.4 create 430
11.4.4.1 create amplicon 430
11.4.4.2 create mid 431
11.4.4.3 create midGroup 432
11.4.4.4 create multiplexer 432
11.4.4.5 create project 433
11.4.4.6 create readGroup 433
11.4.4.7 create reference 434
11.4.4.8 create sample 434
11.4.4.9 create variant 435
11.4.5 dissociate 436
11.4.6 exit 438

Part D: Data Analysis (Amplicon Variant Analysis), continued

11.4.7 list ... 438
11.4.7.1 list amplicon ... 439
11.4.7.2 list mid ... 439
11.4.7.3 list midGroup ... 439
11.4.7.4 list multiplexer ... 440
11.4.7.5 list project ... 440
11.4.7.6 list readData ... 440
11.4.7.7 list readGroup ... 441
11.4.7.8 list reference ... 441
11.4.7.9 list sample ... 441
11.4.7.10 list variant ... 442
11.4.8 load ... 442
11.4.9 open ... 443
11.4.10 remove ... 444
11.4.10.1 remove amplicon ... 444
11.4.10.2 remove mid ... 444
11.4.10.3 remove midGroup ... 445
11.4.10.4 remove multiplexer ... 445
11.4.10.5 remove readData ... 445
11.4.10.6 remove readGroup ... 445
11.4.10.7 remove reference ... 446
11.4.10.8 remove sample ... 446
11.4.10.9 remove variant ... 446
11.4.11 rename ... 447
11.4.11.1 rename amplicon ... 447
11.4.11.2 rename mid ... 447
11.4.11.3 rename midGroup ... 448
11.4.11.4 rename multiplexer ... 448
11.4.11.5 rename project ... 448
11.4.11.6 rename readData ... 448
11.4.11.7 rename readGroup ... 448
11.4.11.8 rename reference ... 449
11.4.11.9 rename sample ... 449
11.4.11.10 rename variant ... 449
11.4.12 report ... 450
11.4.12.1 report variantHits ... 450
11.4.13 save ... 450
11.4.14 set ... 451
11.4.14.1 set verbose ... 451
11.4.14.2 set onErrors ... 451
11.4.14.3 set currDir ... 451
11.4.15 Show ... 452
11.4.15.1 show environment ... 452
11.4.16 update ... 452
11.4.16.1 update amplicon ... 453
11.4.16.2 update mid ... 454
11.4.16.3 update midGroup ... 455
11.4.16.4 update multiplexer ... 455
11.4.16.5 update project ... 456
11.4.16.6 update readData ... 456
11.4.16.7 update readGroup ... 456
11.4.16.8 update reference ... 457
11.4.16.9 update sample ... 457
11.4.16.10 update variant ... 458
11.4.17 utility ... 458
11.4.17.1 utility validateNames ... 459
11.4.17.2 utility validateForComputation ... 459
11.4.17.3 utility makeSetupScript ... 460
11.4.17.4 utility clone ... 460
11.4.17.5 utility execute ... 461

Part D: Data Analysis (Amplicon Variant Analysis), continued

11.5 **Creating and Computing a Project with the AVA-CLI** 462
11.5.1 Setting CLI Parameters 462
11.5.2 Creating a New Project 463
11.5.3 Creating References 463
11.5.4 Creating Amplicons 466
11.5.5 Creating Variants 468
11.5.6 Creating Samples 469
11.5.7 Associating Samples with Amplicons 469
11.5.8 Loading Read Data Sets 471
11.5.9 Associating Read Data Sets with Samples 472
11.5.10 Editing Object Properties 473
11.5.10.1 Updating an Object 473
11.5.10.2 Renaming an Object 474
11.5.10.3 Removing an Object 474
11.5.10.4 Dissociating Relationships 475
11.5.11 Computation 476
11.5.11.1 Validating the Project before Computation 476
11.5.11.2 Managing the Computation 477
11.5.11.3 Loading Automatically Detected Variants 478
11.5.12 Reporting 478
11.5.13 Finishing Touches 479
11.5.13.1 save 479
11.5.13.2 close 479
11.5.13.3 exit 479
11.5.14 Exporting from the Project 480
11.5.14.1 utility makeSetupScript 480
11.5.14.2 utility clone 480
11.5.14.3 list 481
11.5.15 Integrated Project Script 481
11.6 **Creating and Computing an MID Project with the AVA-CLI** 485
11.6.1 Example MID Project Script 487

12. **Amplicon Variant Analyzer – Special Topics** **490**

12.1 **Addressing Simultaneous Multiple Users' Access to an Amplicon Project** 490
12.2 **Intelligent Variant Naming** 493
12.2.1 Tier 1 Naming 493
12.2.2 Tier 2 Naming 494
12.2.3 Tier 3 Naming 494
12.2.4 Tier 4 Naming 495
12.2.5 Naming Example 495
12.3 **Properties Windows for Global and Consensus Alignments** 496
12.3.1 When is the Properties Information Useful? 496
12.3.2 Content of the Three "Properties" Window Types 496
12.3.2.1 Properties Window for a Consensus 497
12.3.2.2 Properties Window for a Forward Read 497
12.3.2.3 Properties Window for a Reverse Read 498
12.4 **Automatic Project Initialization in the GUI** 499
12.4.1 Default Initialization Script Location 499
12.4.2 Default Initialization Script Contents 500
12.4.2.1 Step 1: Loading the "Standard" 454 MIDs 500
12.4.2.2 Step 2: Running User-Customized Initialization Functions 500
12.4.3 Initialization Script Restrictions 501
12.4.4 Initialization Script Error Handling 501
12.5 **Project Initialization and the CLI** 502
12.6 **Multiplex Amplicon Libraries** 503

Important Note: October 2008 marks the first release of the new GS FLX Titanium series chemistry for the Genome Sequencer FLX System. For the time being, the system supports only the non-MID "General" (*e.g.* Shotgun) sequencing applications under the GS FLX Titanium chemistry. For Paired End or Amplicon sequencing, or for the preparation and sequencing of MID libraries of any type, users must continue to use the GS FLX standard series kits and procedures (last updated in December 2007).

However, the Genome Sequencer FLX Software version 2.0.00, associated with the October 2008 release, is fully backward-compatible with, and can process datasets generated with any of the Genome Sequencer System's chemistries (GS 20 chemistry, GS FLX standard chemistry, and GS FLX Titanium chemistry). This manual describes **all** the functionalities of the 2.0.00 software, even those that apply **only** to datasets generated with older chemistries, such that **it completely replaces** the December 2007 issue of the *Genome Sequencer FLX Data Analysis Software Manual* (which addressed the software version 1.1.03).

The Genome Sequencer FLX Titanium series manuals are easily identified by their new cover graphics and distinctive tri-color stripes (reflecting the GS FLX Titanium kits packaging). All the methods, protocols and applications supported on the GS FLX standard chemistry will be enabled on the GS FLX Titanium chemistry in the near future.

Features supported only under the GS FLX standard chemistry: While the GS Amplicon Variant Analyzer software v. 2.0.00 provides important new features, such as the support for MIDs with Amplicon libraries, those libraries are not currently supported under the GS FLX Titanium chemistry. Therefore, the entire content of Part D of this manual, including the new features, applies only to Amplicon libraries prepared under the GS FLX standard chemistry.

9. GS Amplicon Variant Analyzer Application

- This section describes how to use the GS Amplicon Variant Analyzer (AVA) application through its Graphical User Interface (GUI). The AVA software also features a Command Line Interface (CLI) that may be more appropriate for large Projects, especially when large amounts of data need to be imported into, exported from, or automated within a Project. See section 11 for a full description of the CLI, the language that was developed for it, and all the commands it includes.
- Projects are compatible with each other regardless of whether they are set up or computed using the Graphical User Interface (GUI) or the Command Line Interface (CLI). For certain projects, some may find it useful to set up portions of the project definition using the CLI and then enter the GUI for all subsequent tasks.

The sequencing results of Amplicon libraries are designed primarily to identify and quantitate both known and novel DNA variants (*e.g.* rare alleles) by the Ultra Deep Sequencing coverage of one or more region(s) of interest. This is supported by the "GS Amplicon Variant Analyzer" (AVA) software described in this section.

Briefly, the AVA application computes the alignment of reads from Amplicon libraries obtained on the Genome Sequencer FLX Instrument, and identifies differences between the reads and a reference sequence. Variations are displayed both graphically with a histogram indicating positions of variation, and textually with a color-coded multiple alignment that regions and bases that differ from the reference sequence.

The software specifically reports the frequency of user-defined and software-identified variants in a summary Table, allowing for the high-throughput detection and quantitation of known and putative variants in the samples sequenced. In addition, various tools and views allow the user to examine the read alignments in detail to assess whether the software-identified variants appear to be legitimate, and possibly identify new ones. The user can then define these new variants into the system, and decide which variants to include in the analysis for quantitative reports.

9.1 Introduction to the GS Amplicon Variant Analyzer Application

9.1.1 Definitions

A few important terms have special meanings or characteristics in the context of the AVA software. These are defined and described below.

9.1.1.1 Project

An Amplicon **Project** is the main container of an Amplicon Sequencing experiment. In it, you specify the Reference Sequence(s) to which the sequencing reads will be compared, in search of Variants; the Amplicon(s) that constitute the library(ies) you sequenced [and hence, the reads in the Read Data Set(s)]; the Variant(s) that you specifically want the software to search and report on; and the Sample(s) that constitute the organizational basis for the analysis. If the Amplicon library(ies) contain Multiplex Identifiers (MIDs), the Project should further specify the MIDs used and Multiplexers to define the relationship between MIDs and Samples. All these terms correspond to "elements" that constitute the Amplicon Project, and are further defined in the following sub-sections. The Project format allows the user to incrementally add new information (Read Data Sets, of course, but also Sample, Amplicon, Variant and even new Reference Sequence or MID/Multiplexer definitions) to a Project, *e.g.* as the sequencing results from new Runs/regions become available.

9.1.1.2 Reference Sequence

The basic definition of a **Reference Sequence** is quite straightforward: it is simply a string of A, T, G, C (or N) characters representing a DNA sequence against which the sequencing reads will be aligned and compared so variations can be identified and reported. The Reference Sequence(s) also provide the coordinates used to localize other elements defined in the Project (Amplicons and Variants; each Reference Sequence starts at coordinate "1"). You can define any number of Reference Sequences in a Project.

It is important to note that only "nucleotide" characters (A, T, G, C, or N) are accepted when you enter a Reference Sequence into the AVA software (by typing or pasting). For convenience, when pasting sequences, characters that are not nucleotide characters and are also not IUPAC ambiguity characters (such as R for purine, Y for pyrimidine, *etc.*) are <u>removed</u> from the pasted entry. This is useful when pasting sequences from sources that may include non-sequence information (such as white space or numerical position information in the margin of each line). During such pastes, any IUPAC ambiguity characters are converted to "N" characters, as the other ambiguity characters are not supported by the software (typing individual "ambiguous" characters, however, does not result in their conversion to "N"; these are simply ignored and the text "Only ATGC and N" at the top of the Edit Sequence window turns bold and red to alert you that an invalid character was used). The restriction that no ambiguity characters other than N be present in a sequence is a requirement of many alignment algorithms and is not unique to the Genome Sequencer System software.

It is also important to be aware that shorter Reference Sequences are more efficient for computation in the AVA software, whereas a long Reference Sequence could result in unnecessarily long computation times and slow navigation and scrolling in the application's windows. Since in Amplicon sequencing, the interest is in one or a few small regions of DNA, the user should specify such region(s) when defining the Reference Sequence(s) for a Project rather than entering, for example, the entire genome of the organism. If you want to monitor together multiple targets that are distant from one another in the reference genome (for example exons of a given gene), you can create an "artificial" Reference Sequence by concatenating the segments of interest; it is useful to insert a few "N" characters between the concatenated targets if you create artificial Reference Sequences.

9.1.1.3 Amplicon and Target

The term **Amplicon** is used in the AVA software to represent essentially the same entity (sequence) as in the preparation of an Amplicon library, except that it does not include the 19 bp "Primer A" and "Primer B" parts of the Fusion Primers (see the *GS FLX Amplicon DNA Library Preparation Method Manual* for details on the structure of sequencing templates from Amplicon libraries). As such, therefore, they match the sequencing reads from the Read Data Set(s).

In the AVA software, however, an Amplicon is a virtual entity defined *relative to a Reference Sequence* by specifying two primers (the "template-specific" parts of the Fusion Primers). This relative definition is also *directional*: the AVA software names the two template-specific primers "Primer 1" and "Primer 2" in the 5'-Primer 1 $\rightarrow$ Primer 2-3' orientation of the Reference Sequence. Therefore, Amplicon orientation is internal to the AVA software, and is NOT dependent upon the "Primer A" and "Primer B" parts of the Fusion Primers used in library construction.

You can define any number of Amplicons in a Project, each associated with a specific Reference Sequence; you can also associate multiple Amplicons, even overlapping ones, with a given Reference Sequence. Thus, a Reference Sequence may be associated with multiple Amplicons, but an Amplicon may only be associated with one Reference Sequence. Amplicons are also associated with Read Data Sets and with Samples (see below).

The term **Target** specifies the part of an Amplicon that is between the two primers (*i.e.*, the non-primer portion of the Amplicon). This is the sequence that is actually aligned to the Reference Sequence during the computations. It is important to trim the primers before alignment because any variant found therein would be a reflection of primer design (or errors in primer synthesis) rather than representing variations in the DNA sample used to prepare the Amplicon library, and therefore would not have any biological significance.

9.1.1.4 Read Data Set and Read Group

A **Read Data Set** is a group of sequencing reads derived from an Amplicon library. In a Project, Read Data Sets exist within a Read Group (this helps to organize the data) and are associated with pairings of Amplicons and Samples:

- the Amplicon association specifies which Amplicon(s) were included in the sequencing Run that produced this Read Data Set; the reads are identified as belonging to an Amplicon by virtue of their template-specific primers (see section 9.1.1.3, above)
- the Sample association specifies in which Sample(s) to report the results of the computations (see section 9.1.1.6, below, for a more detailed explanation of "Samples").

You can include any number of Read Data Sets in a Project; and associate them with any number of Amplicon-Sample pairs. However, an Amplicon cannot be associated with more than one Sample within a given Read Data Set unless MIDs are used to further associate the reads with specific Samples; see section 9.1.1.7 for more details on this. Note also that the AVA software can only process reads from Amplicon libraries.

In the current release of the AVA software, a Read Data Set is equivalent to an SFF file, *e.g.* as output by the data processing pipeline of the Genome Sequencer System: there will usually be one Read Data Set for each PicoTiterPlate region of each Amplicon sequencing Run you import into the Project (unless you reorganized your SFF files using the SFF Tools). See sections 3.2.4.2 and 13.3.8 in this manual for details on SFF files, section 7 for details on the SFF Tools commands, and section 9.3.2.3 for a note specifying certain restrictions on the use of SFF files that have been manipulated by SFF Tools.

9.1.1.5 Variant

Simply put, a **Variant** is a sequence difference relative to a Reference Sequence. Like Amplicons, Variants are thus defined *relative to a Reference Sequence*. Four kinds of variations can be defined in the AVA software: substitutions, deletions, insertions, and required matches; and a defined Variant can include any number of these, in any combination (haplotypic variations). You can define any number of Variants in a Project, each associated with a specific Reference Sequence; you can also associate any number of Variants to a given Reference Sequence.

Though the multiple alignment views of the AVA software show all variations between the reads displayed and their Reference Sequence, a Variant must be defined in the Project to be reported in the application's Variants tab. Known Variants (*e.g.* from the scientific literature) can be defined directly in a Project. Also, putative substitution and deletion Variants will be automatically identified and defined by the AVA software if they are detected at a preset minimum abundance during computation of the Project; you can examine the alignments of these putative Variants in detail, allowing you to formally "accept" them as legitimate Variants or "reject" them as noise. You can also define new Variants based on variations observed between the Reference Sequence(s) and the reads included in your Project.

The Variants tab thus reports statistics on the observed incidence (in all the reads included in the last computation of the Project) of each defined Variant, broken out by Sample. A Variant's definition specifies one or more Reference Sequence positions whose nucleotide identity must be matched or mutated in some way. Only reads that, in their multiple alignment to the Reference Sequence, span the entire set of Variant positions are eligible to contribute to the statistics computed for that Variant.

9.1.1.6 Sample

The term **Sample**, in the context of the AVA software, can be defined very generically as a virtual "container" specified by the user only as a name (and an optional annotation), and used to group reads for analysis and reporting. The Samples thus represent the organizational foundation for the analysis, whose primary output is the Variants tab, such that the frequency of any or all defined Variants can be compared between the different "Samples" defined in the Project. You can define any number of Samples in a Project, each associated with one or more Read Data Sets and with one or more Amplicons. For example, Samples could correspond to sequencing data from an Amplicon library prepared from a "control" DNA sample; and those associated with a second Sample, to a library prepared from the DNA of an "experimental" tissue or individual. Or, different Samples could correspond to multiple replicate libraries of a biological sample, *e.g.* to allow for statistical comparison between them.

Within a Read Data Set, reads may correspond to one or more Samples. In order to demultiplex the reads, *i.e.* assign them each to the proper Sample, the reads must contain reliably identifiable Sample-specific features. The AVA software can use either of two mechanisms to assign reads to Samples:

1. It can use the known template-specific part of the Adaptor used to prepare the Amplicon library. This works well when the Amplicon identity alone is sufficient to assign the reads to Samples; this restricts one to the case where any given Amplicon within a Read Data Set provides reads for only one Sample (though it allows reads from different Amplicons to belong to the same Sample.)
2. It can use Multiplex Identifiers (MIDs) in conjunction with the template-specific part of the Adaptor. This is required if reads from a given Amplicon need to be assigned to more than one Sample in the same Read Data Set; the MIDs then provide the additional context necessary to resolve the reads to the appropriate Samples.

To perform the read to Sample assignments, the AVA software relies on user-specified, three-way associations between Read Data Sets – Samples – Amplicons (first mechanism), or Read Data Sets – Multiplexers – Amplicons (second mechanism). In the second case, the Multiplexers (see sections 9.1.1.7 and 9.1.1.8) provide the MID to Sample assignment information. Within one Read Data Set, a given Amplicon cannot belong to more than one such three-way association because the software would then be unable to unambiguously determine which association mechanism to use in order to assign reads from that Amplicon to their proper Samples.

Once the read to Sample assignment is made, the AVA software can compute the prevalence of Variants found in the reads, broken out by Sample. These statistics are reported in the Variants tab (section 9.5). Be aware, however, that while you can examine Variant frequency statistics for all the Samples of the Project in the Variants tab, you can view read alignments of only one Sample at a time (*e.g.* in the Global Align tab).

9.1.1.7 MID and MID Group

An **MID** (or Multiplex Identifier) is a short, recognizable sequence tag that can be added to the design of the Adaptors used for library preparation, between the sequencing key and the template-specific primer, to help determine the provenance of the read (see section 12.6). Multiple Amplicon libraries (the Project's Samples) can be prepared that include the same Amplicon target sequences (with the same template-specific primers), each labeled with different MID tags. The MID sequences provide extra context that, in concert with the template-specific primers, allow flexible demultiplexing options, and specifically enable the sequencing of the same Amplicon across multiple Samples within the same Read Data Set (or region of a PicoTiterPlate device): when using MIDs, the Sample-Amplicon associations are indirectly specified in the software by associating Amplicons with Multiplexers (see section 9.1.1.8), which themselves specify the relationship between MIDs and Samples and then apply that information to the associated Amplicons. Note that both non-MID and MID-tagged Amplicons may be used in a Project, but within a given Read Data Set, all the reads for any individual Amplicon must be of one type or the other.

Contrary to the situation with Shotgun (sstDNA) libraries, where an MID sequence can only be on Adaptor A, Amplicon libraries can be constructed with MIDs at either or both ends of the reads. This provides for considerable flexibility in the design of the MID Amplicon libraries. In particular, if the Amplicons are designed such that the read length of the sequencing Run allows full read-through of the Amplicon reads, then placing MIDs at both ends of the reads makes it possible to use them combinatorially, such that a small number of MID tags can encode a much larger number of Samples (per the "Both" encoding; see section 9.1.1.8).

If multiple sets of MIDs are used in a laboratory, it may be useful to define **MID Groups** for each set, allowing them to be referred to as a group. A common grouping may be by length of the MID tags, because there is a restriction that all MIDs used at one end of any given Amplicon be the same length (see section 9.3.2.6). The AVA software is delivered with an MID Group named "454Standard", containing 14 MIDs carefully chosen to be resilient to sequencing and primer synthesis errors.

9.1.1.8 Multiplexer

A **Multiplexer** specifies the association between MIDs and Samples, *i.e.* how the MIDs should be used to assign reads to Samples. Depending on the design of the Amplicon libraries, Multiplexers allow four types of encoding (see section 12.6 for a description of Amplicon library design, in the context of MIDs):

- **Primer 1 MID:** This encoding provides an MID signature only on the end of the read that contains the template-specific primer defined as "Primer 1" in the Project. This will be at the beginning of the "forward" reads, or at the end of "reverse" (complemented) reads. These MIDs are then used to assign the reads to the proper Sample, as defined by the Multiplexer.
- **Primer 2 MID:** This encoding is the same as Primer 1 MID encoding, except that the MID appears at the "Primer 2" end of the Amplicons.
- **Both:** This encoding provides MIDs at both ends of the Amplicons and requires that read length be sufficient to read through to the distal MID, in both orientations. The *paired combination* of MIDs located on the Primer 1 and Primer 2 sides is used to assign reads to their proper Sample, as defined by the Multiplexer.
- **Either:** This encoding also provides MIDs at both ends of the Amplicons, but assigns the reads to their proper Sample on the basis of only the proximal MID on the read, in either orientation. This allows for proper assignment of both forward and reverse reads even if the Amplicon is longer than the read length provided by the sequencing Run script. Note that even if full read-through to the distal end of the read is possible, only the proximal MID will be used for Sample assignment (and any contradiction between the MIDs seen at the two ends will be assumed to be the effect of sequencing artifacts at the distal end of the read).

Selecting the proper encoding: It is crucially important to select the encoding method that truly corresponds to the way the libraries were prepared. For example, if a library was prepared with the 'Either' chemistry in mind, it may be tempting to use a 'Primer 1 MID' or 'Primer2 MID' encoded Multiplexer since the distal MID gets discounted in favor of the proximal MID, in 'Either' encoding. However, the AVA software needs to know that MIDs are expected to be found at both ends: without that knowledge, the trimmer might get a suboptimal alignment of the distal primer, which in certain cases could drop valid reads out of the analysis.

Multiplexers specify the assignment of reads that contain each defined MID (or MID pair) to each specific Sample, *within a Read Data Set*. Different Amplicons within a Read Data Set may simultaneously be sequenced even if they use different Multiplexer encoding methods, or no encoding at all (*i.e.* are sequenced without the use of MIDs), but any given Amplicon can only be sequenced in a single manner within a given Read Data Set. In the software, Multiplexers are associated with Read Data Sets and then one or more Amplicons are associated with those Multiplexers, in the context of the Read Data Sets (creating Read Data Sets – Multiplexers – Amplicons triads). The software then assigns the reads from those Amplicons to Samples according to the rules of the Multiplexer encoding. Operationally, the same restriction exists regarding the association of Amplicons to Multiplexers as exists regarding the association of Amplicons to non-MID Samples (see section 9.1.1.6): a given Amplicon cannot belong to more than one Multiplexer within one Read Data Set, because the software would then be unable to unambiguously resolve which Multiplexer to use to determine the proper Sample assignment for the Amplicon reads.

Multiplexers conveniently encapsulate the correspondence between MIDs and Samples. Without Multiplexers, each instance of an Amplicon in a Project, distinguished from one another only by a choice of different MIDs in their library preparation, would require that a separate Amplicon be defined in the Project. Multiplexers also allow the correspondence between MIDs and Samples to be specified only once and shared across multiple Amplicons that may be sequenced simultaneously, a common experimental design. These and other benefits of using Multiplexers, including the more accurate decoding of MIDs and the reduction of errors in Sample assignment during the demultiplexing phase of computation, are further described in section 10.6.5.

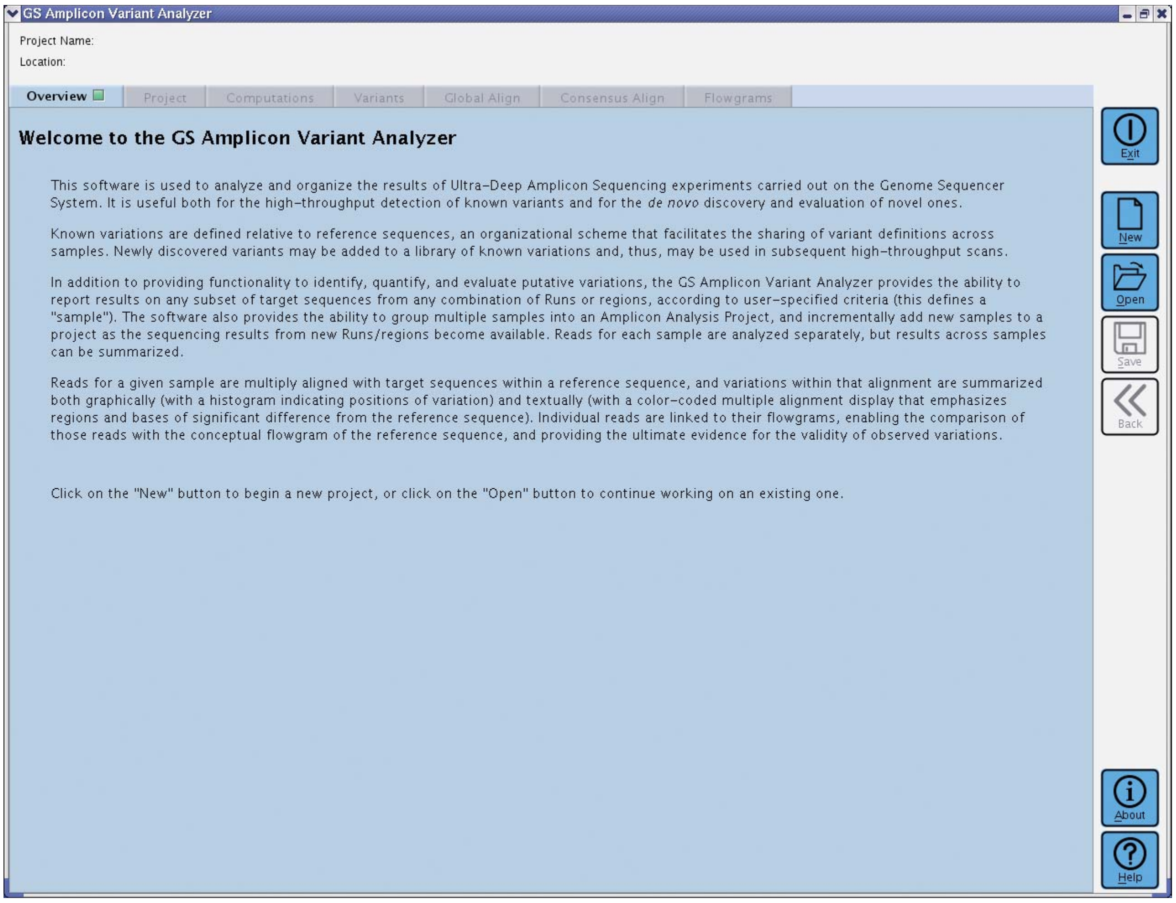
9.1.2 Launching the GS Amplicon Variant Analyzer Application

The GS Amplicon Variant Analyzer application consists of a single command, whose command line structure is the following:

```
gsAmplicon
```

gsAmplicon command interruption: If you interrupt the gsAmplicon command, accidentally or otherwise, by typing control-C, the application will immediately shutdown without giving you a chance to save any of your recent Project changes. For additional safety, you may prefer to start the command in the background, ending the command line with a '&', as in: "`gsAmplicon &`". Other modes of exiting the application, such as clicking the "Exit" button, would elicit warning dialogs if the Project contains any unsaved changes.

This will open the AVA application main window in its Overview tab (Figure 9–1). [A splash screen identifying the application and its version number will be displayed briefly (not shown); you can also view this splash screen at any time, after launching the application, by clicking on the "About" button.] Until a Project is open, the Overview tab provides a brief textual description of the AVA application's usage and capabilities. There are two ways to open a Project at this point: you can create a new Project by clicking on the "New" button, or you can open an existing Project by clicking on the "Open" button (see section 9.1.3.1). Note that if you do not have write-permission to the Project folder, or if another user already has it open, you can open the Project as "Read-Only", but you will be unable to save any changes or carry out any computations (which requires writing to disk). See section 12.1 for more details on this kind of situation.



Project Name:
Location:

Welcome to the GS Amplicon Variant Analyzer

This software is used to analyze and organize the results of Ultra-Deep Amplicon Sequencing experiments carried out on the Genome Sequencer System. It is useful both for the high-throughput detection of known variants and for the *de novo* discovery and evaluation of novel ones.

Known variations are defined relative to reference sequences, an organizational scheme that facilitates the sharing of variant definitions across samples. Newly discovered variants may be added to a library of known variations and, thus, may be used in subsequent high-throughput scans.

In addition to providing functionality to identify, quantify, and evaluate putative variations, the GS Amplicon Variant Analyzer provides the ability to report results on any subset of target sequences from any combination of Runs or regions, according to user-specified criteria (this defines a "sample"). The software also provides the ability to group multiple samples into an Amplicon Analysis Project, and incrementally add new samples to a project as the sequencing results from new Runs/regions become available. Reads for each sample are analyzed separately, but results across samples can be summarized.

Reads for a given sample are multiply aligned with target sequences within a reference sequence, and variations within that alignment are summarized both graphically (with a histogram indicating positions of variation) and textually (with a color-coded multiple alignment display that emphasizes regions and bases of significant difference from the reference sequence). Individual reads are linked to their flowgrams, enabling the comparison of those reads with the conceptual flowgram of the reference sequence, and providing the ultimate evidence for the validity of observed variations.

Click on the "New" button to begin a new project, or click on the "Open" button to continue working on an existing one.

Figure 9–1: The Overview tab showing the textual description of the application, before a Project is open

9.1.3 GS Amplicon Variant Analyzer Application Interface Overview

The top of the main AVA window shows the name and path of the Amplicon Project currently displayed; and a set of seven main buttons are located along the right-hand side of the window. The rest of the AVA window contains a selection of tabs, described in the sections below.

- Make sure to set the resolution of your computer screen to at least 1024×768 pixels (1280×1024 recommended), or the AVA application may not be able to display all the features described below. If the resolution is too low (or if the application window or one of its constituent tabbed panels are resized too small), the software attempts to prioritize which feature to omit. For example, the Global Align tab may not show the color code legend on its lower-left corner; or the button column to the left of the Variation Frequency Plot on the top panel of the Global Align or the Consensus Align tabs may not show the "Save plot data to .xls (spreadsheet) file" and/or the "Save plot snapshot to .png file" buttons. While program features may be omitted in this way, scrollbars can be used as needed to ensure that all data is viewable regardless of the screen resolution or window size.
- The AVA software also features a Command Line Interface (CLI) that may be more appropriate for large Projects, especially when large amounts of data need to be imported into, exported from, or automated within a Project. See section 11 for a full description of the CLI, the language that was developed for it, and all the commands it includes.

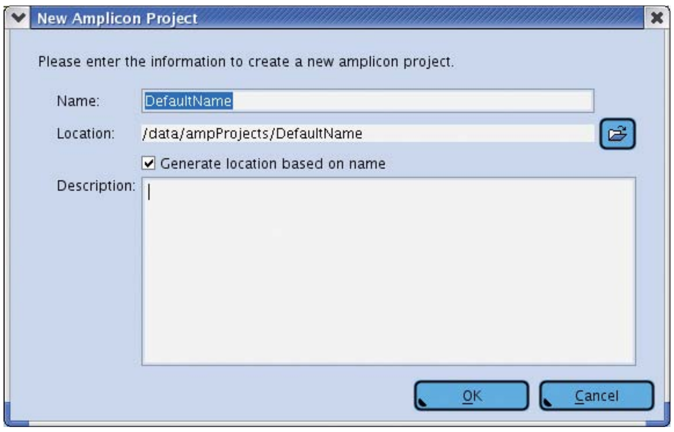
9.1.3.1 Main Buttons

Seven main buttons are always visible, along the right-hand side of the GS Amplicon Variant Analyzer window (Save and Back are grayed-out when their function is not applicable, *e.g.* no Project changes to save):

Icon	Description
Exit	The **Exit** button closes the AVA application.
New	The **New** button opens a "New Amplicon Project" window in which you can provide a name for a new Project, as well as a file-system location in which to save it and a free-text description (Figure 9–2). Clicking "OK" in the "New Amplicon Project" window initializes the Project and takes you to the Project tab of your new Project. For more details on Project initialization, see section 12.4. (See section 10.2.2 for more details on ways to create a new Project, especially synchronizing the Project and Location names, and navigating your file-system using the "folder" icon to the right of the 'Location' field.)
Open	The **Open** button allows you to browse your file-system to find an existing Amplicon Project and open it in the AVA application.
Save	The **Save** button saves the current state of the Project to the disk, *i.e.* all the primary elements and their associations.
Back	The **Back** button takes you back to the previous AVA view. (Note that this button does not carry an "undo" function; see Note, below.)
About	The **About** button opens a splash screen providing some information about the AVA application. (Click the "Close" button to close it).
Help	The **Help** button provides online access to help topics, organized by tab. (Not currently active.)

The AVA software does not support the possibility to "undo" an action such as a computation, an element definition entry, a display selection in a multi-alignment view, *etc*. In some cases, the opposite action or a "clear" action may be available. You can also revert to the last saved state of the Project by re-opening the Project without first clicking the "Save" button.

Saving *vs.* computing a project: Saving a Project does NOT update the results displayed in the Variants, Global Align, Consensus Align or Flowgrams tabs. To update these displays after making changes in a Project, you must re-compute it. Conversely, these results do not require that you save the Project to persist in the Project.


New Amplicon Project

Please enter the information to create a new amplicon project.

Name: DefaultName

Location: /data/ampProjects/DefaultName

☑ Generate location based on name

Description:

Figure 9–2: The New Amplicon Project window, with fields to enter a Project's name, file-system location, and textual description. If the "Generate location based on name" box is checked, typing a Project's name in the Name field also enters it at the end of the path in the Location field. Be aware that later changing the name of the Project from within the application will NOT change the name of the folder that contains it, causing a mismatch between the two; a mismatch would also occur if the "Generate location based on name" box is not checked, and you type different names for the Project (in the Name field) and the folder (at the end of the path). If there is a problem with the selected location, the OK button will be disabled, the location field will be highlighted in red, and, if you position the mouse over the location field, a screen tip will appear indicating the nature of the problem. The button to the right of the Location field allows you to browse your file-system to set a location path. See section 10.2.2 for an example and additional details on ways to create a new Project in the AVA software.

9.1.3.2 The Tabs and Sub-Tabs

The AVA application displays the various aspects of the Amplicon Project in a series of 7 tabs, with the Project tab separated into two panels (themselves comprising 4 and 7 sub-tabs, respectively). When a tab has no content, its name is grayed out and the tab is unavailable. When a tab does have content, a green square icon appears next to its name; clicking on an available tab or sub-tab name brings the information it contains to the front for viewing, as listed below. If the size of your screen does not allow you to view all the tabs, a pair of arrow buttons in each panel allows you to scroll the set of tabs to bring hidden ones into view. The contents and usage of the information included in the tabs are described in full detail in sections 9.2 through 9.8.

- The **Overview** tab provides a basic summary of the Amplicon Project.
- The **Project** tab is used to set up the Project (define all the elements that compose it and their associations) and to navigate it and select particular Sample-Amplicon pairs to view in the Global Align tab. The Project tab contains two panels:
  - The left panel comprises 3 sub-tabs that show various representations of the Project in “tree” form, thus displaying the associations between the various elements that compose it.
  - The right panel comprises 7 sub-tabs that list and show all the characteristics of the various “elements” defined in the Project, in tabular form.
- The **Computations** tab allows the user to compute (or re-compute) the Project, and lists the progress and state of each computation step.
- The **Variants** tab provides summary results of the GS Amplicon Variant Analyzer, listing the observed frequency of each defined Variant in each relevant Sample. A Sample is relevant to a Variant when the Read Data Set(s) with which the Sample is associated contain reads that cover all the Reference Sequence positions specified in the Variant’s definition.
- The **Global Align** tab is populated with the multiple alignment (against the appropriate Reference Sequence) of the reads corresponding to one or more selected Sample-Amplicon pair(s). To select a Sample-Amplicon pair, right-click on an appropriate object in any of the Project Trees or in the Variants tab (only one Sample-Amplicon pair can be selected this way), or use a navigation dialog found within the Global Align tab itself (multiple pairs selection possible). The reads displayed may be <u>Individual</u> reads, corresponding to sequence reads directly extracted from the Read Data Set(s), or <u>Consensus</u> reads, corresponding to sets of Individual reads that were collapsed into a single representative read (in order to simplify the display and eliminate noise from the data). In either case, the tab comprises two data panels:
  - The top panel is a stacked histogram indicating the frequency of all the variations observed between the selected reads and the Reference Sequence associated with the Amplicon(s) from which the reads were derived. The histogram is based on a gapped multiple alignment and, thus, may contain data for positions of the alignment that occur between positions of the Reference Sequence itself. This graph also shows depth of coverage for each position of the alignment.
  - The bottom panel displays the (gapped) sequence multi-alignment of the reads, aligned below the Reference Sequence. Color codes help to highlight variations in the reads in comparison with the Reference Sequence; the reads (Individual or Consensus) can be filtered for display to help identify specific variations and potentially discover haplotypes.

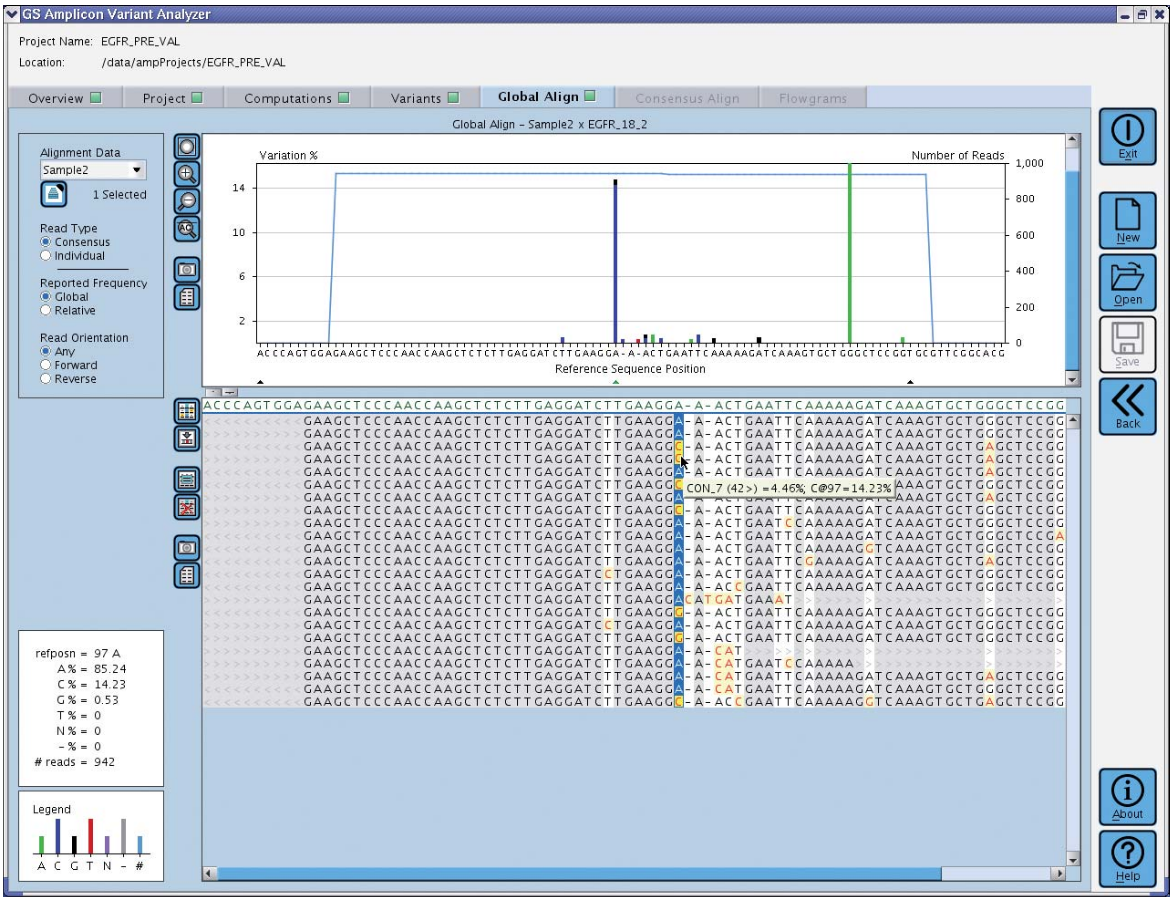
- The **Consensus Align** tab displays the same information as the Global Align tab, but for the set of individual reads that were collapsed into a consensus on the Global Align tab. A line highlighting the differences between that consensus sequence and the Reference Sequence is displayed directly below the Reference Sequence in the alignment.
- The **Flowgrams** tab, finally, displays a tri-flowgram view of an individual read, selected from either the Global Align or the Consensus Align tab. This display is designed to help evaluate the significance of differences between an individual read and a Reference Sequence. As such, the read flowgram displayed is not the raw flowgram of the read, but is a computationally processed version designed to facilitate the comparison of the read flowgram with an idealized flowgram for the Reference Sequence. In particular, flow cycle-shifts may be introduced into one or both flowgrams in order to optimize their alignment, and the flowgram of the read may be computationally reverse-complemented in order that the display always be in the 5'$\rightarrow$3' orientation of the Reference Sequence. Finally, the flowgram only displays the subset of flows relevant to the read's sequence alignment as displayed in the Global Align or Consensus Align tabs. The display is divided into three panels:
  - The top panel shows an aligned, idealized flowgram for the Reference Sequence.
  - The middle panel shows the aligned, (possibly reverse-complemented) flowgram of the read.
  - The bottom panel shows a difference flowgram (read minus reference), where any variation from the Reference Sequence appears as a non-zero value. Specifically, extra signals in the read, relative to the Reference Sequence, are displayed as positive differences in this panel; and missing signals in the read, relative to the Reference Sequence, are displayed as negative differences.

When a tab is divided into panels, the panels can be resized by dragging the separator between them. Also, when the panels are stacked vertically (as in the case of the Global Align, the Consensus Align and the Flowgrams tabs), two small buttons are present at the left edge of the separator(s); these buttons allow you to collapse one of the panels, leaving the entire height of the window to the other one(s). This action is reversible (use the other button to re-expand the panel).

9.1.3.3 Buttons and Plots

The plots, multi-alignment views and data tables displayed in the various tabs of the AVA application are scrollable and/or zoomable graphical elements. They share certain common buttons and functions, *e.g.* to perform the scrolling and zooming. When they do appear, these graphic elements have some or all of the following features (see in Figure 9–3, an example for a Global Align window which has many of these elements):

- Scroll bars for horizontal and/or vertical scrolling (appearing below and to the right of the element, if necessary).
- A column of buttons along the upper left edge of the graphic elements, used for navigation (including various zooming functions) and/or to save snapshot images or text files of the displayed data.
- Additional functional buttons, also in the column at the left of graphic elements, to carry out actions such as applying a selection filter on the reads currently displayed, defining novel Variants, assembling consistent reads (which might span overlapping Amplicons) into consensi, running a computation of the Project, adding or removing Project elements or associations, *etc.*
- "Mousing" functions (pointing, clicking or dragging the mouse, touchpad, pen, *etc.* over the graphical element) to view data values and adjust the zoom level.
- When a plot with DNA sequence information is present (Variation Frequency Plot and Flowgrams), a legend is shown, giving the color code for the nucleotides [as well as gaps ("-") and depth of coverage ("#"), if appropriate].



GS Amplicon Variant Analyzer

Project Name: EGFR_PRE_VAL

Location: /data/ampProjects/EGFR_PRE_VAL

Global Align – Sample2 x EGFR_18_2

Alignment Data: Sample2 — 1 Selected

Read Type: [✓ Consensus] Individual

Reported Frequency: [✓ Global] Relative

Read Orientation: [✓ Any] Forward Reverse

CON_7 (42>) =4.46%; C@97=14.23%

```
refposn = 97 A
    A % = 85.24
    C % = 14.23
    G % = 0.53
    T % = 0
    N % = 0
    - % = 0
# reads = 942
```

Figure 9–3: An example Global Align tab showing many of the common graphical element functions

9.1.3.3.1 Scroll Bars

The scroll bars have the standard functions, and appear when the data in either direction is too large to be shown in the viewable area. In Figure 9–3, for example, the horizontal scrollbar appears in the multi-alignment pane, and the vertical one appears in the plot pane; others are not displayed because axes scales are set such that the full breadth of the data can be seen.

9.1.3.3.2 Navigation Buttons

Many of the buttons appearing to the left of an element are used for navigation on the element, and have the following general functions:

Fit – Fit means to scale out to the limits of the data. The y-axis is rounded to an attractive-looking number rather than stopping at the exact data limit.

Zoom in – Zoom in by a factor of 1.5. This button zooms only the primary (left) y-axis scale; use the **Zoom to labels** and **Freehand zooming** functions described below to zoom the x-axis.

Zoom out – Zoom out by a factor of 1.5. This button will zoom only the primary y-axis scale and, unlike other zoom operations, this will zoom out past the data limits (to allow you to get a better perspective of the data, especially when attempting to visually separate data on the primary and secondary y-axes).

Zoom to labels – This button zooms the x-axis of the flowgram so that the nucleotide/flow characters can fit below the axis.

Snapshot – Save a snapshot image of the current view to disk. This will open a dialog asking for the location and filename to save a PNG format snapshot image. The saved image contains only the currently visible region of the element: in particular, if the element has a scrollbar, only the current, scrolled view is saved.

Text file – Save a text-formatted version of the element. This will open a dialog asking for the location and filename to save the text file. It then saves the data, along with summary information describing the data source. In most cases, this is a Microsoft Excel-compatible, tab-delimited text file of the underlying data for the element. For plots, the text file includes data that may be outside of the current view (due to scrollbars). For a flowgram view (see section 9.8), the data for all three plots are saved to one file, with some white space between the three sections of plot data. For summary data tables, the file contains the tabular data, also including data that may be beyond the current scroll region. For multiple-alignment displays, the text file is an ACE file format version of the alignment, rather than an Excel-compatible spreadsheet. For the Variants Table, finally, the text file includes an extra column (Variant Status), compared to the Table visible in the GUI, placed between the Variant Name and Max Value columns (*i.e.* at column 3 in the text output); note that although the GUI lacks an explicit column for this, the Variant Status data is accessible via the tooltips as you pause the mouse over Variants).

9.1.3.3.3 Mousing Functions

When the mouse cursor is located over a graphical element, you can perform additional functions by moving, clicking or dragging the mouse:

Mouse Tracker – Whenever the cursor is located over a position in a plot or a multi-alignment display, detailed data values for that position are shown in a related Mouse Tracker area, at the bottom left of the window. This allows you to see the specific numerical value for any data point as well as other detailed data associated with the display position.

Freehand Zoom In – The mouse can be used to zoom in on specific regions of a plot. To zoom in, hold the left mouse button down and drag a box around the area of interest (see Figure 9–4). Releasing the button zooms to the area circumscribed by the box. If the plot has both primary and secondary Y-axes, only the primary axis data is zoomed.

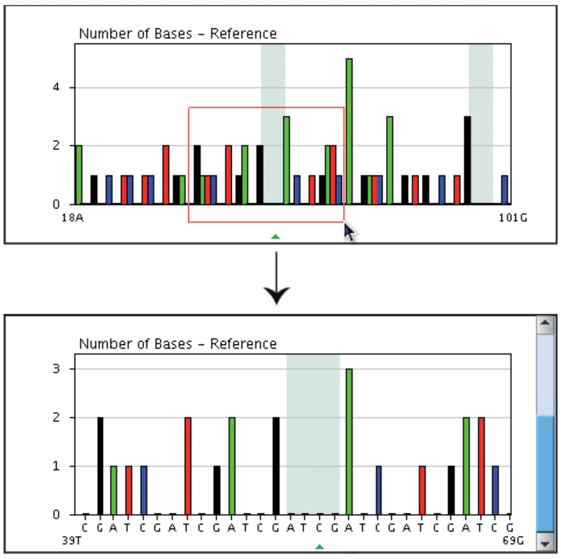

Number of Bases – Reference
18A
101G
Number of Bases – Reference
C G A T C G A T C G A T C G A T C G A T C G A T C G A T C G
39T
69G

Figure 9–4: Freehand zoom in to a flowgram region

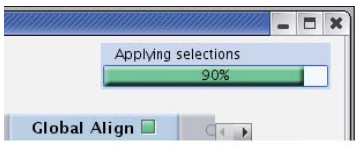
Freehand zoom out – For plots only, right-clicking the plot will cause the plot to zoom out by a factor of 1.5 in both the X and (primary) Y directions, centered on the middle of the current view. This zoom will not zoom farther than the limits of the data.

Screen tips – When you pause the mouse over a button or other display control, over an element definition data, or over most of the Variants or multi-alignment results, a screen tip appears providing some information about the object under the pointer. Some of the most useful examples of this are specified in the tab sections, below.

9.1.3.3.4 Progress bars

When an operation takes more than a few seconds, a Progress bar appears temporarily in the upper-right corner of the application window, to display the progress of the operation (Figure 9–5**A**). Double-clicking on this progress bar opens the Progress window, which contains individual progress bars for the operation or any of its sub-processes (Figure 9–5**B**). In certain contexts, you can cancel an operation by clicking the "Cancel" button that appears to the right of a progress bar in the Progress Window (when cancelling is not possible, such as when new data is being loaded, the Cancel button is present, but grayed out). The Progress window stays open, even after the entire operation completes, until you close it manually.

A



B

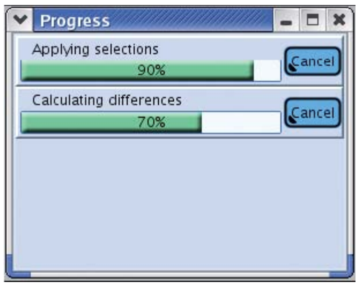


Figure 9–5: (A) A Progress bar (B) The Progress window

9.1.3.3.5 Special Action Buttons

Another category of buttons appears to the left of some graphic elements or in the display options area of certain tabs. Since these are tab-specific, these buttons are described in the corresponding tab sections, below.

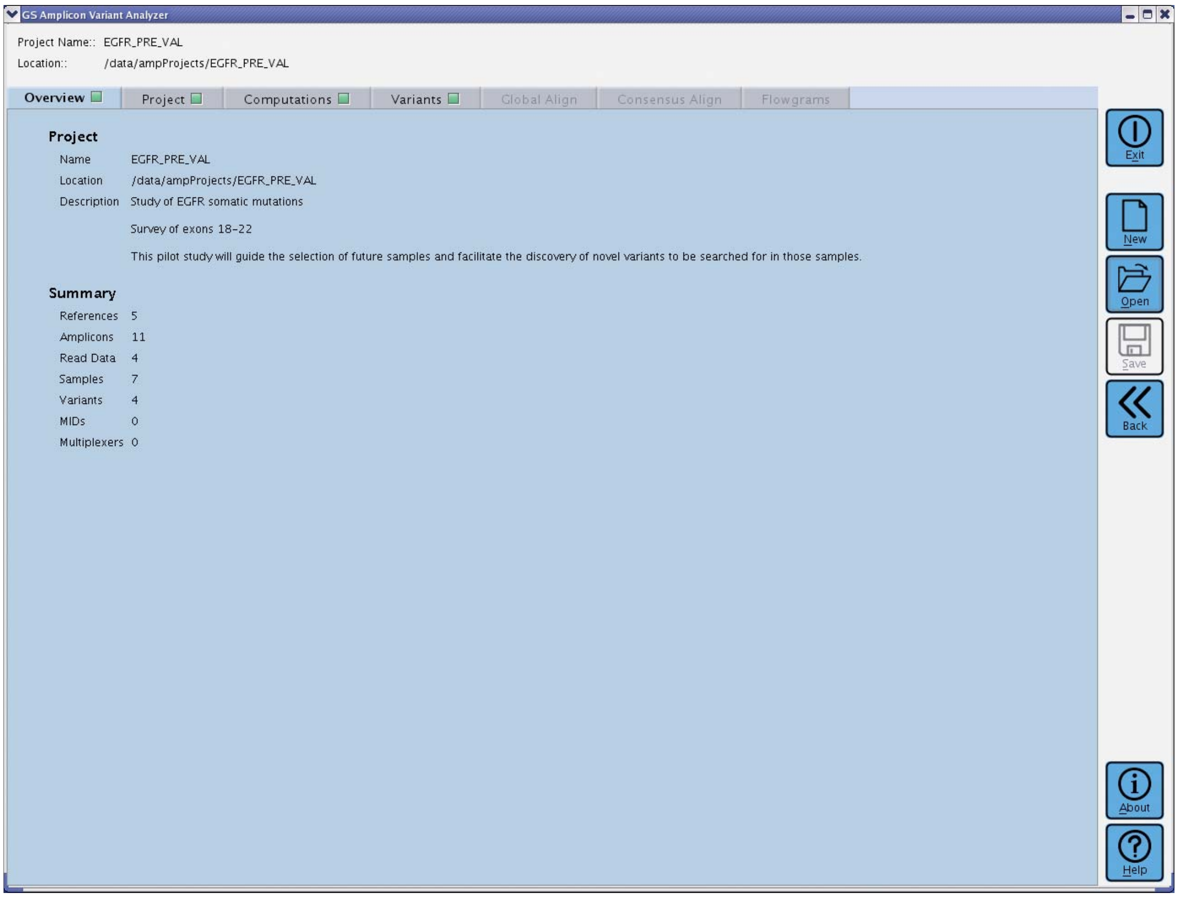
9.1.3.4 File Browsing in Linux

The Linux File Brower used for opening Projects, finding Read Data Sets, and creating New Projects lacks some of the usually expected controls. For example, there are no buttons for going up a directory, creating a new folder, *etc*. However, you can find these functions by right-clicking in the browser window and choosing from the options presented in a contextual menu (Go up, Go Home, View Details/List, Refresh, and New Folder).

9.2 The Overview Tab

This simple tab provides a basic summary of the Amplicon Project (Figure 9–6): the name, location and description of the Project entered in the New Application Project window when the Project was created (or as edited thereafter, *e.g.* from the Project tab); and the number of Reference Sequences, Amplicons, Read Data Sets, Samples, Variants, MIDs and Multiplexers defined in the Project. These numbers also appear on the seven Definition Table sub-tabs of the Project tab.

- When the application is launched, but before a Project is open, the Overview tab displays a brief, general description of the GS Amplicon Variant Analyzer application's usage and capabilities (see Figure 9–1).
- Most of the screenshots in this section are derived from the Project shown in Figure 9–6. Since this Project does not use MIDs, the 454Standard MID set that is automatically loaded when a new Project is created (see section 12.4), has been manually removed to simplify the display of the Project.



Project Name:: EGFR_PRE_VAL
Location:: /data/ampProjects/EGFR_PRE_VAL

Project

Name	EGFR_PRE_VAL
Location	/data/ampProjects/EGFR_PRE_VAL
Description	Study of EGFR somatic mutations
	Survey of exons 18-22
	This pilot study will guide the selection of future samples and facilitate the discovery of novel variants to be searched for in those samples.

Summary

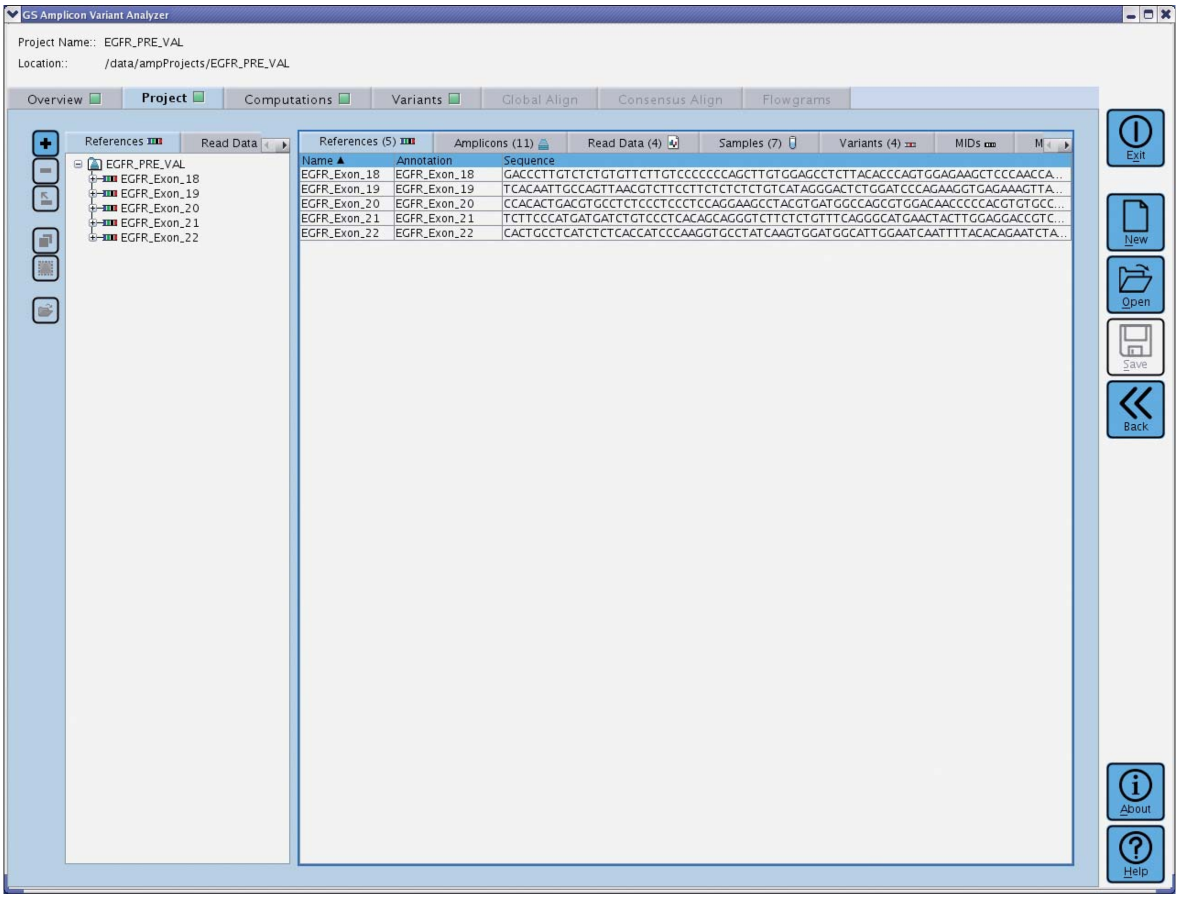
References	5
Amplicons	11
Read Data	4
Samples	7
Variants	4
MIDs	0
Multiplexers	0

Figure 9–6: The Overview tab

9.3 The Project Tab

This is one of the most complex tabs of the AVA application. It is used to set up and navigate an Amplicon Project. Setting up an Amplicon Project means to define all the elements that constitute it, and all their associations. There are seven types of Project elements: Reference Sequences, Amplicons, Read Data Sets / Read Groups, Samples, Variants, and optionally, MIDs / MID Groups, and Multiplexers.

The Project tab is divided into two panels (Figure 9–7): the left-hand panel comprises four sub-tabs, each with one "Tree" representation of the Project that show the diverse interrelations between the Project's elements; and the right-hand panel comprises seven sub-tabs, each with the "Definition Table" for one type of element. Clicking on an element in any tree view that is represented in a Definition Table (Reference, Amplicon, Read Data, Sample, Variant, MID, or Multiplexer) causes the right-hand panel to load the appropriate sub-tab Definition Table with the element selected. Expanding or collapsing a node in the tree or clicking on an item in the tree that is not represented in a Definition Table, such as a Read Group or an MID Group, does not impact the right-hand panel. You can click and drag the divider between the two panels to adjust the space assigned to each panel. If the size of your screen or the position of the panel divider does not allow you to view all the sub-tabs of a panel, a pair of arrow buttons appears in the upper-right corner of that panel, allowing you to scroll the set of sub-tabs to bring hidden ones into view.



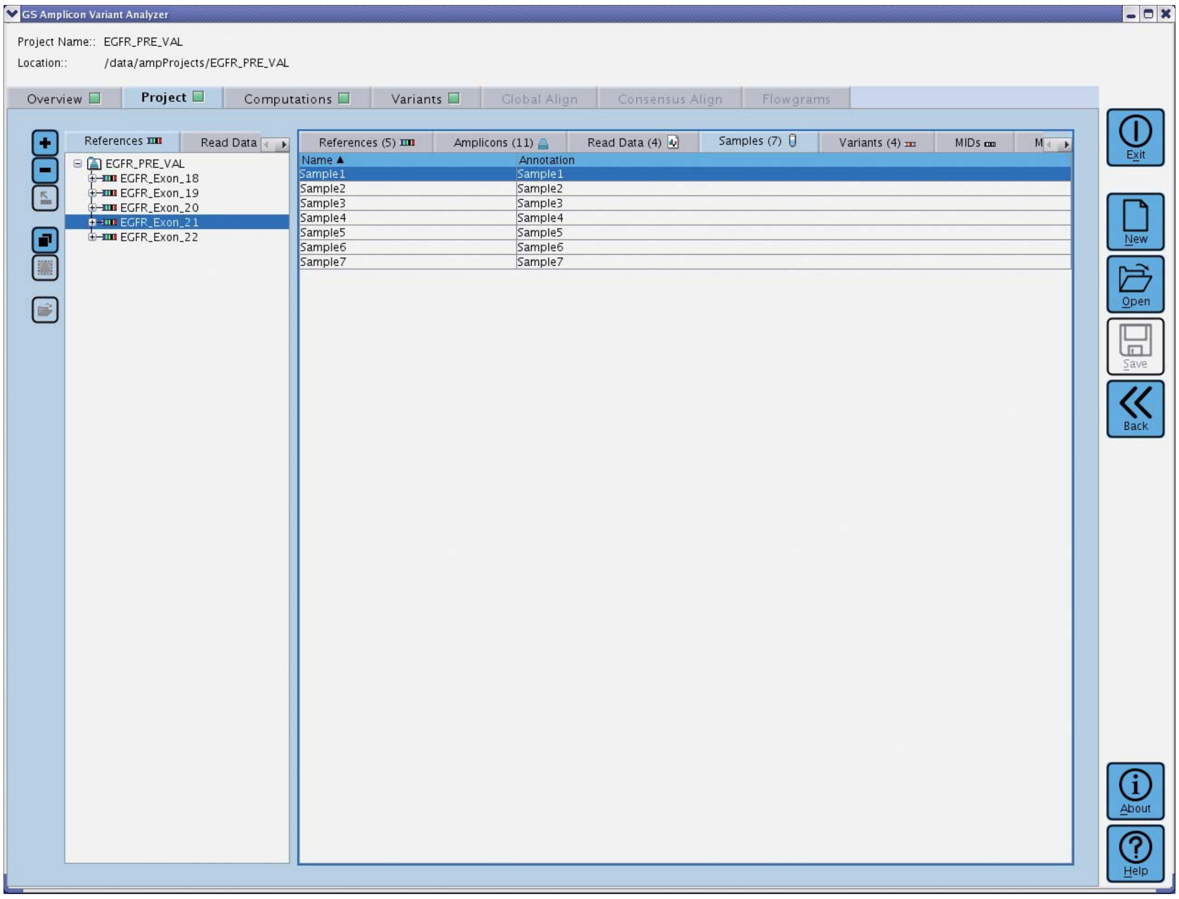
Project Name:: EGFR_PRE_VAL
Location:: /data/ampProjects/EGFR_PRE_VAL

EGFR_PRE_VAL
EGFR_Exon_18
EGFR_Exon_19
EGFR_Exon_20
EGFR_Exon_21
EGFR_Exon_22

Name ▲	Annotation	Sequence
EGFR_Exon_18	EGFR_Exon_18	GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACACCCAGTGGAGAAGCTCCCAACCA...
EGFR_Exon_19	EGFR_Exon_19	TCACAATTGCCAGTTAACGTCTTCCTTCTCTCTCTGTCATAGGGACTCTGGATCCCAGAAGGTGAGAAAGTTA...
EGFR_Exon_20	EGFR_Exon_20	CCACACTGACGTGCCTCTCCCTCCCTCCAGGAAGCCTACGTGATGGCCAGCGTGGACAACCCCCACGTGTGCC...
EGFR_Exon_21	EGFR_Exon_21	TCTTCCCATGATGATCTGTCCCTCACAGCAGGGTCTTCTCTGTTTCAGGGCATGAACTACTTGGAGGACCGTC...
EGFR_Exon_22	EGFR_Exon_22	CACTGCCTCATCTCTCACCATCCCAAGGTGCCTATCAAGTGGATGGCATTGGAATCAATTTTACACAGAATCTA...

Figure 9–7: The Project tab

A column of six buttons runs down the left edge of the Project tab (inactive and grayed-out in Figure 9–7), whose individual functions are discussed in detail in sections 9.3.1 and 9.3.2. These buttons are context sensitive in that the target of their actions can be either an object in the Tree or in the Definition Table panel, whichever was clicked last: the application provides a visual reminder of which panel is "active" (and will be subjected to the action of the left margin buttons) by surrounding the active panel with a thin blue, rectangular border. For example, Figure 9–8 shows a Reference Sequence selected in the left panel and a Sample selected in the right panel, and some of the left margin buttons are active; the blue border is around the right panel, so the Sample from the right panel is the current target of the available buttons.


GS Amplicon Variant Analyzer

Project Name:: EGFR_PRE_VAL

Location:: /data/ampProjects/EGFR_PRE_VAL

Overview | Project | Computations | Variants | Global Align | Consensus Align | Flowgrams

References | Read Data

- EGFR_PRE_VAL
  - EGFR_Exon_18
  - EGFR_Exon_19
  - EGFR_Exon_20
  - EGFR_Exon_21
  - EGFR_Exon_22

References (5) | Amplicons (11) | Read Data (4) | Samples (7) | Variants (4) | MIDs | M

Name ▲	Annotation
Sample1	Sample1
Sample2	Sample2
Sample3	Sample3
Sample4	Sample4
Sample5	Sample5
Sample6	Sample6
Sample7	Sample7

Figure 9–8: The Project tab with the right-hand Definition Table panel highlighted with a rectangular blue border indicating that the panel is "active". The left margin buttons that are active (*i.e.* not grayed-out) will operate on the "Sample1" that is selected on the right, and not the "EGFR_Exon_21" that is selected in the tree on the left.

Re-computing a changed Project: Project results in the Variants, Global Align, Consensus Align, or the Flowgrams tabs are representative of the state of the Project as defined in the Project tab, at the time of the last computation (see section 9.4). If a change is made to a Project element that is germane to these results, the results will remain but will be out of date until you re-compute. This includes changes in the definition of Reference Sequences, Amplicons, Variants and Samples, and the addition or removal (or inactivation) of Read Data Sets; as well as changes in their associations, including changes in the definition of MIDs or Multiplexers. If you find that the data in these tabs does not reflect the current state of the Project, try re-computing it.

9.3.1 The “Project Tree” Sub-Tabs

In a new Project, each “Project Tree” tab contains a single folder representing the Project itself. Right-clicking on this folder opens a contextual menu that includes a “Properties” option; this opens a “Project Properties” window in which you can enter or edit the name and description of the Project. You cannot modify the location of the Project from within the Project. (Note that changing the name of the Project this way will NOT also change the name of the folder that contains it, in your file-system, so be aware of the possibility of mismatch between the Project name and its file-system location.)

The “tree” Project views provide a convenient way to add, remove and organize (“associate”) the various elements that compose a Project, and to navigate it afterwards. There are 5 ways to construct or otherwise manipulate a tree.

- You can use the buttons located to the left of the window’s left panel
- You can right-click on an existing element in a Project Tree, which opens a contextual menu that includes the same actions of the buttons
- You can drag elements from the Definition Tables on the right panel of the tab, to the appropriate element in a tree view (see section 9.3.2)
- As a special case, the associations between Amplicons or Variants and their Reference Sequences can also be specified in the Definition Tables of these elements, and will then appear in the References Tree.
- As another special case, when MIDs and Multiplexers are used, editing the associations between MIDs and Samples for a given Multiplexer (see section 9.3.2.7.3) may cause any Amplicons previously associated with Samples using that Multiplexer to shift to new positions in the tree, to reflect associations to the Multiplexer’s new Samples.

The functions of the five action buttons to the left of the Tree panel that are applicable to trees are described below. Right-clicking on an element in a tree opens a contextual menu that contains the same actions [plus, if you right-click on a Sample-Amplicon pair, the “Global Align” action (described in section 9.6.1)].

➕ The “Add” button allows you to create a new element in the tree (except for Read Data Sets; see the “Import Data” button, below), as a branch under the element selected at the time you clicked the button, automatically creating an association between the two. This action is contextual, *i.e.* the type of element you can create with this button depends on which “tree” you are in and which type of element is selected at the time you click the button. When the context allows for the creation of more than one type of element, a contextual menu opens to let you choose; if there is only one possibility, the new element and association are created directly.

➖ The “Remove from Project” button deletes the element selected from the Project altogether. Of course, this severs all the associations it may have had with other elements. If the association is based on a definitional relationship then those other elements are deleted as well (specifically, deleting a Reference Sequence will delete any Amplicons or Variants that are defined based on their association with the Reference Sequence). If there are related elements in purely associational relationships, then it does NOT delete those other elements, even if they were in a lower branch of the tree in the particular tree view from which the operation is carried out. For example, if you remove a Sample from the Sample Tree, all Amplicons associated with that Sample remain in the Project (even if they are no longer associated with any Sample at all) and keep their full definition and all other associations they may have. In all cases, a warning message is displayed to indicate exactly what elements or associations will be removed from the project as a result of the removal action.

The "Remove association and remain in project" button severs the association between the element selected at the time you click the button and the element above it in the tree, but leaves all elements otherwise fully defined in the Project. This button is contextual as well, as not all links can be severed. For example, you cannot sever the link between a Sample and an Amplicon in the References Tree (though you can in the Read Data and Samples Trees).

The "Select Amplicons associated with item" button provides the ability to select, within the Definition Table, all the Amplicons associated with an item in the tree, based on the relationships of that item that exist in the tree.

For example, selecting a Reference, a Sample, or a Multiplexer in one of the trees, and clicking this button causes the Definitions Table panel to switch to the Amplicons Definition Table sub-tab. Within that Amplicon table, this multi-selects the subset of Amplicons that are associated with the Reference, Sample, or Multiplexer used to trigger the operation. You can drag the items you multi-selected in the Table to another valid tree location by holding down the control key before left-clicking on one of the selected items with the mouse. (Note: The Amplicon on which you clicked will initially become deselected, but it will be added back to the selection at the moment the selection gets dragged.)

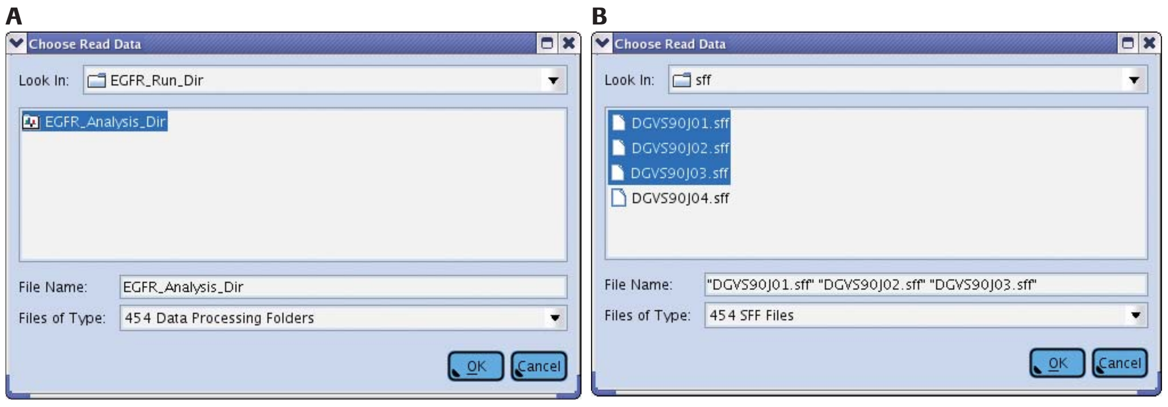
The primary utility of this button is that it allows complex Sample-Amplicon or Multiplexer-Amplicon relationships to be cloned to another Sample or Multiplexer with a single drag-and-drop operation, rather than making many individual manual selections, drags, and drops to re-create the whole set. See also section 9.3.2 for information on how dragged Amplicons can "trickle" down a Tree.

The "Import data" button allows you to add Read Data Set(s) to the Project. Clicking this button opens a "Choose Read Data" window from which you can browse your file-system to select the Read Data Set(s) of interest. These can be:

- The Data Processing ('D_') folder of a sequencing Run: select '454 Data Processing Folders' in the 'Files of Type' drop down menu, at the bottom of the window; (Figure 9–9**A**; Data Processing directories are marked by a special icon:).
- Individual SFF file(s): select '454 SFF Files' in the 'Files of Type' drop down menu (Figure 9–9**B**).

Note that the data set(s) must comprise reads from an Amplicon library(ies) to be useable in the AVA software.

A **B**



A: Choose Read Data

Look In: EGFR_Run_Dir

EGFR_Analysis_Dir

File Name: EGFR_Analysis_Dir

Files of Type: 454 Data Processing Folders

B: Choose Read Data

Look In: sff

DGVS90J01.sff
DGVS90J02.sff
DGVS90J03.sff
DGVS90J04.sff

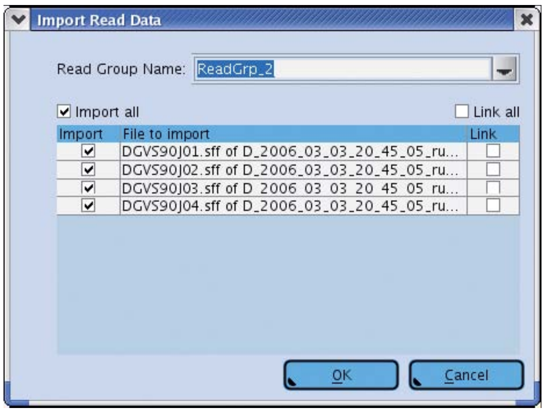
File Name: "DGVS90J01.sff" "DGVS90J02.sff" "DGVS90J03.sff"

Files of Type: 454 SFF Files

Figure 9–9: The Choose Read Data window showing (A) an example with a Data Processing Directory selected, and (B) an example with some SFF files selected.

When the selection is made and you click the "OK" button to accept it, an "Import Read Data" window opens (Figure 9–10). In this window, you can:

- select the exact file(s) to import (*e.g.* from the list of SFF files corresponding to the PicoTiterPlate device regions of the sequencing Run, if an Data Processing ('D_') folder was selected in the previous step) by clicking the "Import all" or the appropriate check box(es) to the left of the Read Data Set name(s);
- determine whether to import full copies of the Read Data Set(s) into the Project, or only symbolic links to the Set(s) selected, by clicking the "Link all" or the appropriate check box(es) to the right of the Read Data Set name(s) (see first Note below);
- incorporate the Read Data Set(s) you are importing into an existing Read Group or into a new one (which you can name in the appropriate field, in this window) (see second Note below).



Read Group Name: ReadGrp_2

✓ Import all   ☐ Link all

Import	File to import	Link
✓	DGVS90J01.sff of D_2006_03_03_20_45_05_ru...	☐
✓	DGVS90J02.sff of D_2006_03_03_20_45_05_ru...	☐
✓	DGVS90J03.sff of D 2006 03 03 20 45 05 ru...	☐
✓	DGVS90J04.sff of D_2006_03_03_20_45_05_ru...	☐

Figure 9–10: The Import Read Data window

- All the files related to an Amplicon Project are collected in a single folder, at a location determined by the user at the time the Project was created (or as modified thereafter). To save file transfer time and disk space, you can choose to import only a symbolic link to the data file(s) rather than the files themselves. However, be aware that if you do this and then the file that is the target of the link is moved, the AVA software will not be able to compute (or re-compute) the Project.

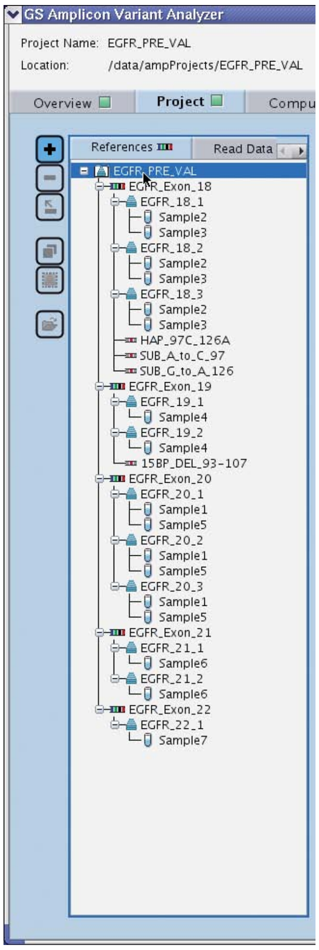
- Read Groups are distinct entities (which can be created at the root node of the Read Data Tree, using the "Add" button), and although you can rename an existing Read Group to match the name of another pre-existing Read Group, this will not cause the Read Data Sets to be merged into the pre-existing group.

The "Add" action above will only create empty Project elements with generic names. You can re-name an element from either a Project Tree or a Definition Table. In a Project Tree, click twice with a slight pause between the clicks to activate an editor directly on the name of the element in the tree. (Note that you can also change the name of the Project this way but that this will NOT also change the name of the folder that contains it, in your file-system, so be aware of the possibility of mismatch between the Project name and its file-system location.) Once elements are created (and possibly named), they can be fully defined in the corresponding "Definition Table" tab (see section 9.3.2). The approach of creating Project elements in a Project Tree view before fully defining them is convenient because it allows the user to set the structure of the Project up-front, possibly even before any sequencing reads are imported; see the documentation on "Samples" (sections 9.1.1.6, 9.3.1.3, 9.3.2.4, and 10.6.1) for an explanation of the usefulness (and complexity) of this.

For large Projects, especially when large amounts of data need to be imported into, exported from, or automated within a Project, the Command Line Interface (CLI) of the AVA software may be more appropriate than the Graphical User's Interface (GUI) described in the present section. See section 11 for a full description of the CLI, the language that was developed for it, and all the commands it includes.

9.3.1.1 The References Tree

The **References** Tree sub-tab shows the Reference Sequences as the main limbs of the Project Tree, with the Amplicons and Variants associated with each Reference Sequence as the next branching level, and the Samples associated with each Amplicon in the third level (Figure 9–11). This tree is useful to help easily visualize all the Amplicons and defined Variants that are associated with each Reference Sequence, and all the Samples that will report on each Amplicon. You can also use this tree to populate the Global Align tab with the multi-aligned reads of any Sample-Amplicon pair you have created in your Project and for which computations have been carried out (see section 9.6.1).



```
Project Name:  EGFR_PRE_VAL
Location:      /data/ampProjects/EGFR_PRE_VAL

EGFR_PRE_VAL
  EGFR_Exon_18
    EGFR_18_1
      Sample2
      Sample3
    EGFR_18_2
      Sample2
      Sample3
    EGFR_18_3
      Sample2
      Sample3
    HAP_97C_126A
    SUB_A_to_C_97
    SUB_G_to_A_126
  EGFR_Exon_19
    EGFR_19_1
      Sample4
    EGFR_19_2
      Sample4
    15BP_DEL_93-107
  EGFR_Exon_20
    EGFR_20_1
      Sample1
      Sample5
    EGFR_20_2
      Sample1
      Sample5
    EGFR_20_3
      Sample1
      Sample5
  EGFR_Exon_21
    EGFR_21_1
      Sample6
    EGFR_21_2
      Sample6
  EGFR_Exon_22
    EGFR_22_1
      Sample7
```

Figure 9–11: The References Tree sub-tab of the Project tab's left-hand panel

9.3.1.2 The Read Data Tree

The **Read Data** Tree sub-tab shows the Read Groups/Read Data Sets as the main limbs of the Project Tree, with the Samples associated to each Read Data Set as the next branching level, and the Amplicons associated to each Sample in the last level (Figure 9–12). If the libraries were prepared with MIDs and Multiplexers are defined in the Project, the Multiplexers are displayed in this Tree, between the Read Data Sets and the Samples. (Read Groups are only a means to associate several Read Data Sets together, *e.g.* the various PicoTiterPlate device regions of a sequencing Run, for better ease of handling.)

You can use this tree to populate the Global Align tab with the multi-aligned reads of any Sample-Amplicon pair you have created in your Project for which computations have been carried out (see section 9.6.1). The principal use of the Read Data Tree, however, is to establish which Read Data sets supply the Amplicon reads to particular Samples. Thus, rather than merely establishing that reads from some Amplicon generally exist in the Project for a given Sample (as on the Sample Tree; see section 9.3.1.3), the Read Data Tree represents specifically which Read Data Set supplies those reads (and which Multiplexer defines the read assignments to the Samples, if applicable).

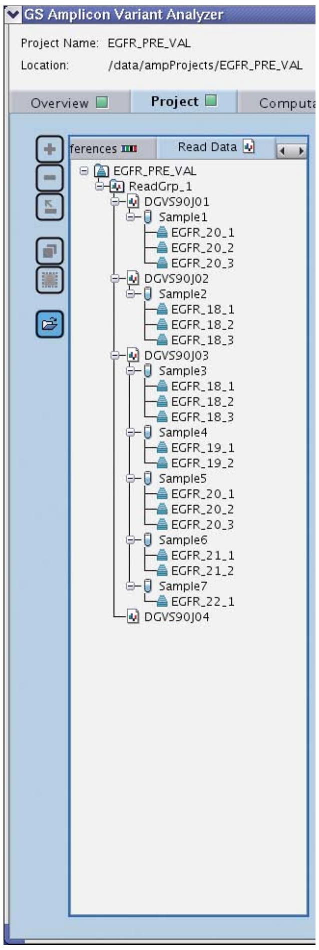
The AVA software will not allow you to associate a given Amplicon with more than one Sample or Multiplexer within the branch of a Read Data Set. This is important because the demultiplexing phase of computation (see section 9.4) depends on the uniqueness of such associations.

False Amplicon associations in the Read Data Tree:
Be careful to limit the Amplicons "lower" branches of this tree to those to which the specific Read Data Set truly contributes. False Sample-Amplicon associations could easily creep into a Read Data Set branch of your Project set up when you use the dragging method (section 9.3.2) to associate Samples with Read Data Sets; while convenient, this method brings the Sample *with all its associated Amplicons* into the Read Data Tree (unless any of these Amplicons are already associated with another Sample in this branch of the tree; see Note above). Similarly, if you drag one or more Amplicons to the root node or to a Read Group node in the Read Data Tree, they will get associated with every eligible Sample under the receiving node (see section 9.3.2). After you create such associations, therefore, make sure to "prune" the tree to remove any Amplicons that don't belong to any given Read Data Set branch or to any given Sample; you can "prune" using the "Remove association and remain in project" button or its equivalent right-click contextual menu option. (Note that deleting an association between a Sample and an Amplicon within the Read Data Tree has no effect on the association between those entities in the Samples Tree; see section 9.3.1.3.)

This is important because the Read Data Tree provides the specific Run information for each of the Sample-Amplicon pairs used by the AVA software to determine which read sequences (Amplicons) to look for in each Read Data Set, and to which Sample each read belongs. The presence of false Amplicon associations in this tree would not only needlessly lengthen the time required for the Trimming step (as the software would search all the reads for primer pairs that are, in fact, not present), but it could even cause a read to be assigned to a non-existent Amplicon, should a spurious match occur.

Note that the Samples Tree, by comparison, represents all the Sample-Amplicon associations relevant to the Project *design*, whether or not any Read Data Set(s) containing such reads have (yet) been imported into the Project (see section 9.3.1.3): all Sample-Amplicon associations seen in any branch of the Read Data Tree are also seen in the Samples Tree; but Sample-Amplicon associations present in the Samples Tree do not (or should not) necessarily be present in any given branch of the Read Data Tree

- Dragging a Sample from its Definition Table onto a Read Data Tree node that already contains this Sample will not create a duplicate "Sample" sub-branch on the tree. However if, at the time of this dragging action, the Sample has Amplicon associations that are not displayed in the tree node, these associations will be added to the branch (unless an association already exists between any of these Amplicons and another Sample in this branch of the tree; see the first Note above). As explained above, you should then "prune" the Read Data Tree of any false Read Data-Amplicon associations that may have been created.
- A Sample must have at least one Amplicon associated with it to be associated with a Read Data. This is because, as seen above, the association of Amplicons to Read Data is intrinsic to this tree, just as the Sample-Read Data is. One can view the information contained in this tree as Read Data-Sample-Amplicon association triads.
- For Amplicon libraries created with MIDs, the method of dragging a Sample with its associated Amplicons to a Read Data Tree node is NOT used. Rather, a Multiplexer is first associated with the Read Data Tree node, and then one or more Amplicons are dragged to the Multiplexer node. All Sample associations (to Read Data Sets and to Amplicons) are made indirectly, as defined by the Multiplexer *i.e.*, using the MIDs.



GS Amplicon Variant Analyzer

Project Name: EGFR_PRE_VAL

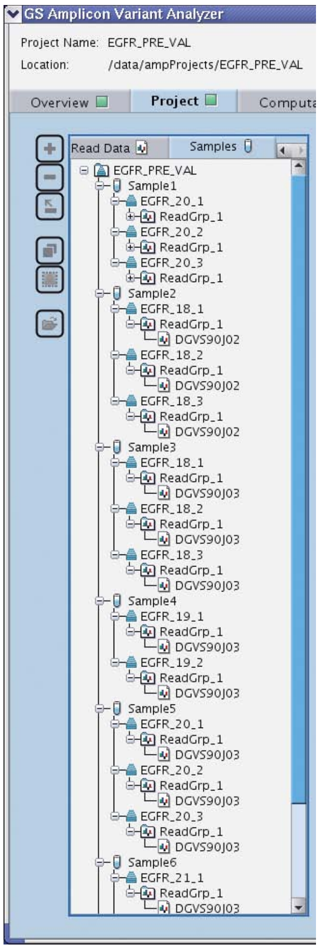
Location: /data/ampProjects/EGFR_PRE_VAL

```
EGFR_PRE_VAL
  ReadGrp_1
    DGVS90J01
      Sample1
        EGFR_20_1
        EGFR_20_2
        EGFR_20_3
    DGVS90J02
      Sample2
        EGFR_18_1
        EGFR_18_2
        EGFR_18_3
    DGVS90J03
      Sample3
        EGFR_18_1
        EGFR_18_2
        EGFR_18_3
      Sample4
        EGFR_19_1
        EGFR_19_2
      Sample5
        EGFR_20_1
        EGFR_20_2
        EGFR_20_3
      Sample6
        EGFR_21_1
        EGFR_21_2
      Sample7
        EGFR_22_1
    DGVS90J04
```

Figure 9–12: The Read Data Tree sub-tab of the Project tab's left-hand panel. In this example we see that Sample1 has reads for Amplicon EGFR_20_3 which can be found in Read Data Set DGVS90J01. Sample5 also has reads for EGFR_20_3, but those are found in a different Read Data Set (DGVS90J03), which is allowed. Read Data Set DGVS90J04 has been imported into the Project, but no Sample-Amplicon pairs have yet been associated with it. Hence, DGVS90J04 would be excluded from Computations if carried out at this point, since the AVA software would not know to which Samples and Amplicons its reads belong (see section 9.4).

9.3.1.3 The Samples Tree

The **Samples** Tree sub-tab shows the Samples as the main limbs of the Project Tree, with the Amplicons associated to each Sample as the next branching level, and the Read Group/Read Data Sets associated to each Amplicon in the third/fourth level (Figure 9–13). Since this tree representation lists together all the Amplicons associated with each Sample, it is useful to navigate the results from a given Sample, irrespective of which Read Data Set supplied the reads for each Amplicon. You can use it to design your project, showing not only Sample-Amplicon pairs for which Read Data Set(s) already exists in your Project (shown in the Read Data Tree), but also any other Sample-Amplicon pairs that you expect to see over the course of the entire Project, even if the Read Data Set has not yet been imported into the Project (or even yet exists). Therefore, this tree does not have the functional constraints of the Read Data Tree, which provides the specific Run information for each of the Sample-Amplicon pairs that will be used for computation by the AVA software (see section 9.3.1.2). You can also use this tree to populate the Global Align tab with the multi-aligned reads of any Sample-Amplicon pair you have created in your Project for which computations have been carried out (see section 9.6.1).



Project Name: EGFR_PRE_VAL
Location: /data/ampProjects/EGFR_PRE_VAL

```
EGFR_PRE_VAL
  Sample1
    EGFR_20_1
      ReadGrp_1
    EGFR_20_2
      ReadGrp_1
    EGFR_20_3
      ReadGrp_1
  Sample2
    EGFR_18_1
      ReadGrp_1
        DGVS90J02
    EGFR_18_2
      ReadGrp_1
        DGVS90J02
    EGFR_18_3
      ReadGrp_1
        DGVS90J02
  Sample3
    EGFR_18_1
      ReadGrp_1
        DGVS90J03
    EGFR_18_2
      ReadGrp_1
        DGVS90J03
    EGFR_18_3
      ReadGrp_1
        DGVS90J03
  Sample4
    EGFR_19_1
      ReadGrp_1
        DGVS90J03
    EGFR_19_2
      ReadGrp_1
        DGVS90J03
  Sample5
    EGFR_20_1
      ReadGrp_1
        DGVS90J03
    EGFR_20_2
      ReadGrp_1
        DGVS90J03
    EGFR_20_3
      ReadGrp_1
        DGVS90J03
  Sample6
    EGFR_21_1
      ReadGrp_1
        DGVS90J03
```

Figure 9–13: The Samples Tree sub-tab of the Project tab's left-hand panel

9.3.1.4 The MIDs Tree

This is the simplest of the Tree sub-tabs, as it simply lists the MIDs that are defined in the Project, with the optional MID Groups if applicable (Figure 9–14).

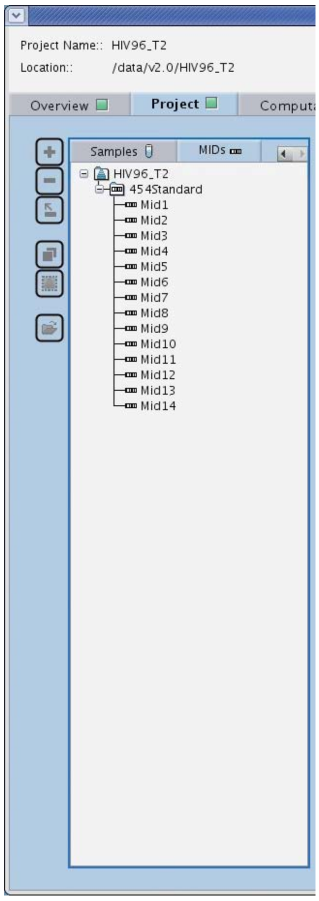


Figure 9–14: The MIDs Tree sub-tab of the Project tab’s left-hand panel

9.3.2 The "Definition Table" Sub-Tabs

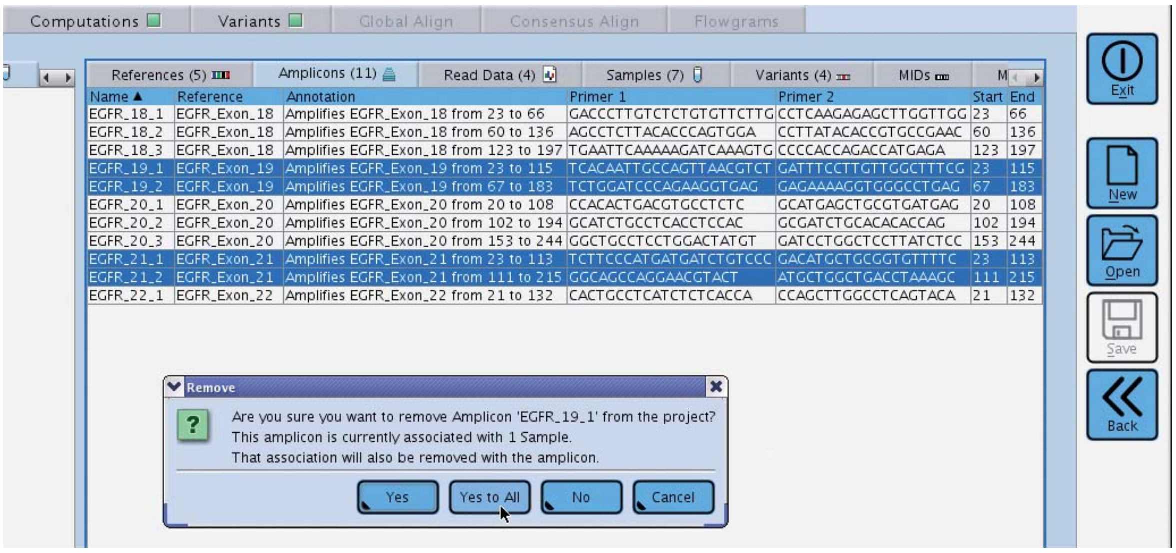
The right-hand panel of the Project tab contains seven sub-tabs, one for each type of element that makes up an Amplicon Project (except "Group" element types). The number of elements listed in a Definition Table appears next to the tab name, in parenthesis. You can select the row for a particular element in a Definition Table by clicking on any of its cells and you can select multiple rows by holding down the 'shift' or 'control' keys while making your selections. You can use the Tables on these sub-tabs to create or delete elements of the corresponding type (or "import" Read Data Sets) for your Project, using most of the same buttons or right-click functions as for the "Tree" sub-tabs of the left panel (see section 9.3.1).

To add a new element to the Project (except for Read Data Sets and Groups), either click in the appropriate sub-tab to make it the focus of the application, then click the "Add" button to the left of the Project tab; or right-click on an existing element in the appropriate Definition Table and select the "Add" option from the contextual menu that appears. The "Add" action is used to create new elements that can be completely defined using tools of the Project tab. However, certain data, specifically the Read Data Sets, are defined outside the application and must be imported into the Project. For this data, use the "Import" action.

To import Read Data Sets into the Project, either click in the Read Data sub-tab to make it the focus of the application, then click the "Import data" button to the left of the Project tab; or right-click on an existing element in the Read Data Definition Table and select the "Import" option from the contextual menu that appears.

To remove an existing element from the Project (including a Read Data Set), either select it in its Definition Table and click the "Remove from project" button to the left of the Project tab; or right-click on the element to be deleted in its Definition Table and select the "Remove" option from the contextual menu that appears. A confirmation window will appear; click "Yes" to delete the element. If you make multiple selections and choose the "Remove" action, the confirmation window will prompt you to remove each one individually, or you can click the "Yes to all" button to confirm the removal of all selected elements at once (Figure 9–15).

The "Duplicate item" button is used to create copies of items in the Definition Tables. This is another contextual button that operates on an item that is selected in one of the Definition Table sub-tabs. If a single item is selected in the Definition Table, clicking on this button will add an extra row to the table that is identical to the selected item except that its name will have a suffix of the form "_copy_NUM" (where "NUM" increases by 1 when more than one copy is made of an original item). A copy of a copy adds another "copy" suffix (*e.g.*, "ItemName_copy_2_copy_1" would result from a duplication of "ItemName_copy_2"). The duplication operation only duplicates data that is explicitly associated with the item in the Table row: it does not duplicate any associations the item might have as implied by the tree structures (such as Sample-Amplicon associations) unless they are specified in the Table (such as the Reference association in the Amplicon and Variant Definition Tables, and the content of Multiplexers). The duplication of Read Data is not currently supported.



Name ▲	Reference	Annotation	Primer 1	Primer 2	Start	End
EGFR_18_1	EGFR_Exon_18	Amplifies EGFR_Exon_18 from 23 to 66	GACCCTTGTCTCTGTGTTCTTG	CCTCAAGAGAGCTTGGTTGG	23	66
EGFR_18_2	EGFR_Exon_18	Amplifies EGFR_Exon_18 from 60 to 136	AGCCTCTTACACCCAGTGGA	CCTTATACACCGTGCCGAAC	60	136
EGFR_18_3	EGFR_Exon_18	Amplifies EGFR_Exon_18 from 123 to 197	TGAATTCAAAAAGATCAAAGTG	CCCCACCAGACCATGAGA	123	197
EGFR_19_1	EGFR_Exon_19	Amplifies EGFR_Exon_19 from 23 to 115	TCACAATTGCCAGTTAACGTCT	GATTTCCTTGTTGGCTTTCG	23	115
EGFR_19_2	EGFR_Exon_19	Amplifies EGFR_Exon_19 from 67 to 183	TCTGGATCCCAGAAGGTGAG	GAGAAAAGGTGGGCCTGAG	67	183
EGFR_20_1	EGFR_Exon_20	Amplifies EGFR_Exon_20 from 20 to 108	CCACACTGACGTGCCTCTC	GCATGAGCTGCGTGATGAG	20	108
EGFR_20_2	EGFR_Exon_20	Amplifies EGFR_Exon_20 from 102 to 194	GCATCTGCCTCACCTCCAC	GCGATCTGCACACACCAG	102	194
EGFR_20_3	EGFR_Exon_20	Amplifies EGFR_Exon_20 from 153 to 244	GGCTGCCTCCTGGACTATGT	GATCCTGGCTCCTTATCTCC	153	244
EGFR_21_1	EGFR_Exon_21	Amplifies EGFR_Exon_21 from 23 to 113	TCTTCCCATGATGATCTGTCCC	GACATGCTGCGGTGTTTTC	23	113
EGFR_21_2	EGFR_Exon_21	Amplifies EGFR_Exon_21 from 111 to 215	GGCAGCCAGGAACGTACT	ATGCTGGCTGACCTAAAGC	111	215
EGFR_22_1	EGFR_Exon_22	Amplifies EGFR_Exon_22 from 21 to 132	CACTGCCTCATCTCTCACCA	CCAGCTTGGCCTCAGTACA	21	132

Remove

Are you sure you want to remove Amplicon 'EGFR_19_1' from the project?
This amplicon is currently associated with 1 Sample.
That association will also be removed with the amplicon.

Yes | Yes to All | No | Cancel

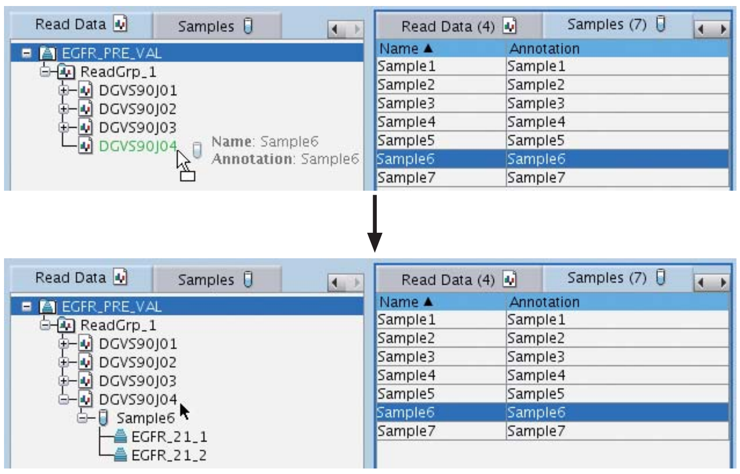
Figure 9–15: The Amplicons Definition Table with a multiple rows selected. The user is prompted to remove the selected rows one at a time, which can be done by clicking "Yes" to each; or you can remove all the selected rows at once using the "Yes to All" button.

Contrary to the case with the tree tabs, the only associations you can create or modify within the Definition Tables are the ones between:

- Amplicons or Variants and their Reference Sequence
- Read Data Sets and their Read Data Groups, or MIDs and MID Groups, which are done via a drop down menu that appears when you double-click in the corresponding cell, in these Tables
- The triads Amplicons/Read Data Sets/Samples (indirectly), when MIDs and Multiplexers are used. Specifically, modifying the MID and Sample associations for a given Multiplexer, using the functionality found in the Multiplexer Definition Table, will dynamically update the associated Samples for any Amplicons of a Read Data Set that are associated with the changed Multiplexer.

The "Remove association and remain in project" button or its right-click equivalent are never available when the focus of the application (the last place you clicked) is on a sub-tab of the right-hand panel because not all associations are explicit in the Definition Tables (especially the multiple associations an element may have); indeed, the ability to view the network of element associations in the Project is one of the main benefits of the Project Tree views.

However, associations can also be established by dragging an element from its Definition Table on a right-hand sub-tab, to an appropriate element on a tree view in a left panel sub-tab: hold the mouse button until the name of the "destination" element on the tree view turns green, and then release the mouse button (Figure 9–16).



Name ▲	Annotation
Sample1	Sample1
Sample2	Sample2
Sample3	Sample3
Sample4	Sample4
Sample5	Sample5
Sample6	Sample6
Sample7	Sample7

Figure 9–16: Creating an association by dragging an element from its Definition Table on a right-hand sub-tab to an appropriate element on a tree view in a left panel sub-tab. In this case, Sample6 (with its two previously-associated Amplicons EGFR_21_1 and EGFR_21_2) are being associated to Read Data Set DGVS90J04. (See the "Caution" in section 9.3.1.2 for special information about dragging Samples into the Read Data Tree.)

- The dragging action must always proceed from a Definition Table on the right to an appropriate tree node on the left (or from a tree node to another tree node); you cannot establish an association by dragging a tree node object to an object in its Definition Table.
- The software will not allow you to establish invalid associations, such as linking an Amplicon to a Variant: only elements that are valid destinations for the element you are dragging will turn green and allow the creation of the new association.
- If you drag one (or more) valid element(s) to the root node (the "Project" folder) in the Samples Tree, or to the root node, a Read Group node or a Read Data node in the Read Data Tree, the element will become associated with all the relevant elements in the nodes below and will be added to all the corresponding branches of that tree (unless this would cause an Amplicon to become associated with more than one Sample or Multiplexer within a Read Data Set branch of the Read Data Tree; in such a case, the existing association remains and only new, non-conflicting associations are created).
  - This is particularly useful when the experimental design requires the association of one or more Amplicons to a large number of Samples, in one or more Read Data Sets. Such a design may be especially common for experiments using MIDs, where the same Amplicon(s) are to be monitored in multiple MID-tagged libraries (Samples). In this case, Amplicons dragged to an upper node will "trickle down" and become associated with all the relevant and eligible objects below that node. However, the restriction that an Amplicon be associated with only one Sample (or Multiplexer) within a given Read Data Set imposes the following behavior for this function:
    - The "trickle" will only be accepted if there is at least one Read Data at or below the level of the drag that has at most one Sample or Multiplexer currently associated with it [so it is unambiguous which Sample or Multiplexer the dragged Amplicon(s) should be associated with] and if the Amplicons are not already associated with those Samples or Multiplexers.
    - If multiple Amplicons are dragged together and some of them are already associated with some of the Samples or Multiplexers under the recipient node, but others aren't, then the non-associated Amplicons will become associated, and the others will be ignored.

In general, to add or edit the information for an element in its Definition Table, double-click in the corresponding cell in the Table. In some cases, the content of the cell will be highlighted and you can type in the new content; in other cases, the double-click opens a separate window for data entry. Two characteristics, Name and Annotation, are common to all element types; the method to enter or edit them is provided below. The editing methods applicable to the other characteristics of each element type are specified in the sub-sections below.

To edit a Project element's Name in its Definition Table

1. Double-click in the "Name" cell for the element you want to re-name.
2. Overtype the new name.
3. Press "Enter" or click elsewhere.

To enter or edit a Project element's Annotation its Definition Table

1. Double-click in the "Annotation" cell for the element you want to re-name. An "Edit Annotation" window will open (Figure 9–17).
2. Type or overtype the information desired.
3. Click "OK".

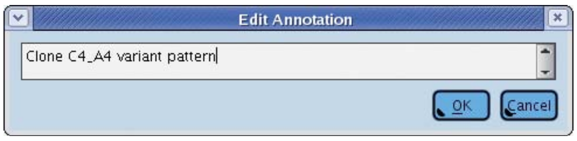
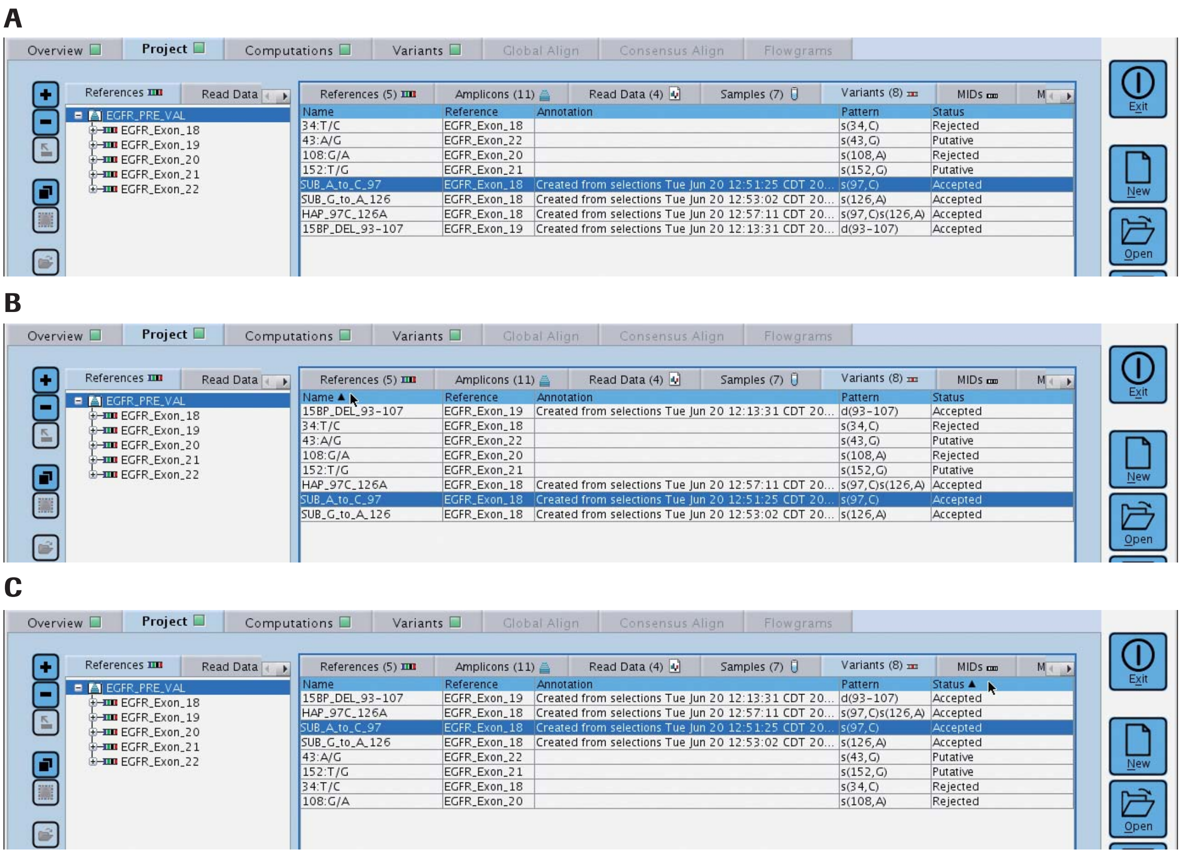


Figure 9–17: The Edit Annotation window, used to enter or edit the Annotation from any type of Project element

Each Definition Table sub-tab has a header row that labels the content of each column. The data in the table can be sorted by column content. A black triangle indicator appears to the right of the header label of a column if a sort has been applied to it: an upward-pointing triangle indicates that an ascending sort has been applied to the table data, while a downward-pointing triangle indicates a descending sort. Keep in mind that not all sorts are purely alphabetical or numerical; Project element Names, in particular, are broken down into numerical and non-numerical sections and the sections are sorted either numerically or alphabetically as appropriate.

Clicking on a column header that does not currently have a sort applied to it will cause an ascending sort of the table based on the data in that column. Clicking on a column header that already has an existing sort applied to it will toggle the sort type (ascending or descending). Only one column may have an active sort applied to it at a time, but the sort operations are stable and maintain the prior table order in cases where ties are encountered during the sort operation. This allows you to apply more than one sort at a time. For instance, if you wanted to sort the Variants Definition Table (Figure 9–18**A**) by status with the entries subsorted by variant name, you would first click on the 'Name' column header to sort by variant name (Figure 9–18**B**) and then you would click on the 'Status' column to sort the entries by status (Figure 9–18**C**).



A

Name	Reference	Annotation	Pattern	Status
34:T/C	EGFR_Exon_18		s(34,C)	Rejected
43:A/G	EGFR_Exon_22		s(43,G)	Putative
108:G/A	EGFR_Exon_20		s(108,A)	Rejected
152:T/G	EGFR_Exon_21		s(152,G)	Putative
SUB_A_to_C_97	EGFR_Exon_18	Created from selections Tue Jun 20 12:51:25 CDT 20...	s(97,C)	Accepted
SUB_G_to_A_126	EGFR_Exon_18	Created from selections Tue Jun 20 12:53:02 CDT 20...	s(126,A)	Accepted
HAP_97C_126A	EGFR_Exon_18	Created from selections Tue Jun 20 12:57:11 CDT 20...	s(97,C)s(126,A)	Accepted
15BP_DEL_93-107	EGFR_Exon_19	Created from selections Tue Jun 20 12:13:31 CDT 20...	d(93-107)	Accepted

B

Name ▲	Reference	Annotation	Pattern	Status
15BP_DEL_93-107	EGFR_Exon_19	Created from selections Tue Jun 20 12:13:31 CDT 20...	d(93-107)	Accepted
34:T/C	EGFR_Exon_18		s(34,C)	Rejected
43:A/G	EGFR_Exon_22		s(43,G)	Putative
108:G/A	EGFR_Exon_20		s(108,A)	Rejected
152:T/G	EGFR_Exon_21		s(152,G)	Putative
HAP_97C_126A	EGFR_Exon_18	Created from selections Tue Jun 20 12:57:11 CDT 20...	s(97,C)s(126,A)	Accepted
SUB_A_to_C_97	EGFR_Exon_18	Created from selections Tue Jun 20 12:51:25 CDT 20...	s(97,C)	Accepted
SUB_G_to_A_126	EGFR_Exon_18	Created from selections Tue Jun 20 12:53:02 CDT 20...	s(126,A)	Accepted

C

Name	Reference	Annotation	Pattern	Status ▲
15BP_DEL_93-107	EGFR_Exon_19	Created from selections Tue Jun 20 12:13:31 CDT 20...	d(93-107)	Accepted
HAP_97C_126A	EGFR_Exon_18	Created from selections Tue Jun 20 12:57:11 CDT 20...	s(97,C)s(126,A)	Accepted
SUB_A_to_C_97	EGFR_Exon_18	Created from selections Tue Jun 20 12:51:25 CDT 20...	s(97,C)	Accepted
SUB_G_to_A_126	EGFR_Exon_18	Created from selections Tue Jun 20 12:53:02 CDT 20...	s(126,A)	Accepted
43:A/G	EGFR_Exon_22		s(43,G)	Putative
152:T/G	EGFR_Exon_21		s(152,G)	Putative
34:T/C	EGFR_Exon_18		s(34,C)	Rejected
108:G/A	EGFR_Exon_20		s(108,A)	Rejected

Figure 9–18: (A) A Variants Definition Table with unsorted entries. (B) A Variants Definition Table with entries sorted by name. Clicking on the 'Name' header performs an ascending sort of the rows as indicated by the upward-pointing black triangle. (C) A Variants Definition Table with entries sorted by status. Clicking on the 'Status' header performs an ascending sort of the rows as indicated by the upward-pointing black triangle. Since an ascending name-sort was performed first (panel B), the rows within each 'Status' category have an ascending name-sort.

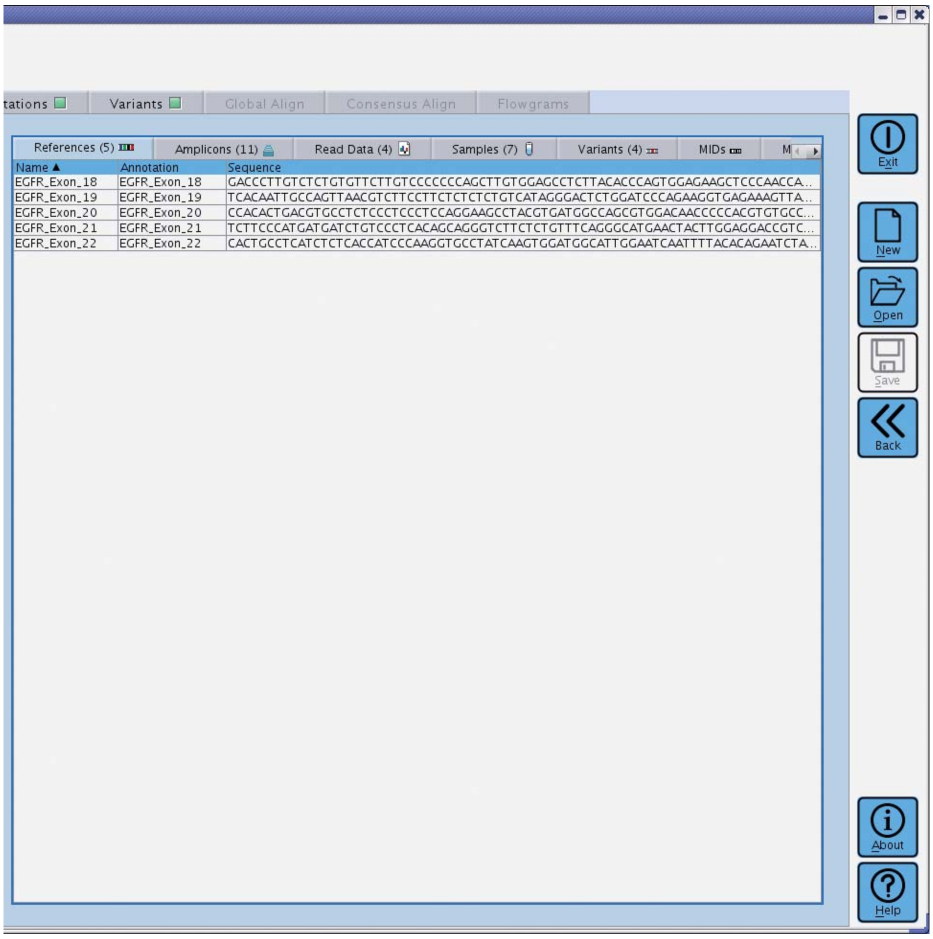
If you add a new entity to a Definition Table and a sort has been applied to a column for which the new row has a default value (such as the 'Name' field), the new row will be inserted into the table in its proper place according to the default value and the sort order. If you edit a cell in a sorted column and you change the value to something that would cause the row to change its position according to the sort order, however, sorting is automatically turned off for that column. This behavior prevents the row from moving out from under you as you edit a cell in a sorted column.

When sorting is turned off, none of the header labels for the table will have the black triangle indicator. You can re-enable sorting by clicking on any column header you want to sort by. If the first column header you click on for sorting is the same one that was used for sorting just prior to sorting deactivation, you will restore the sort order (ascending or descending) that was last used for the column. Clicking on any other column header will result in a default ascending sort for that column.

9.3.2.1 The References Definition Table

The **References** Definition Table lists all the Reference Sequences defined in the Project, with the following three characteristics (Table columns; see Figure 9–19):

- Name
- Annotation (free user-entered text)
- Sequence



Name ▲	Annotation	Sequence
EGFR_Exon_18	EGFR_Exon_18	GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACACCCAGTGGAGAAGCTCCCAACCA...
EGFR_Exon_19	EGFR_Exon_19	TCACAATTGCCAGTTAACGTCTTCCTTCTCTCTCTGTCATAGGGACTCTGGATCCCAGAAGGTGAGAAAGTTA...
EGFR_Exon_20	EGFR_Exon_20	CCACACTGACGTGCCTCTCCCTCCCTCCAGGAAGCCTACGTGATGGCCAGCGTGGACAACCCCCACGTGTGCC...
EGFR_Exon_21	EGFR_Exon_21	TCTTCCCATGATGATCTGTCCCTCACAGCAGGGTCTTCTCTGTTTCAGGGCATGAACTACTTGGAGGACCGTC...
EGFR_Exon_22	EGFR_Exon_22	CACTGCCTCATCTCTCACCATCCCAAGGTGCCTATCAAGTGGATGGCATTGGAATCAATTTTACACAGAATCTA...

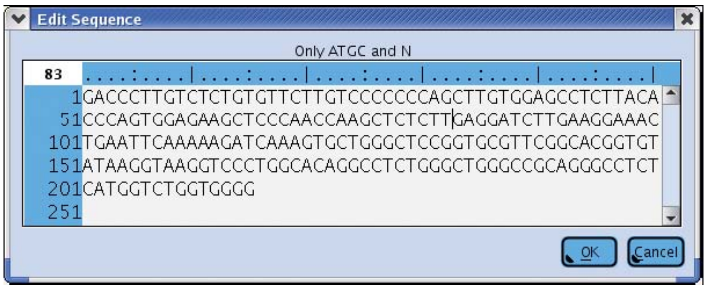
Figure 9–19: The References Definition Table sub-tab of the Project tab's right-hand panel

For the procedures used to add or remove Reference Sequences in a Project, see section 9.3.2 (or 9.3.1, to accomplish this in a "Project Tree" view). For the procedures used to enter/edit the Name or Annotation information for a Reference Sequence, see section 9.3.2. The sub-section below provide the procedure to enter/edit the other characteristic of Reference Sequences, the DNA sequence itself.

9.3.2.1.1 To Enter or Edit the DNA Sequence of a Reference Sequence

1. Double-click in the "Sequence" cell of the Reference Sequence you are defining. An "Edit Sequence" window will open (Figure 9–20).
2. Paste or type the sequence (only A, T, G, C, or N characters; see Caution, below).
3. Click "OK".

Characters restriction: Be aware that only "nucleotide" characters (A, T, G, C, or N) are accepted when you enter a Reference Sequence into the AVA software (by typing or pasting). For convenience, when pasting sequences, characters that are not nucleotide characters and are also not IUPAC ambiguity characters (such as R for purine, Y for pyrimidine, *etc.*) are removed from the pasted entry. This is useful when pasting sequences from sources that may include non-sequence information (such as white space or numerical position information in the margin of each line). During such pastes, any IUPAC ambiguity characters are converted to "N" characters, as the other ambiguity characters are not supported by the software (typing individual "ambiguous" characters, however, does not result in their conversion to "N"; these are simply ignored and the text "Only ATGC and N" at the top of the Edit Sequence window turns bold and red to alert you that an invalid character was used). The restriction that no ambiguity characters other than N be present in a sequence is a requirement of many alignment algorithms and is not unique to the Genome Sequencer System software.



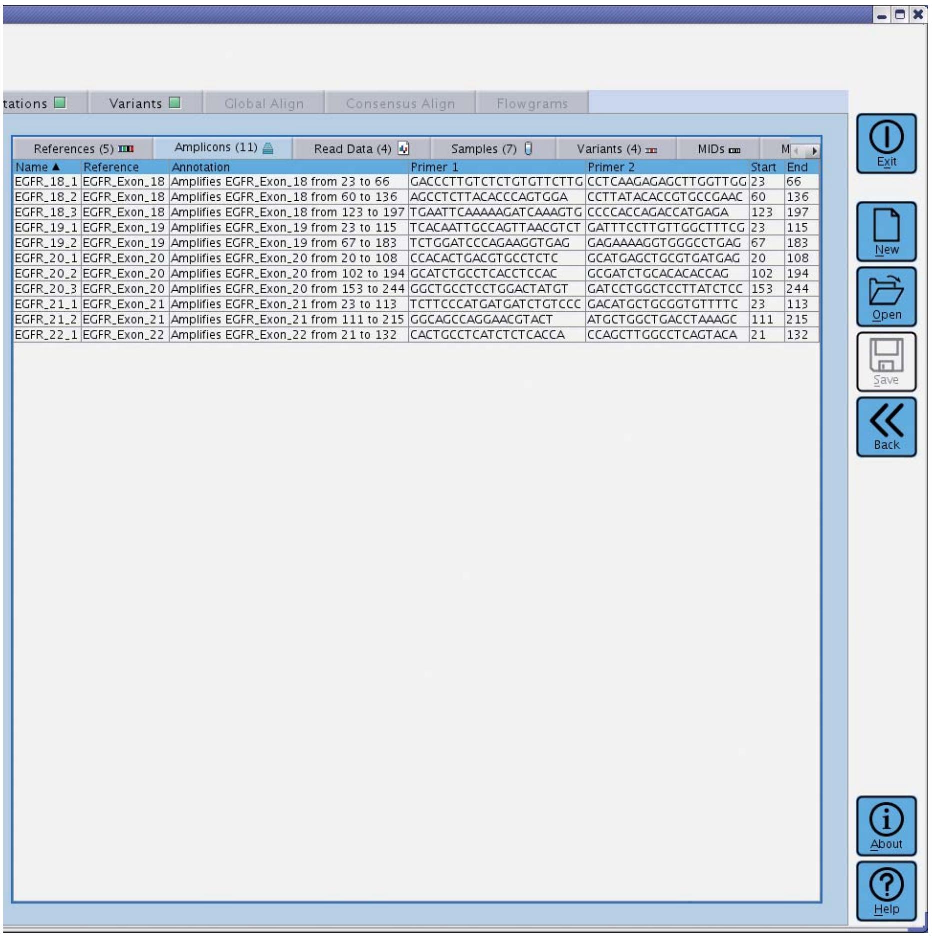
```
  1GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACA
 51CCCAGTGGAGAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGAAAC
101TGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGCGTTCGGCACGGTGT
151ATAAGGTAAGGTCCCTGGCACAGGCCTCTGGGCTGGGCCGCAGGGCCTCT
201CATGGTCTGGTGGGG
251
```

Figure 9–20: The Edit Sequence window, used to enter or edit the DNA sequence of a Reference Sequence element

9.3.2.2 The Amplicons Definition Table

The **Amplicons** Definition Table lists all the Amplicons defined in the Project, with the following seven characteristics (Table columns; see Figure 9–21):

- Name
- Reference Sequence (to which the Amplicon is associated)
- Annotation (free user-entered text)
- Primer 1 (5'→3' sequence of the forward primer of the Amplicon)
- Primer 2 (5'→3' sequence of the reverse primer of the Amplicon)
- Start (first nucleotide of the Target, on the Reference Sequence)
- End (last nucleotide of the Target, on the Reference Sequence)



Name ▲	Reference	Annotation	Primer 1	Primer 2	Start	End
EGFR_18_1	EGFR_Exon_18	Amplifies EGFR_Exon_18 from 23 to 66	GACCCTTGTCTCTGTGTTCTTG	CCTCAAGAGAGCTTGGTTGG	23	66
EGFR_18_2	EGFR_Exon_18	Amplifies EGFR_Exon_18 from 60 to 136	AGCCTCTTACACCCAGTGGA	CCTTATACACCGTGCCGAAC	60	136
EGFR_18_3	EGFR_Exon_18	Amplifies EGFR_Exon_18 from 123 to 197	TGAATTCAAAAAGATCAAAGTG	CCCCACCAGACCATGAGA	123	197
EGFR_19_1	EGFR_Exon_19	Amplifies EGFR_Exon_19 from 23 to 115	TCACAATTGCCAGTTAACGTCT	GATTTCCTTGTTGGCTTTCG	23	115
EGFR_19_2	EGFR_Exon_19	Amplifies EGFR_Exon_19 from 67 to 183	TCTGGATCCCAGAAGGTGAG	GAGAAAAGGTGGGCCTGAG	67	183
EGFR_20_1	EGFR_Exon_20	Amplifies EGFR_Exon_20 from 20 to 108	CCACACTGACGTGCCTCTC	GCATGAGCTGCGTGATGAG	20	108
EGFR_20_2	EGFR_Exon_20	Amplifies EGFR_Exon_20 from 102 to 194	GCATCTGCCTCACCTCCAC	GCGATCTGCACACACCAG	102	194
EGFR_20_3	EGFR_Exon_20	Amplifies EGFR_Exon_20 from 153 to 244	GGCTGCCTCCTGGACTATGT	GATCCTGGCTCCTTATCTCC	153	244
EGFR_21_1	EGFR_Exon_21	Amplifies EGFR_Exon_21 from 23 to 113	TCTTCCCATGATGATCTGTCCC	GACATGCTGCGGTGTTTTC	23	113
EGFR_21_2	EGFR_Exon_21	Amplifies EGFR_Exon_21 from 111 to 215	GGCAGCCAGGAACGTACT	ATGCTGGCTGACCTAAAGC	111	215
EGFR_22_1	EGFR_Exon_22	Amplifies EGFR_Exon_22 from 21 to 132	CACTGCCTCATCTCTCACCA	CCAGCTTGGCCTCAGTACA	21	132

Figure 9–21: The Amplicons Definition Table sub-tab of the Project tab's right-hand panel

For the procedures used to add or remove Amplicons in a Project, see section 9.3.2 (or 9.3.1, to accomplish this in a "Project Tree" view, and concurrently create associations). For the procedures used to enter/edit the Name or Annotation information for an Amplicon, see section 9.3.2. The sub-sections below provide the procedures to enter/edit the other characteristics of Amplicons.

9.3.2.2.1 To Enter or Edit the Reference Sequence to which an Amplicon is associated

1. Ensure that the column labeled "Reference" in the table is wide enough to allow you to distinguish among the Reference Sequences from which you want to select. To widen the column, click on the separator line, in the table header, between the Reference and Annotation columns, and drag the separator to the right.
2. Double-click in the "Reference Sequence" cell for the Amplicon you are defining. A drop down menu will expand, showing all the Reference Sequences currently listed in the Project.
3. Select the proper Reference Sequence from the drop down menu. The new association will automatically appear on the References Tree, on the left panel.

If the Reference Sequence does not yet contain a DNA sequence (see section 9.3.2.1.1), you will still be able to associate Amplicons to it, but you will not be able to fully define them. In particular, you will not be able to specify the Target Start and End for the Amplicons (see section 9.3.2.2.3, below) because these are set using the position numbering from the Reference Sequence.

9.3.2.2.2 To Enter or Edit the Primer Sequences for the Amplicon

As mentioned earlier (section 9.1.1.3), Primer 1 and Primer 2 correspond to the "sequence-specific" part of the two Fusion Primers used to construct the Amplicon library, excluding the 19 bp "Primer A" and "Primer B" parts of the Fusion Primers (see the *GS FLX Amplicon DNA Library Preparation Method Manual* for details on the structure of sequencing templates from Amplicon libraries).

- ▶ Both Primer 1 and Primer 2 should be entered as their true 5'→3' sequence. To find the End of the Target (section 9.3.2.2.3, below), the software automatically determines the reverse-complement of Primer 2 (Primer 2') and aligns this to the Reference Sequence.

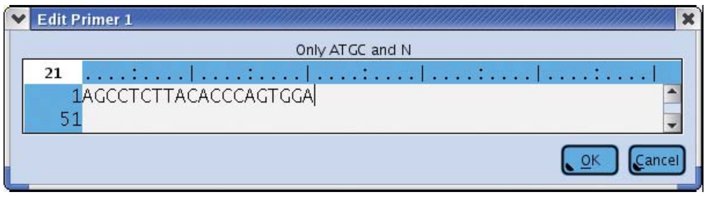
- ▶ The AVA software does not require any knowledge of A vs. B beads from the GS emPCR Kits; reads that align in the same orientation as the given Reference Sequence are considered 'forward' reads, and those that must be reverse complemented to align are considered 'reverse' reads.

1. Double-click in the "Primer 1" (or "Primer 2") cell for the Amplicon you are defining. An "Edit Primer 1" (or "Edit Primer 2") window will open (Figure 9–22).
2. Paste or type the sequence (only A, T, G, C, or N characters; see Caution, below).
3. Click "OK".

■

Characters restriction: Be aware that only "nucleotide" characters (A, T, G, C, or N) are accepted when you enter a Primer Sequence into the AVA software (by typing or pasting). For convenience, when pasting sequences, characters that are not nucleotide characters and are also not IUPAC ambiguity characters (such as R for purine, Y for pyrimidine, *etc.*) are <u>removed</u> from the pasted entry. This is useful when pasting sequences from sources that may include non-sequence information (such as white space or numerical position information in the margin of each line). During such pastes, any IUPAC ambiguity characters are converted to "N" characters, as the other ambiguity characters are not supported by the software (typing individual "ambiguous" characters, however, does not result in their conversion to "N"; these are simply ignored and the text "Only ATGC and N" at the top of the Edit Sequence window turns bold and red to alert you that an invalid character was used). The restriction that no ambiguity characters other than N be present in a sequence is a requirement of many alignment algorithms and is not unique to the Genome Sequencer System software.

Although the AVA software allows "N" characters in the primer sequences, the primer trimming and demultiplexing computational steps perform better if only A, T, G, or C characters are used. If your primer design involves "wobble" positions in which more than one base may appear, it is preferable to define the primer sequence with one of the alternative bases rather than an N at those positions. For record keeping purposes, you may document the choice of data entry in the corresponding Amplicon's Annotation field.



```
1AGCCTCTTACACCCAGTGGA
```

Figure 9–22: The Edit Primer 1 window, used to enter or edit the DNA sequence of Primer 1 for an Amplicon element

9.3.2.2.3 To Enter or Edit the Target Start and End Positions

There is no requirement that the Reference Sequence contain either of the two Primer sequences, although they typically will be contained therein. The Target, however, must be present in the Reference Sequence, and the Target Start and End positions indicate the first and last bases of the Target, inclusive, relative to the Reference Sequence.

To simplify the setting of the Start and End values, and reduce the risk of data entry errors, the AVA software searches for the Primers within the Reference Sequence, on the assumption that they will most likely be present and, if found, uses the primer positions to establish default values for the Target Start and End (see below). As such, prior to entering or editing the Target Start and End positions, you must define the sequence of the Reference Sequence with which the Amplicon is associated, as well as the two Amplicon Primers (sections 9.3.2.1.1 and 9.3.2.2.2).

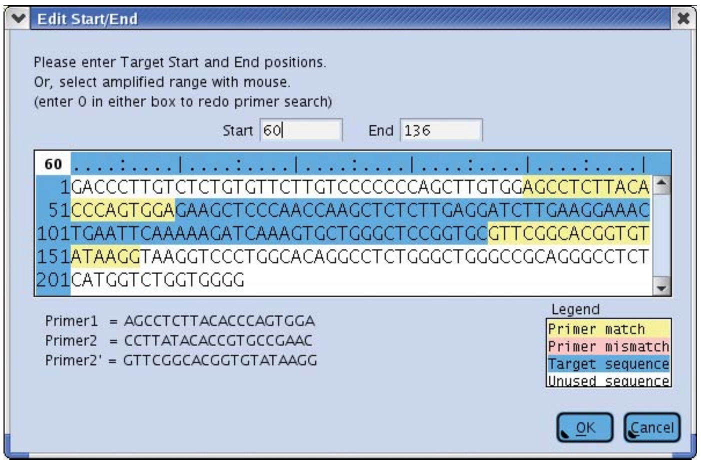
1. Double-click in either the “Start” or the “End” cell for the Amplicon you are defining. Clicking in either the “Start” or the “End” cell opens the same “Edit Start/End” window (Figure 9–23). This window includes:

   a. a brief set of instructions

   b. a pair of data entry boxes for the Start and End nucleotides

   c. the Reference Sequence to which the Amplicon is associated, with a color-coded overlay indicating the Primer sequences (matched and mismatched), the Target sequence (the part of the Reference Sequence between the two Primers), and the “unused” sequence (the part of the Reference Sequence outside the two Primers)

   d. the sequence of the two Primers, plus the reverse-complement of Primer 2 (Primer 2’)



```
Edit Start/End

Please enter Target Start and End positions.
Or, select amplified range with mouse.
(enter 0 in either box to redo primer search)

                    Start 60          End 136

60  ....:....|....:....|....:....|....:....|....:....|
  1GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACA
 51CCCAGTGGAGAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGAAAC
101TGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGCGTTCGGCACGGTGT
151ATAAGGTAAGGTCCCTGGCACAGGCCTCTGGGCTGGGCCGCAGGGCCTCT
201CATGGTCTGGTGGGG

Primer1  = AGCCTCTTACACCCAGTGGA
Primer2  = CCTTATACACCGTGCCGAAC
Primer2' = GTTCGGCACGGTGTATAAGG
```

Legend
Primer match
Primer mismatch
Target sequence
Unused sequence

Figure 9–23: The Edit Start/End window, used to define the start and end of the Target for an Amplicon element (the part excluding the Primers), by locating the Primers 1 and 2 on the Reference Sequence with which the Amplicon is associated.

▶▶▶

2 There are 3 ways to set (or reset) the Start and End nucleotides of the Target:

a. If the Target's Start and End have not been specified for this Amplicon before (*i.e.* the Start and End cells were empty when you double-clicked them), the software automatically searches for the Primers (Primer 1 and Primer 2') in the Reference Sequence; if it finds them (exact matches only), the software marks the Primers in yellow and the Target sequence (between the two Primers) in blue, and specifies default values for the Target's Start and End positions in the boxes at the top of the window. The user should verify that the default positions are correct since, in some rare circumstances, there may be multiple Primer1-Primer2' pairs of matches within the same Reference Sequence and the software simply gives the first such pair it finds. This Primer search function can also be elicited by typing a "0" (or a negative number) in either the Start or the End entry box. It is possible that exact matches for the Primers are not found in the Reference Sequence, as either or both Primers may actually not be represented by the Reference Sequence or, due to design considerations (or primer synthesis or sequencing errors), the Primers may slightly differ from the Reference Sequence so that they have a close, but inexact match. Whatever the reason, if no exact match can be found for Primer1, the AVA software will default the Target Start to the first base of the Reference Sequence; if no exact match can be found for Primer2', the default for Target End will be the last base of the Reference Sequence. If this happens, verify that you have correctly defined the Primer and the Reference Sequence to which this Amplicon is associated; if the sequences are correct, but the default values supplied are incorrect, use one of the following methods to specify the Target Start and End positions.

b. If you know the exact positions of the Target's Start and End relative to the Reference Sequence, you can type them in the entry boxes at the top of the window. The AVA software will automatically update the color-coded display to indicate how well Primer1 and Primer2' match to the Reference Sequence in the regions abutting the supplied Start and End positions. If you know that the Reference Sequence doesn't actually contain the Primers themselves, then you can safely ignore this feedback. However, if the Reference Sequence is supposed to contain the primers and there are one or more mismatched bases indicated by pink highlighting in the display, you should check that you entered the Start and End positions correctly, that you entered the correct Primers (and that both are in the 5'→3' orientation), that you have the correct Amplicon-Reference sequence association, and that the Reference Sequence itself has the correct sequence.

c. You can also use the mouse drag method: the software interprets click-and-dragging the mouse in the sequence as an attempt to select the amplified range (the Target), so it aligns the sequence beyond the drag point with the Primers (Primer 2' if dragging to the right and Primer 1 if dragging to the left). Matching nucleotides are overlaid in yellow and non-matching nucleotides, in pink. This method may be especially useful if the Primers you used to generate the Amplicon library did not exactly match the Reference Sequence you are using for analysis.
To use this method, do the following:

I. Click near the beginning of the Reference Sequence and drag the mouse to the right. The stretch of sequence (the length of Primer 2') beyond the drag point will be displayed in color to indicate matches with the Primer. Stop and release the mouse button when the whole stretch is yellow, indicating a perfect match with Primer 2' (or allow for mismatches, shown in pink, if appropriate). The software enters the start and stop points of the dragging action in the Start and End entry boxes, respectively; this sets the Target's End nucleotide.

II. Click again on the last nucleotide of the Target, as just defined, and drag the mouse to the left. Stop and release the mouse as before, when you have located Primer 1. Again, the software enters the start and stop points of the dragging action, but in the End and Start entry boxes, respectively; this sets both the Target's Start and End nucleotides.

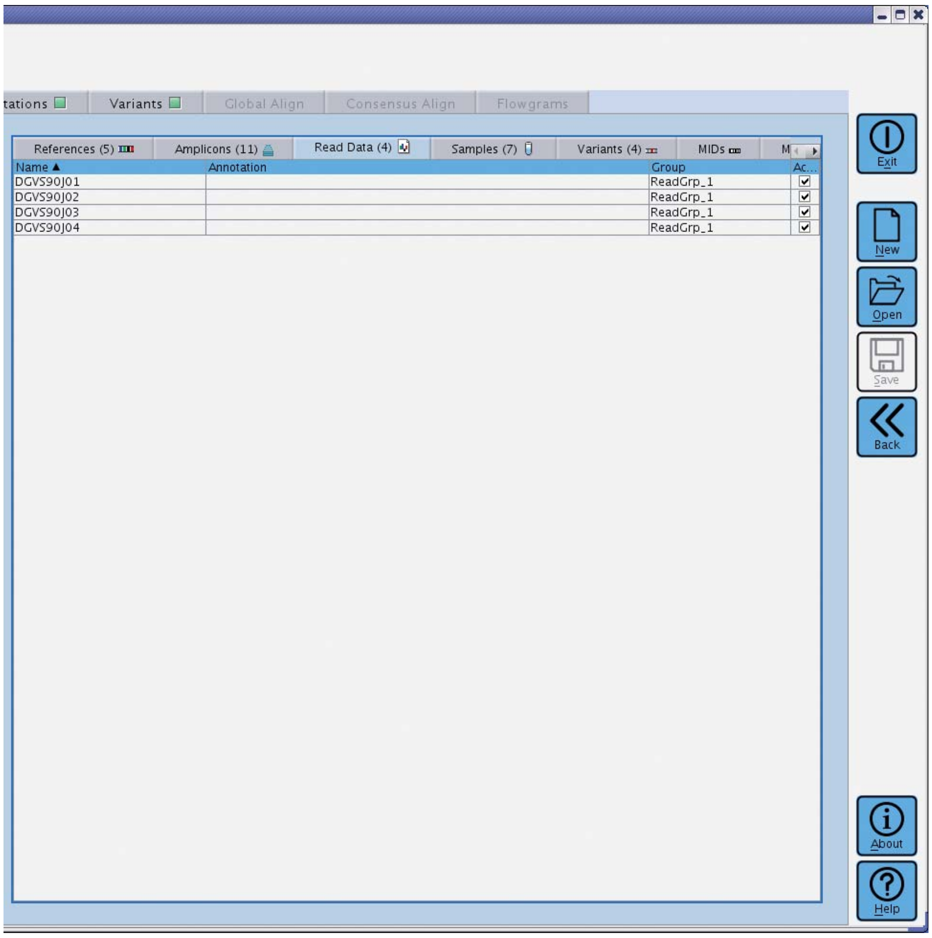
3 Click "OK".

■

9.3.2.3 The Read Data Definition Table

The **Read Data** Definition Table lists all the Read Data Sets defined in the Project, with the following four characteristics (Table columns; see Figure 9–24):

- Name
- Annotation (free user-entered text)
- Group (the Read Group to which the Read Data Set belongs)
- Active (if checked, the Read Data Set will be included in the next computation of the Project)



Name ▲	Annotation	Group	Ac...
DGVS90J01		ReadGrp_1	✓
DGVS90J02		ReadGrp_1	✓
DGVS90J03		ReadGrp_1	✓
DGVS90J04		ReadGrp_1	✓

Figure 9–24: The Read Data Definition Table sub-tab of the Project tab's right-hand panel

For the procedures used to add or remove Read Data Sets in a Project, see section 9.3.2 (or 9.3.1, to accomplish this in a "Project Tree" view, and concurrently create associations). For the procedures used to enter/edit the Name or Annotation information for a Read Data Set, see section 9.3.2. The sub-sections below provide the procedures to enter/edit the other characteristics of Read Data Sets.

In the current software release, there is a restriction on the use of SFF Tools to process SFF files before importing them into an AVA Project: no two Read Data Sets can contain reads with the same accno prefixes. Since accno prefixes identify SFF files that the reads come from, the disallowed situation could occur if one split an SFF file and then loaded two or more of the resulting parts into a Project as separate files: the reads from all the parts would all have the same accno prefix as the original file. If two files contain reads with the same accno prefix, the Alignment Phase of the computation will fail with a "Done – Error" error message.

Additionally, in the current release, if the user renames SFF files prior to import into an AVA Project, such that the read accnos' prefixes don't match the SFF file name, there will be no error during computation, but the software will be unable to retrieve the corresponding reads' flowgrams in the GUI when using the right-click option from the Global and Consensus Alignment tabs.

9.3.2.3.1 To Edit the Read Group of a Read Data Set

If you want to transfer a Read Data Set to another pre-existing Read Data Group, double-click the drop down menu in the Group cell for the Read Data Set and select the Read Group you want from the available choices. You can also reassign a Read Data Set to a different Read Group by dragging the Read Data Set to a Read Group node of the Read Data Tree (a multiple-selection of Read Data Sets will assign them all to the Read Group on which you drop them). While you cannot change the name of a Read Group from within the Read Data Definition Table, you can do so in the Read Data Tree (as with any other rename operation in the tree, click once on the Group name, pause and click a second time to activate the name editor). Note that all Read Groups are distinct entities, and although you can rename an existing Read Group to match the name of another pre-existing Read Group, this will not cause the Read Data Sets to be merged into the pre-existing group.

Note also that you cannot import the same Read Data Set more than once in a Project, even if you intend to assign them to different Read Groups. If you attempt to do so, an error message will appear on screen.

9.3.2.3.2 To Edit the "Active" status of a Read Data Set

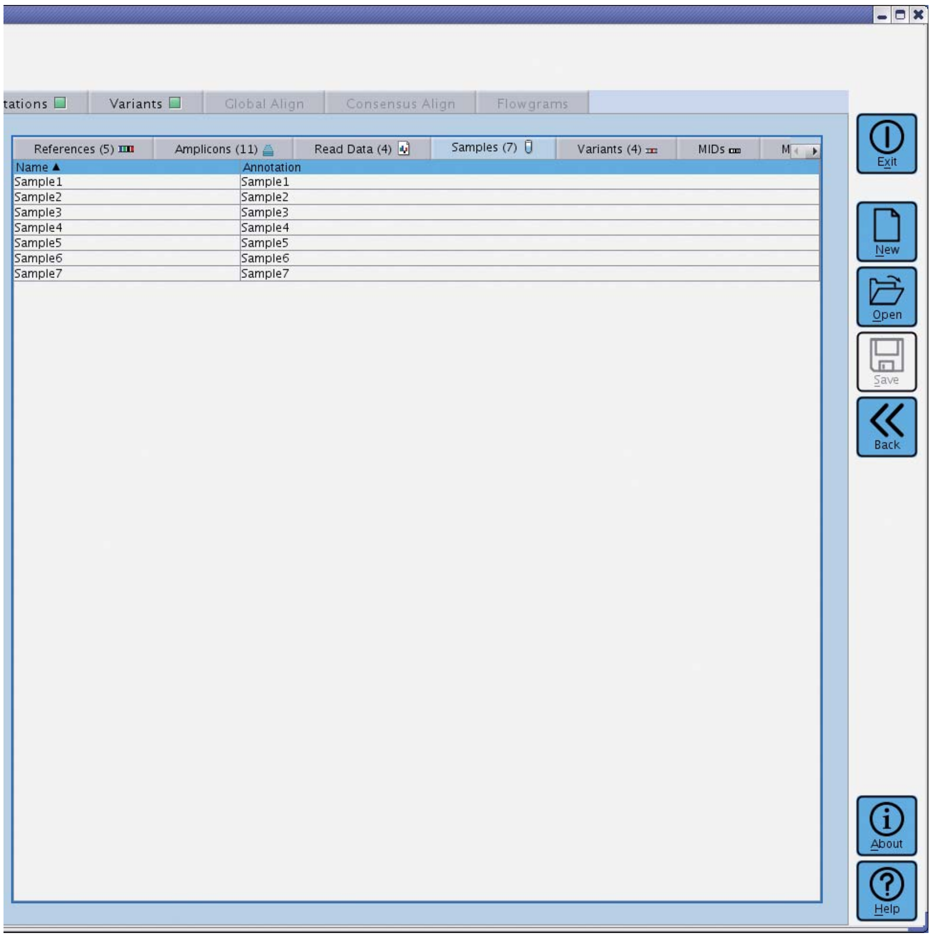
You may elect to include or exclude from future computations, any Read Data Set defined in the Project. Excluding a Read Data Set from future computations does not remove it from the Project. If you discover that a Read Data Set is unsuitable for the Project in some manner, it can be useful to keep it in the project but mark it as inactive (and possibly update its Annotation) to serve as a reminder not to reintroduce the data at some future point.

1. To include a Read Data Set in future computations of the Project, check its box in the "Active" column of the Read Data Definition Table.
2. To exclude a Read Data Set in future computations of the Project, uncheck its box in the "Active" column of the Read Data Definition Table.

9.3.2.4 The Samples Definition Table

The **Samples** Definition Table lists all the Samples defined in the Project, with only the following two characteristics (Table columns; see Figure 9–25):

- Name
- Annotation (free user-entered text)



Name ▲	Annotation
Sample1	Sample1
Sample2	Sample2
Sample3	Sample3
Sample4	Sample4
Sample5	Sample5
Sample6	Sample6
Sample7	Sample7

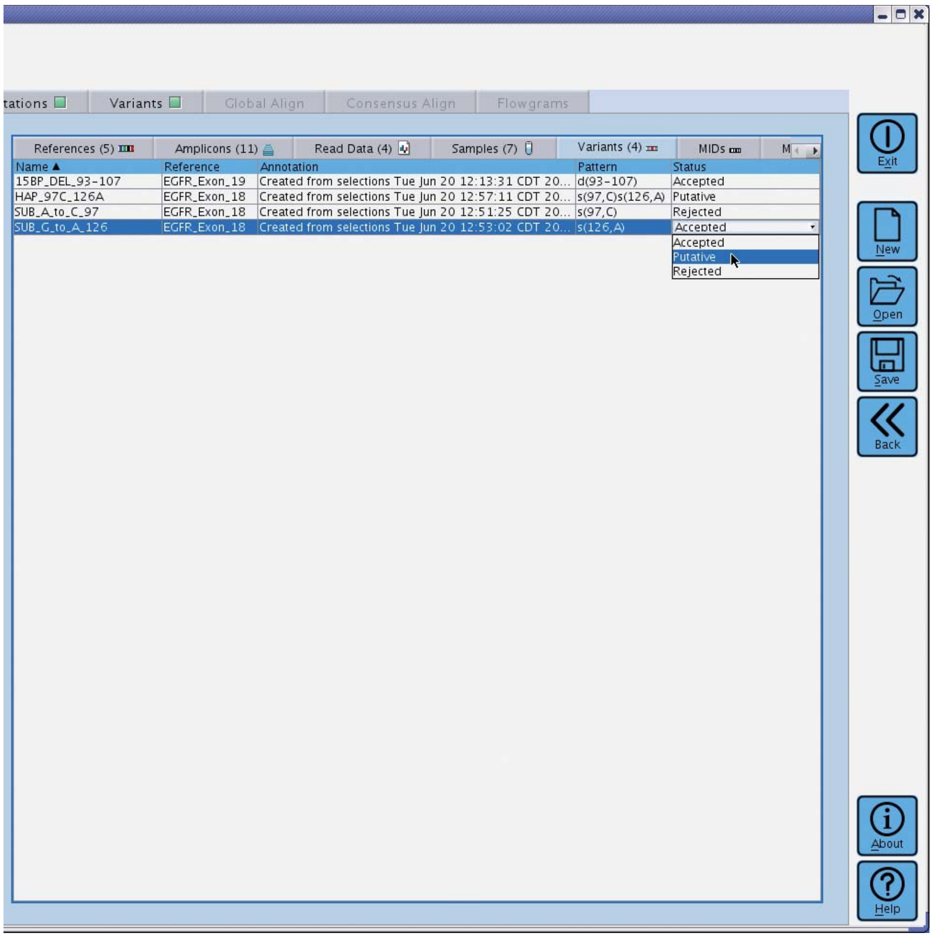
Figure 9–25: The Samples Definition Table sub-tab of the Project tab's right-hand panel

For the procedures used to add or remove Samples in a Project, see section 9.3.2 (or 9.3.1, to accomplish this in a "Project Tree" view, and concurrently create associations). For the procedures used to enter/edit the Name or Annotation information for a Sample, see section 9.3.2.

9.3.2.5 The Variants Definition Table

The **Variants** Definition Table lists all the Variants defined in the Project, with the following five characteristics (Table columns; see Figure 9–26):

- Name
- Reference Sequence (with which the Variant is associated)
- Annotation (free user-entered text)
- Pattern (definition of the nature of the Variant)
- Status (workflow category)



Name ▲	Reference	Annotation	Pattern	Status
15BP_DEL_93-107	EGFR_Exon_19	Created from selections Tue Jun 20 12:13:31 CDT 20...	d(93-107)	Accepted
HAP_97C_126A	EGFR_Exon_18	Created from selections Tue Jun 20 12:57:11 CDT 20...	s(97,C)s(126,A)	Putative
SUB_A_to_C_97	EGFR_Exon_18	Created from selections Tue Jun 20 12:51:25 CDT 20...	s(97,C)	Rejected
SUB_G_to_A_126	EGFR_Exon_18	Created from selections Tue Jun 20 12:53:02 CDT 20...	s(126,A)	Accepted

Accepted
Putative
Rejected

Figure 9–26: The Variants Definition Table sub-tab of the Project tab's right-hand panel

For the procedures used to add or remove Variants in a Project, see section 9.3.2 (or 9.3.1, to accomplish this in a "Project Tree" view, and concurrently create associations). For the procedures used to enter/edit the Name or Annotation information for a Variant, see section 9.3.2. The sub-sections below provide the procedures to enter/edit the other characteristics of Variants.

9.3.2.5.1 To Enter or Edit the Reference Sequence to which a Variant is associated

1. Ensure that the column labeled “Reference” in the table is wide enough to allow you to distinguish among the Reference Sequences from which you want to select. To widen the column, click on the separator line, in the table header, between the Reference and Annotation columns, and drag the separator to the right.
2. Double-click in the “Reference Sequence” cell for the Variant you are defining. A drop down menu will expand, showing all the Reference Sequences currently listed in the Project.
3. Select the proper Reference Sequence from the drop down menu. The new association will automatically appear on the References Tree, on the left panel.

If the Reference Sequence does not yet contain a DNA sequence (see section 9.3.2.1.1), you will still be able to associate Variants to it, but you will not be able to fully define them. In particular, you will not be able to specify the Pattern for the Variant (see section 9.3.2.5.2, below) because this is set using the position numbering from the Reference Sequence.

9.3.2.5.2 To Enter or Edit the Pattern of a (Known) Variant

If you already know one or more Variants (*e.g.* from the scientific literature or from previous experiments), you can define them in the Project and have the AVA software report on the frequency at which they occur in the Read Data Sets included in the Project. Note that novel Variants observed in the reads of the Project itself can also be defined as described below, but the best way to specify novel Variants is to examine the multiple alignments of the putative Variants found by the AVA software during computation and to “Accept” them if you determine that they are legitimate (see section 9.3.2.5.3, below); also, you can “declare” novel Variants not identified by the software after you identify and evaluate them in the Global Align or Consensus Align tabs (see sections 9.6 and 9.7).

The AVA software uses 4 types of constraints to define Variants, and writes them following a strict Variant Definition Syntax, summarized in Table 9–1. A Variant can be specified by one or more constraints which, collectively comprise the “Pattern” that defines the Variant.

Constraint Type	Syntax	Description
Must match	m(p) or m(p_1-p_2)	A read satisfies this constraint when the nucleotide(s) at position “p” or in the range “p_1-p_2” (inclusive) of the Reference Sequence are identical to those of the Reference Sequence.
Substitute base	s(p,n)	A read satisfies this constraint when the nucleotide at position “p” is “n” (where “n” is different from the nucleotide at that position in the Reference Sequence).
Insert bases	i(p.5, n…)	A read satisfies this constraint when the (one or more) nucleotide(s) “n…” are present between positions “p” and “p+1” of the Reference Sequence. Note that a deletion may not also exist at positions “p” or “p+1”, as this combination of insertion and deletion would rather define a substitution.
Delete bases	d(p) or d(p_1-p_2)	A read satisfies this constraint when the nucleotide(s) at position “p” or in the range “p_1-p_2” (inclusive) of the Reference Sequence are absent. Note that directly neighboring insertions may not also exist, as this combination would rather define a substitution.

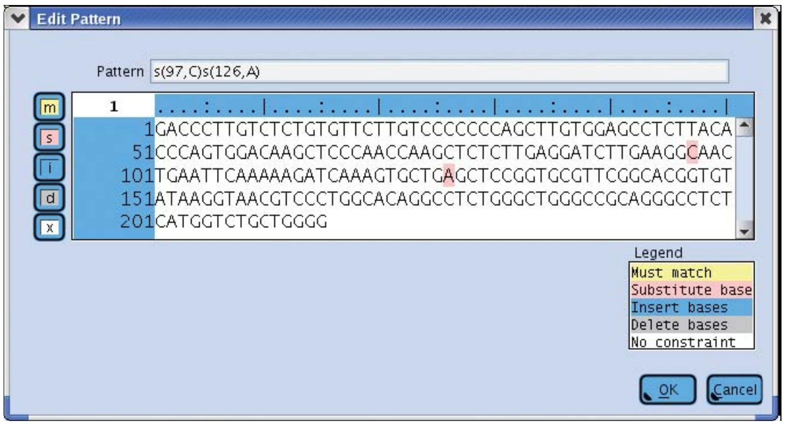
Table 9–1: The language of variations in the AVA software: the Variant Definition Syntax. A Variant may comprise multiple constraints (though any given nucleotide may only have a single constraint) enabling the encoding of haplotypes or other complicated variation Patterns. In these more complicated cases, the Variant is encoded by specifying multiple, concatenated constraints, optionally separated by whitespace.

To enter or edit the Pattern defining a Variant in the Variants sub-tab of the Project tab, do the following:

1. Double-click in the “Pattern” cell for the Variant you are defining. An “Edit Pattern” window will open (Figure 9–27). This window includes:

   a. a “Pattern” data entry box to define and view the nature of the Variant, using the Variant Definition Syntax (see above)

   b. a box containing a DNA sequence based on the Reference Sequence to which the Variant is associated, with a color-coded overlay to facilitate the graphical definition and visualization of the Variant



Edit Pattern

Pattern s(97,C)s(126,A)

```
  1  ....:....|....:....|....:....|....:....|....:....|
  1GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACA
 51CCCAGTGGACAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGCAAC
101TGAATTCAAAAAGATCAAAGTGCTGAGCTCCGGTGCGTTCGGCACGGTGT
151ATAAGGTAACGTCCCTGGCACAGGCCTCTGGGCTGGGCCGCAGGGCCTCT
201CATGGTCTGCTGGGG
```

Legend

Must match

Substitute base

Insert bases

Delete bases

No constraint

Figure 9–27: The Edit Pattern window, used to specify the difference(s) compared to the Reference Sequence that define the Variant

2. There are 2 ways to set (or reset) the definition of a Variant, using this window:

   a. You can enter it directly in the Pattern box at the top of the window, using the Variant Definition Syntax (see above). The software automatically adds variations entered in the Pattern box to the graphic sequence box, below it. If the syntax used is incorrect, the AVA software parses the entry and suggests a correction or provides a tip, in the area below the sequence box

   b. You can enter it graphically by selecting nucleotide(s) in the sequence field and assigning the appropriate type of constraint using the buttons to the left of the sequence box. The software automatically adds constraints entered graphically, in the Pattern box. There are 4 buttons, matching the four types of constraints, and a fifth button for clearing a previously specified constraint, that can be used to graphically define a Variant. The 5 buttons, and their use, are as follows:

   m I. Must match (shown in yellow overlay)
      01. Select one nucleotide (click) or a nucleotide range (click-and-drag) in the sequence.
      02. Click the Must match button.

   s II. Substitute base (shown in pink overlay)
      01. Select one nucleotide (click) in the sequence.
      02. Click the Substitute base button; the “One base” window will open (not shown).
      03. Type the substituting nucleotide.
      04. Click OK; the sequence changes to that of the Variant.

   i III. Insert bases (shown in blue overlay)
      01. Select the one nucleotide (click) in the sequence before which the insertion is located.
      02. Click the Insert bases button; the “Enter insert sequence” window will open (not shown).
      03. Type the nucleotide(s) to be inserted (only A, T, G, or C characters).
      04. Click OK; the insertion appears in the sequence. The position of the inserted nucleotides use decimals so that the original Reference Sequence positions are maintained (*e.g.* position 66.5 means that the insertion is between the nucleotides at positions 66 and 67 of the Reference Sequence).

▶▶▶

2 d IV. Delete bases (shown in gray overlay)

01. Select one nucleotide (click) or a nucleotide range (click-and-drag) in the sequence.

02. Click the Delete bases button; the nucleotides in the sequence are replaced with dashes.

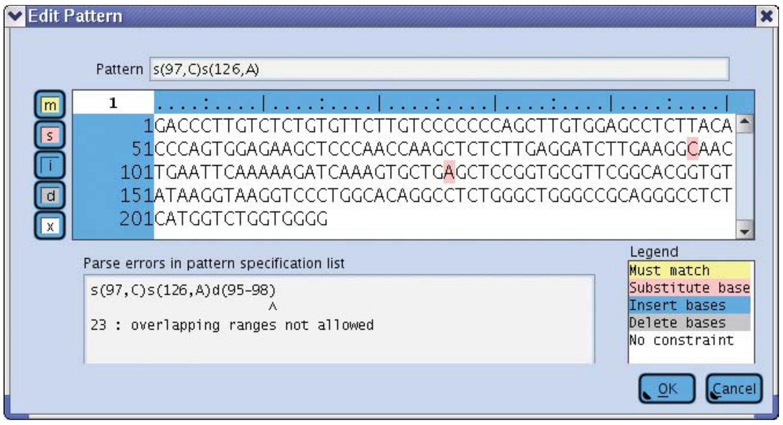
x V. No constraint (shown with white background; the Reference Sequence)

01. Select one nucleotide (click) or a nucleotide range (click-and-drag) in the sequence that already have one of the above changes assigned

02. Click the No constraint button; the nucleotide(s) in the sequence revert to the Reference Sequence. (This function is useful if you incorrectly entered a constraint in the definition of a Variant.)

3 Click OK.

If an erroneous pattern is directly entered into the Pattern field of the “Edit Pattern” window, the AVA software will remove portions of the pattern until what remains is valid both syntactically and semantically, as seen in Figure 9–28. In this case, one or more parsing error messages will appear describing the nature of the problem.


Edit Pattern

Pattern s(97,C)s(126,A)

```
    1 ....:....|....:....|....:....|....:....|....:....|
    1GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACA
   51CCCAGTGGAGAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGCAAC
  101TGAATTCAAAAAGATCAAAGTGCTGAGCTCCGGTGCGTTCGGCACGGTGT
  151ATAAGGTAAGGTCCCTGGCACAGGCCTCTGGGCTGGGCCGCAGGGCCTCT
  201CATGGTCTGGTGGGG
```

Parse errors in pattern specification list

```
s(97,C)s(126,A)d(95-98)
                      ^
23 : overlapping ranges not allowed
```

Legend
Must match
Substitute base
Insert bases
Delete bases
No constraint

Figure 9–28: The Edit Pattern window, with an error in the pattern specification. The user attempted to specify both a substitution at positions 97 and 126 as well as a deletion spanning positions 95–98. Since position 97 cannot both be deleted and involved in a substitution, the software automatically removed the deletion from the Pattern specification at the top of the window, but shows the full erroneous pattern with an appropriate error message in an error window. The compatible substitution at position 126 is left as part of the pattern.

9.3.2.5.3 To Edit the Status of a Variant

Variants exist within a Project, with one of three possible Status values: “Accepted”, “Putative”, or “Rejected”. Variants defined manually by the user (see section 9.3.2.5.2) receive the “Accepted” status by default. By contrast, variations between the Read Data Sets and the References that are identified by the AVA software (during computation; see section 9.4), are initially proposed as “Putative” Variants. After you have examined the data underlying a Variant and determined whether you believe it to be legitimate or not, you can change its assigned Status as described below.

Note, however, that the main use of the Variant Status feature is as part of a “Discovery Workflow”, on the Variants tab. The purpose of this process is precisely to determine whether the Variants included in the Project appear to be legitimate or not and to mark them as such (“Accepted” or “Rejected”); The Variants tab is also where the control is located to “load” the putative Variants discovered by the software, into the Project. For these reasons, Variant Status assignment might more often be done on the Variants tab than on the Variants sub-tab of the Project tab, as described here. See section 9.5 for a description of the Variants tab and for more details on this Discovery Workflow.

To edit the Status of a Variant in the Variants sub-tab of the Project tab, do the following:

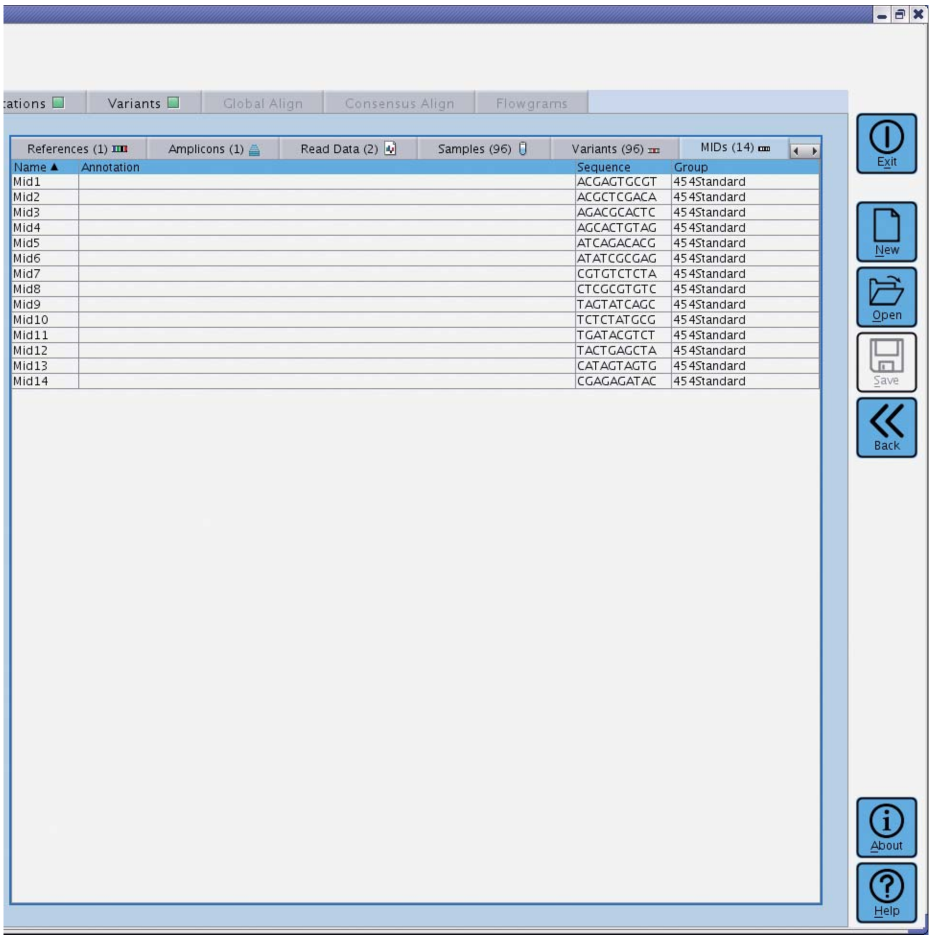
1. Ensure that the “Status” column in the Variants Definition Table is wide enough to allow you to distinguish among the Status choices. To widen the column, click on the separator line, in the table header, between the Pattern and Status columns, and drag the separator to the left.
2. Double-click in the “Status” cell for the Variant you are defining. A drop down menu will expand, showing the available Status options.
3. Select the proper Status from the drop down menu.

- If you open a Project from a prior version of the AVA software which did not have the Status field, the status value for any pre-existing Variants in the Project will be set to “Accepted”.
- It is often more useful to use the “Rejected” status than to actually remove a Variant from the Project. When you mark a Variant as rejected, you prevent it from being rediscovered by the auto-variant detection process, during further computations of the Project. Variants can safely be removed after you are done adding new data to the Project.

9.3.2.6 The MIDs Definition Table

The **MIDs** Definition Table lists all the MIDs defined in the Project, with the following four characteristics (Table columns; see Figure 9–29):

- Name
- Annotation (free user-entered text)
- Sequence
- Group (the MID Group to which the MID belongs)



Name ▲	Annotation	Sequence	Group
Mid1		ACGAGTGCGT	454Standard
Mid2		ACGCTCGACA	454Standard
Mid3		AGACGCACTC	454Standard
Mid4		AGCACTGTAG	454Standard
Mid5		ATCAGACACG	454Standard
Mid6		ATATCGCGAG	454Standard
Mid7		CGTGTCTCTA	454Standard
Mid8		CTCGCGTGTC	454Standard
Mid9		TAGTATCAGC	454Standard
Mid10		TCTCTATGCG	454Standard
Mid11		TGATACGTCT	454Standard
Mid12		TACTGAGCTA	454Standard
Mid13		CATAGTAGTG	454Standard
Mid14		CGAGAGATAC	454Standard

Figure 9–29: The MIDs Definition Table sub-tab of the Project tab's right-hand panel

MIDs may be created (and assigned to MID Groups) in the MIDs Definition Table even before the sequence of the MID has been filled in by the user. Such MIDs, without defined sequences, may even be used in the definitions of the Samples encoded by Multiplexers (section 9.3.2.7.1). This flexibility allows users to define of the logical structure of an experiment in advance of knowing the specifics of the MID sequences themselves.

For the procedures used to add or remove MIDs in a Project, see section 9.3.2 (or 9.3.1, to accomplish this in a "Project Tree" view). For the procedures used to enter/edit the Name or Annotation information for an MID, see section 9.3.2. The sub-sections below provide the procedure to enter/edit the other characteristics of MIDs.

- **Criteria for good MID Sets; standard and custom MID Groups:**
  An MID Group ("454Standard") containing 14 ten-base MIDs is provided and recommended for use when designing multiplexing libraries for the AVA software. This group includes the set of 12 tags known as MID1 through MID12 that are available in kit form to make Shotgun (sstDNA) MID libraries, and includes two additional MIDs not included in these kits (MID13 and MID14). The MIDs of the 454Standard MID Group have been especially selected for the following qualities:
  - Their sequences are as divergent as possible and include only single nucleotide homopolymers, minimizing the risk that eventual sequencing errors would make any of these MIDs look like one of the other MIDs of the Group (which would cause the assignment of the read to a wrong Sample). Indeed, no fewer than six changes (insertions, deletions, and/or substitutions) separate any two MIDs of the 454Standard MID Group.)
  - All MIDs of the Group are of the same length (10-mers, in this case). The AVA software requires that all the MIDs used for a given end of the Amplicons (Primer A/Primer 1 or Primer B/Primer 2) have the same length, so they are on equal footing for error distance calculation purposes. It is permitted, however, to use a different MID length on each side of the Amplicons being handled by a Multiplexer. For instance, custom 5 bp MIDs might be used on the Primer 1 side and 10 bp MIDs might be used on the Primer 2 side.
  - None start with a "G" nucleotide, the last base of the sequencing key, to ensure clear reading of the MID tag.
  - All MIDs are read within 4 or 5 nucleotide flow cycles, to minimize usage of Run flows to read the MID tags and leave as much as possible for the sample sequence.

  The software does not constrain the user to only these 14 MIDs, however: any other set of sequences can be designed for use as multiplexing tags, incorporated between the sequencing key and the template-specific primer of the Adaptors used to prepare the Amplicon libraries, and defined as MIDs (with optional custom MID Groups) in the Amplicon Project. This flexibility can be useful, for example, if the user prefers to use shorter MIDs; or if Amplicon libraries already exist that have intrinsic sequences that can be used for demultiplexing; or if the experiment requires the multiplexing of more Samples than can be differentiated with the 14 MIDs of the 454Standard MID Group. If you design your own MIDs, it is recommended that you keep in mind the 4 criteria listed above, for best results.

  Amplicon library chemistry requires the inclusion of the MID sequence as part of the Fusion Primers used in library preparation. These must be obtained separately by the user because they contain sequences that are specific to each Amplicon, so MID Adaptors used for Amplicon libraries are not available as kit reagents. A naming convention has been adopted to distinguish between Shotgun (sstDNA) and Amplicon library MIDs; fully uppercase MID names are Shotgun (sstDNA) library MIDs, and initial uppercase Mids are MIDs used in Amplicon libraries. Thus, MID1 and Mid1 refer to the same sequence content for an MID, but are each used in the context of their respective library types; MID1 is an Adaptor available in a kit, while Mid1 is not.

Contrary to the situation with the GS *De Novo* Assembler and the GS Reference Mapper applications, the number of acceptable "reading errors" in the MIDs is not set by the user in the AVA software. Rather, the software dynamically calculates how many errors can be accepted by analyzing the set of MIDs used and determining how close they are to each other in terms of the minimum number of insertions, deletions, or substitutions that would be required to transform one MID into another.

9.3.2.6.1 To Enter or Edit the DNA Sequence of an MID

1. Double-click in the "Sequence" cell of the MID you are defining. An "Edit Sequence" window will open (Figure 9–30).
2. Paste or type the sequence (only A, T, G, or C characters; see Caution, below).
3. Click "OK".

Characters restriction: Be aware that only "nucleotide" characters (A, T, G, C) are accepted when you enter an MID Sequence into the AVA software (by typing or pasting). For convenience, when pasting sequences, characters that are not nucleotide characters and are also not IUPAC ambiguity characters (such as R for purine, Y for pyrimidine, *etc.*) are removed from the pasted entry. This is useful when pasting sequences from sources that may include non-sequence information (such as white space or numerical position information in the margin of each line). If any IUPAC ambiguity characters are included, the paste will be cancelled entirely, and an error message will be displayed explaining the problem. If you directly type individual "ambiguous" characters, however, or any character other than A, T, G, or C, these characters are simply ignored.

The restriction that no ambiguity characters be present in an MID sequence is crucial because MIDs are intended to designate specific Samples. If you have a degenerate MID design in which multiple MID sequences specify the same Sample, enter all the specific MID sequences into the system and use the Multiplexer Sample editor to specify all the MIDs that encode each Sample (see section 9.3.2.7.3)

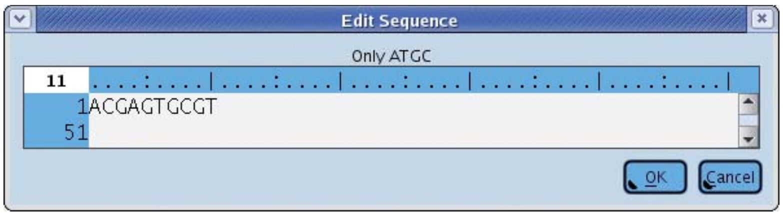


Figure 9–30: The Edit Sequence window, used to enter or edit the DNA sequence of an MID Sequence element

9.3.2.6.2 To Edit the MID Group of an MID

To transfer an MID to another pre-existing MID Group, double-click the drop down menu in the Group cell for the MID and select the MID Group you want from the available choices. You can also reassign an MID to a different MID Group by dragging the MID to an MID Group node of the MIDs Tree (a multiple selection of MIDs will assign them all to the MID Group on which you drop them). While you cannot change the name of an MID Group from within the MIDs Definition Table, you can do so in the MIDs Tree (as with any other rename operation in the tree, click once on the Group name, pause and click a second time to activate the name editor). Note that all MID Groups are distinct entities, and although you can rename an existing MID Group to match the name of another pre-existing MID Group, this will not cause the MIDs to be merged into the same group.

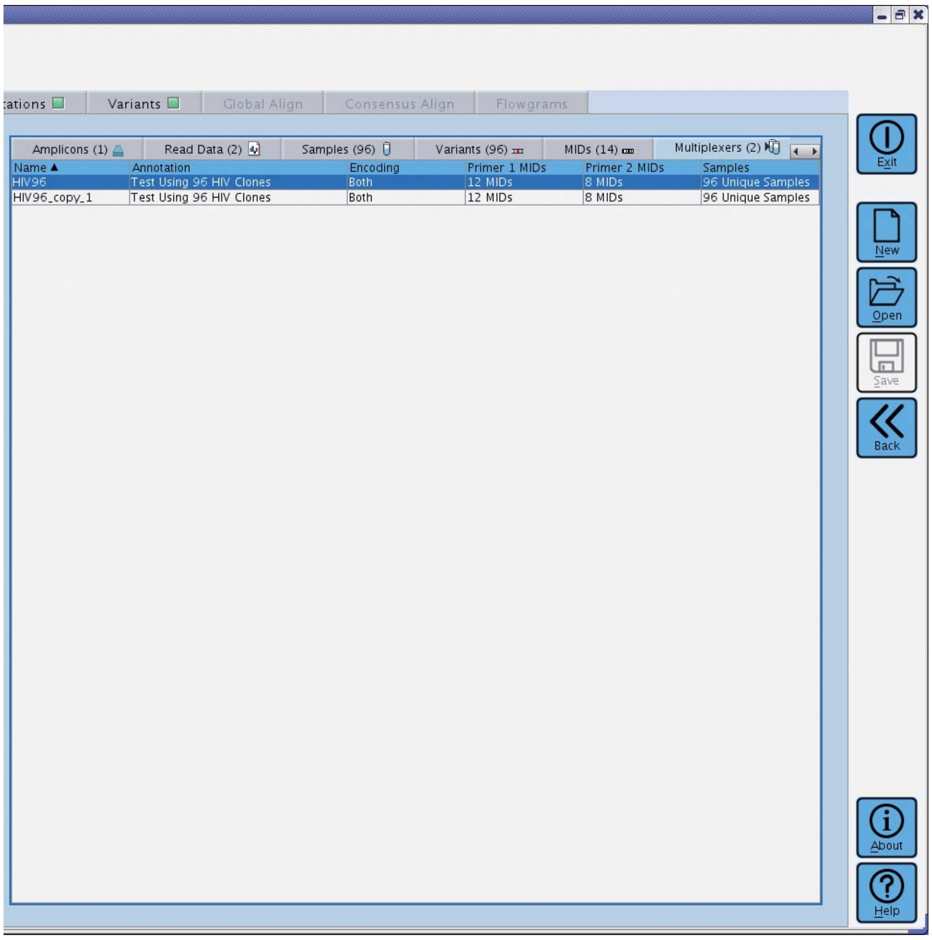
A valid MID Group should contain MIDs with sequences of the same length, and each MID sequence should be distinct. When assigning MIDs with defined sequences to an MID Group, the software will prevent you from making an inconsistent assignment, such as adding an MID with an a defined sequence to an MID Group that already has at least one defined MID sequence of a different length: if dragging to the MIDs Tree, the MID Group node will not activate to allow you to release the dragged MID; if using the drop down menu from the MIDs Definition Table, the drop down menu will only display valid MID Group choices.

Although the AVA application is designed to minimize the possibility of creating inconsistent MID Groups, to allow flexibility in editing, it allows the addition of MIDs with undefined sequences to MID Groups, and the editing of MID sequences even after they have already been assigned to an MID Group. During such editing you are allowed to bring an MID Group into a temporarily inconsistent state, with the assumption that you will eventually fix it prior to computation. If you use inconsistent MIDs when defining Multiplexers, the Multiplexer setup dialogs will provide you with error messages (section 9.3.2.7.2), and you will also be provided with error warnings prior to computation (section 9.4). Ignoring the warnings will prevent the portions of the computation that depend on the faulty Multiplexers from executing.

9.3.2.7 The Multiplexers Definition Table

The Multiplexers Definition Table lists all the Multiplexers defined in the Project, with the following six characteristics (Table columns; see Figure 9–31):

- Name
- Annotation (free user-entered text)
- Encoding
- Primer 1 MIDs
- Primer 2 MIDs
- Samples



Name ▲	Annotation	Encoding	Primer 1 MIDs	Primer 2 MIDs	Samples
HIV96	Test Using 96 HIV Clones	Both	12 MIDs	8 MIDs	96 Unique Samples
HIV96_copy_1	Test Using 96 HIV Clones	Both	12 MIDs	8 MIDs	96 Unique Samples

Figure 9–31: The Multiplexers Definition Table sub-tab of the Project tab's right-hand panel

For the procedures used to add or remove Multiplexers in a Project, see section 9.3.2 (or 9.3.1, to accomplish this in a "Project Tree" view). For the procedures used to enter/edit the Name or Annotation information for a Multiplexer, see section 9.3.2. The sub-sections below provide the procedure to enter/edit the other characteristics of Multiplexers.

Note on Sample encoding using MIDs and Multiplexers:
In the standard non-MID demultiplexing scheme, the AVA software looks for the template-specific primer sequences (Primer 1 and Primer 2) of the defined Amplicons at the beginning of each read. Once the Amplicon to which a read belongs is identified, the Sample-Amplicon associations defined for the Read Data Set that the read comes from are used to assign the read to its appropriate Sample. In other words, when MIDs are not used, the assignment of a read to an Amplicon, using the template-specific primers, is sufficient to further assign the read to the proper Sample. As explained before (see section 9.1.1.6, and Note and Caution sidebars in section 9.3.1.2), this scheme imposes the restriction that an Amplicon may only belong to a single Sample within a Read Data Set, to allow for unambiguous Sample assignment of the reads.

MIDs and Multiplexers allow this restriction to be lifted and experiments to be designed in which reads from a given Amplicon are monitored in multiple Samples, in the same Read Data Set. In this scheme, the MID sequence detected within a read is used to assign reads to Samples. This MIDs to Samples assignment is the function of Multiplexers, as described in this section.

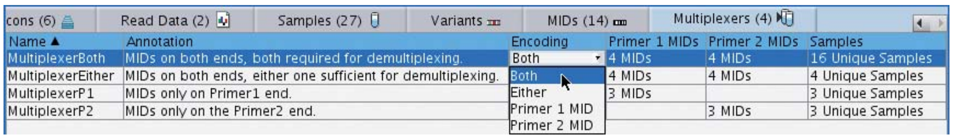
The Primer 1 and Primer 2 sequences of a read are still used, however, to determine to which Amplicon a read belongs. Moreover, different Amplicons, within a Read Data Set, may be associated with different Multiplexers. Thus, when MIDs and Multiplexers are used, the demultiplexing process involves:

1. decoding the Primer1 and Primer2 regions of the read to determine which Amplicon it represents;
2. using the user-specified association between Amplicons and Multiplexers for the Read Data Set to determine the appropriate Multiplexer to apply; and
3. using the MID sequence detected in the reads, in conjunction with that Multiplexer, to assign the read to the proper Sample.

9.3.2.7.1 To Enter or Edit the Sample Encoding using Multiplexers

The AVA software provides 4 ways to encode the Sample to which a read belongs in the Multiplexer, based on the construction of the libraries (see section 12.6 for details on Amplicon library design with MIDs). The proper option must be selected from a drop down menu in the Multiplexers Definition Table (Figure 9–32). The options, further described in the sub-sections below, are:

- Both
- Either
- Primer 1 MID
- Primer 2 MID



Name ▲	Annotation	Encoding	Primer 1 MIDs	Primer 2 MIDs	Samples
MultiplexerBoth	MIDs on both ends, both required for demultiplexing.	Both	4 MIDs	4 MIDs	16 Unique Samples
MultiplexerEither	MIDs on both ends, either one sufficient for demultiplexing.	Both	4 MIDs	4 MIDs	4 Unique Samples
MultiplexerP1	MIDs only on Primer1 end.	Either	3 MIDs		3 Unique Samples
MultiplexerP2	MIDs only on the Primer2 end.	Primer 1 MID		3 MIDs	3 Unique Samples
		Primer 2 MID			

Figure 9–32: The Encoding drop down menu, on the Multiplexers tab

Selecting the proper encoding: It is crucially important to select the encoding method that truly corresponds to the way the libraries were prepared. For example, if libraries were prepared with 'Either' chemistry in mind, it may be tempting to use a 'Primer 1 MID' or 'Primer2 MID' encoded Multiplexer since the distal MID gets discounted in favor of the proximal MID in 'Either' encoding. However, the AVA software needs to know that MIDs are expected to be found at both ends: without that knowledge, the trimmer might get a suboptimal alignment of the distal primer, which in certain cases could drop valid reads out of the analysis.

9.3.2.7.1.1 "Primer 1 MID" and "Primer 2 MID" Encoding

The "Primer 1 MID" and "Primer 2 MID" encoding options assume that the libraries were prepared with MID sequences incorporated in the design of only one of the Adaptors used in the preparation of the Amplicon libraries. The MID is placed between the sequencing key and the template-specific primer that will be identified either as Primer 1 or as Primer 2 (as entered in the Amplicon definition table).

When either of these encoding options is selected for a Multiplexer, only the corresponding Primer MID field, Primer 1 MIDs or Primer 2 MIDs, needs to be filled in the Multiplexer's Definition Table, to identify the MIDs used in the scheme (see section 9.3.2.7.2). For example, a Multiplexer encoded as "Primer 1 MID" will have an empty column in the Definition Table for the "Primer 2 MIDs" field. The maximum number of Samples that can be encoded with this scheme is equal to the number of MIDs defined in the Primer 1 MIDs or Primer 2 MIDs field.

The AVA software uses the encoding type to automatically determine where to search for MIDs within reads, taking read orientation into account. For example, if both forward and reverse reads are sequenced for an experiment where the "Primer 1 MID" encoding is being used, forward reads will have the MID at the beginning of the read, just before of the template-specific Primer 1 sequence; and reverse reads will have the reverse complement of the MID near the end of the read, just after the reverse complement of the template-specific Primer 1 sequence. For this reason, if reads will be obtained in both orientations (as is recommended) and Primer 1 MID or Primer 2 MID encoding is used, it is important to design Amplicons of a length shorter than the read length provided by the sequencing Run. If the ability to read through the Amplicons is in doubt, the "Either" encoding (section 9.3.2.7.1.3) should be used instead, to guarantee that both forward and reverse reads have an MID at their beginning.

For details on Sample assignment using the Primer 1 MID or Primer 2 MID encoding options, see section 9.3.2.7.3.1.

9.3.2.7.1.2 "Both" Encoding

"Both" encoding involves the incorporation of MID sequences in both Adaptors used in the preparation of the Amplicon libraries, such that each read contains both a Primer 1 MID and a Primer 2 MID. Therefore, both the "Primer 1 MIDs" and the "Primer 2 MIDs" fields of the Multiplexer Definition Table must have at least one MID selection (see section 9.3.2.7.2).

With this encoding scheme, both a Primer 1 MID and a Primer 2 MID must be observed in a read to assign it to the proper Sample. Note that there is no requirement that the same set of MIDs be used at both ends of the Amplicons: the two MIDs used to determine Sample assignment are completely independent from one another. In addition, with the "Both" encoding, the order of the appearance of MIDs is significant: for example, "Primer 1 Mid1 – Primer 2 Mid6" is a different encoding than "Primer 1 Mid6 – Primer 2 Mid1". Given this combinatorial feature, the maximum number of Samples that can be encoded with this scheme is equal to the product of the number of MIDs defined in the Primer 1 MIDs and the Primer 2 MIDs fields.

It is also important to remember that in order to be able to read the distal MID, the length of the Amplicon library product must be within the read length provided of the sequencing Run script. For details on Sample assignment using the Both encoding option, see section 9.3.2.7.3.2.

9.3.2.7.1.3 "Either" Encoding

The "Either" encoding method is a hybrid between the single primer MID and "Both" methods. The libraries are tagged with MIDs on both the Primer 1 and Primer 2 ends, but only one MID needs to be observed on a read to assign it to the proper Sample. This can be useful if the Amplicon library products are longer than the read length of the sequencing Run, and you are sequencing them from both ends. One limitation of this encoding scheme is that a given MID can be used for only one Sample, for each of the Primer 1 and Primer 2 ends.

As with the Both encoding scheme, there is no requirement that the same set of MIDs be used at both ends of the Amplicons. Although the number of MIDs used at the Primer 1 and Primer 2 ends will typically be the same, the software even allows degenerate designs in which the "Either" encoding is used and the number of Primer 1 MIDs and Primer 2 MIDs differ. Regardless, a read must be able to be uniquely assigned to the same Sample whether its Primer 1 MID or the Primer 2 MID are used for demultiplexing. Therefore, the maximum number of Samples that can be encoded with a Multiplexer that uses this scheme is equal to the smaller of the number of MIDs defined in the Primer 1 MIDs and Primer 2 MIDs field.

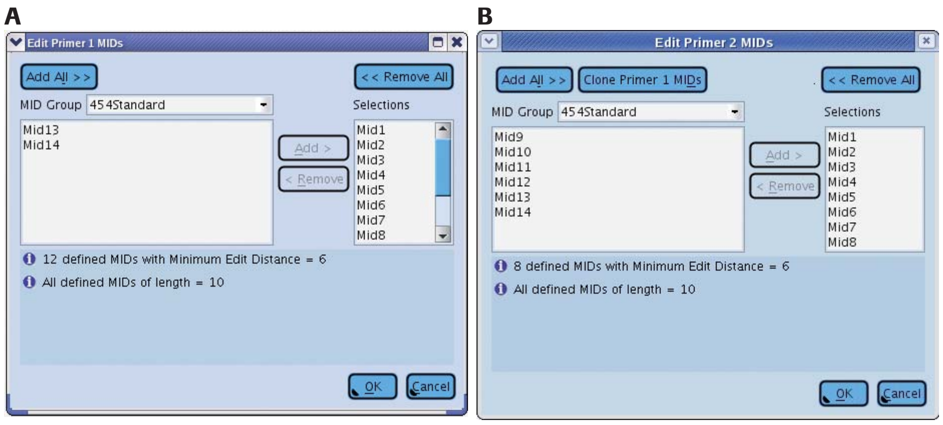
For details on Sample assignment using the Either encoding option, see section 9.3.2.7.3.3.

9.3.2.7.2 To Enter or Edit the Primer 1 MIDs and Primer 2 MIDs

The user must specify the list of MIDs that the AVA software must search for to demultiplex reads using a Multiplexer. This information is set in the Primer 1 MIDs and the Primer 2 MIDs columns of the Multiplexer Definition Table. If Primer 1 MID or Primer 2 MID encoding is chosen, only the corresponding 'Primer MIDs' cell is available for that Multiplexer; if Either or Both encoding is chosen, both cells are available and must be filled.

To specify the MIDs for one end of a Multiplexer, double-click on the appropriate 'Primer MIDs' cell for that Multiplexer. The Edit Primer 1 MIDs (or Edit Primer 2 MIDs) window opens (Figure 9–33). The window will not open unless at least one MID entry has already been specified into the MID Definition Table (though the MIDs do not have to have sequences defined at this stage). Select the MIDs of interest on the list on the left, and click Add. To remove MIDs that have been previously selected, highlight them in the list on the right, and click Remove .

Certain shortcuts are available to carry out these tasks, such as an Add All and a Remove All button; and an MID Group drop down menu allows the user to restrict the list on the left to the MIDs contained in any of the MID Groups available in the Project. Also, if the Primer 1 and Primer 2 MIDs are the same, you can first define the Primer 1 MIDs and then use the Clone Primer 1 MIDs button in the Edit Primer 2 MIDs window (Figure 9–33**B**), to create the same list for that set.

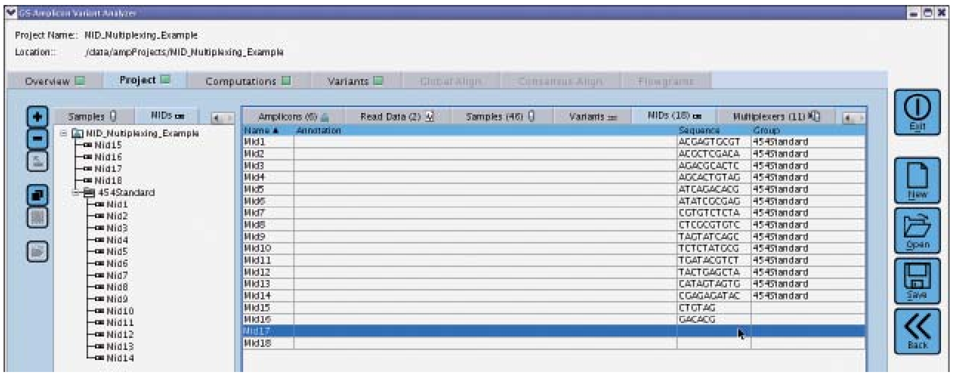


A:
MID Group 454Standard
Mid13
Mid14
Selections: Mid1, Mid2, Mid3, Mid4, Mid5, Mid6, Mid7, Mid8
12 defined MIDs with Minimum Edit Distance = 6
All defined MIDs of length = 10

B:
MID Group 454Standard
Mid9
Mid10
Mid11
Mid12
Mid13
Mid14
Selections: Mid1, Mid2, Mid3, Mid4, Mid5, Mid6, Mid7, Mid8
8 defined MIDs with Minimum Edit Distance = 6
All defined MIDs of length = 10

Figure 9–33: (A) The Edit Primer 1 MIDs window and (B) the Edit Primer 1 MIDs window. Note the Clone Primer 1 MIDs button in the Edit Primer 2 MIDs window.

The MID Group drop down menu also contains some "virtual" groups that are automatically generated by the AVA software based on the MIDs currently defined in the Project. Figure 9–34 shows an example where all 14 of the 454Standard MIDs (10-mers) have already been loaded into the Project, and four new MIDs have been added without groups: two 6-base MIDs (Mid15 and Mid16) and two MIDs for which no sequence has yet been defined (Mid17 and Mid18).



Project Name:: MID_Multiplexing_Example
Location:: /data/ampProjects/MID_Multiplexing_Example

Name ▲	Annotation	Sequence	Group
Mid1		ACGAGTGCGT	454Standard
Mid2		ACGCTCGACA	454Standard
Mid3		AGACGCACTC	454Standard
Mid4		AGCACTGTAG	454Standard
Mid5		ATCAGACACG	454Standard
Mid6		ATATCGCGAG	454Standard
Mid7		CGTGTCTCTA	454Standard
Mid8		CTCGCGTGTC	454Standard
Mid9		TAGTATCAGC	454Standard
Mid10		TCTCTATGCG	454Standard
Mid11		TGATACGTCT	454Standard
Mid12		TACTGAGCTA	454Standard
Mid13		CATAGTAGTG	454Standard
Mid14		CGAGAGATAC	454Standard
Mid15		CTGTAG	
Mid16		GACACG	
Mid17			
Mid18			

Figure 9–34: MID Tree and Definition Table view showing the 454Standard group MIDs, plus four newly defined MIDs (Mid15 - Mid18), two 6-mers and two that have no defined sequence.

Figure 9–35**A**, then shows the MID Group drop down menu for the MIDs in Figure 9–34:

- The AVA software always provides an "[All MIDs]" option on this menu to allow all the MIDs defined in the Project to be viewed in the left list, irrespective of their group status (and even if an MID in the Project has not yet been assigned a sequence).
- In addition, the software creates virtual MID Groups based on the length of the MIDs defined in the Project. This is useful because, as mentioned above (see Note in section 9.3.2.6), all the MIDs used on a given end of an Amplicon must be of the same length.
  - Note that MIDs without a defined sequence will appear in all length-restricted lists (*e.g.* see Figure 9–35**B**). This allows undefined MIDs to be selected in a Multiplexer scheme and defined later. Once an MID has a sequence defined, it will lose its wild card status and will only appear in the list appropriate to its length.

A **B**

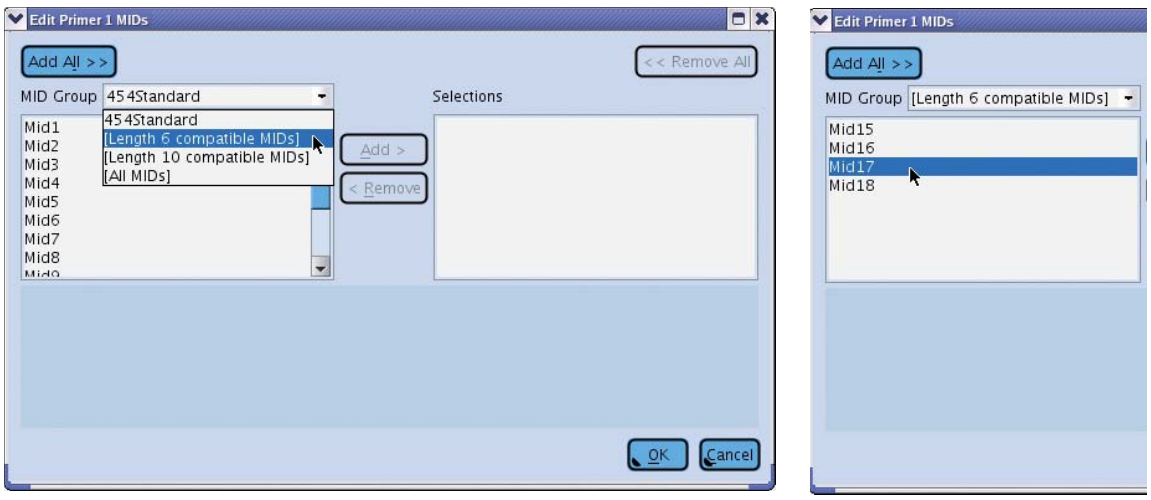
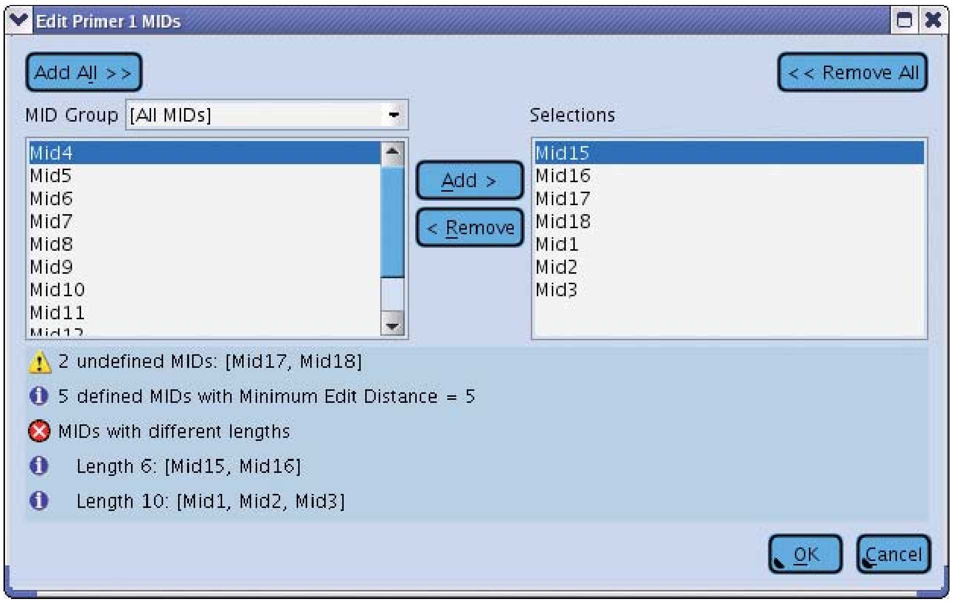


Figure 9–35: (A) The Edit MIDs window showing the MID Group drop down menu. A defined group "454Standard" is listed along with three custom automatically generated groups, the "[All MIDs]" group and two groups based on MID length. (B) The "[Length 6 compatible MIDs]" group restricts the left list to those MIDs that are exactly 6 bases long, along with those that have no sequence yet defined.

The lower part of the Edit Primer 1 MIDs (or Edit Primer 2 MIDs) window provides information (and errors or warnings, as appropriate) concerning the MIDs selected. For example, this includes a summary of the number of MIDs selected, their length, and the minimum edit distance (the minimum number of insertion, deletion, or substitution sequencing errors that could turn one of the selected MID sequences into one of the others) (see Figure 9–33, above).

The types of errors and warnings provided may include MIDs not all the same length, or undefined MIDs (Figure 9–36). Note that the software gives the benefit of the doubt to undefined MIDs, and calls the attention of the user with a warning but does not assume an error. This provides the advantage that the structure of a Multiplexer can be defined independently, and possibly in advance, of the knowledge of the MID sequences themselves. However, prior to computation, all the MIDs used in defining Multiplexers that are associated with active Read Data Set must, naturally, be defined. The software also calculates the minimum edit distance even for defined MIDs of different lengths, assuming that corrections will be made prior to Project computation (*i.e.* that MIDs of unequal length will be corrected or eliminated).



2 undefined MIDs: [Mid17, Mid18]
5 defined MIDs with Minimum Edit Distance = 5
MIDs with different lengths
Length 6: [Mid15, Mid16]
Length 10: [Mid1, Mid2, Mid3]

Figure 9–36: Examples of errors and warnings flagged on the Edit Primer 1 MIDs window.

9.3.2.7.3 To Enter or Edit the Samples Assignment

The most complex part of setting up a Multiplexer is to specify the assignment of the MIDs to the Samples. This is done in the Edit Samples window, which is accessed by double-clicking in the cell of the Samples column, for the Multiplexer of interest (in the Multiplexer Definition Table). This window can take 3 different forms, depending on the type of encoding selected, as described in the sections below. Note that Primer 1 MIDs and/or Primer 2 MIDs, as appropriate, must have been selected for that Multiplexer for the Edit Samples window to be available; if any errors or warnings exist regarding the selected MIDs (see section 9.3.2.7.2), these will be displayed in the Edit Samples window as well.

9.3.2.7.3.1 Sample Assignment with "Primer 1 MID" or "Primer 2 MID" Encoding

With these single-end MID encoding schemes, the Edit Samples window simply lists all the MIDs selected for the Multiplexer (see section 9.3.2.7.2), and the Sample is selected from the drop down menu (or can be typed) in the cell to the right of each MID name (Figure 9–37**A**). The user can also type into the cells the names of Samples that have not yet been defined in the project: new samples with those names will automatically be created and appear in the Project's Samples Definition Table when the user clicks OK. These samples will not be created, however, if the user clicks Cancel. A Sample assignment may be removed from a given MID by choosing the "--remove--" option from the drop down menu.

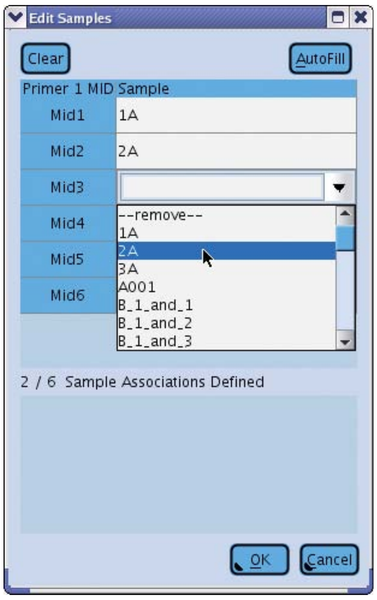
If the drop down menus are too narrow to display the full Sample names, you can widen the Edit Samples window, making more room for the 'Sample' column.

Certain shortcuts are available on this window as well: clicking the Autofill button assigns default-named Samples to any MID that does not yet have an assigned Sample (Figure 9–37**B**); and a Clear button empties all the Sample cells. The default Sample names contain three parts, in the following format:

Sample_<Multiplexer name>_<MID name>

As with typed-in novel Sample names, the Autofill Samples will only be added to the Project if the user clicks the OK button.

A

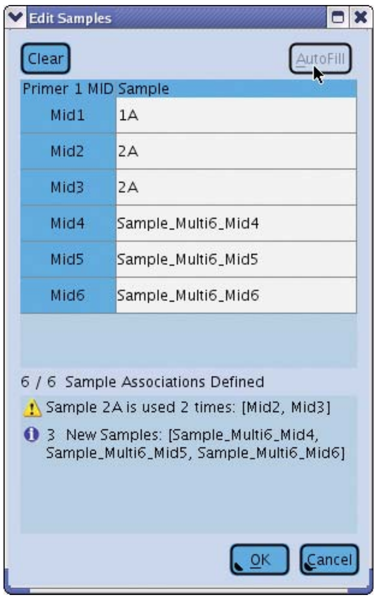


Primer 1 MID	Sample
Mid1	1A
Mid2	2A
Mid3	
Mid4	
Mid5	
Mid6	

```
--remove--
1A
2A
3A
A001
B_1_and_1
B_1_and_2
B_1_and_3
```

2 / 6 Sample Associations Defined

B



Primer 1 MID	Sample
Mid1	1A
Mid2	2A
Mid3	2A
Mid4	Sample_Multi6_Mid4
Mid5	Sample_Multi6_Mid5
Mid6	Sample_Multi6_Mid6

6 / 6 Sample Associations Defined

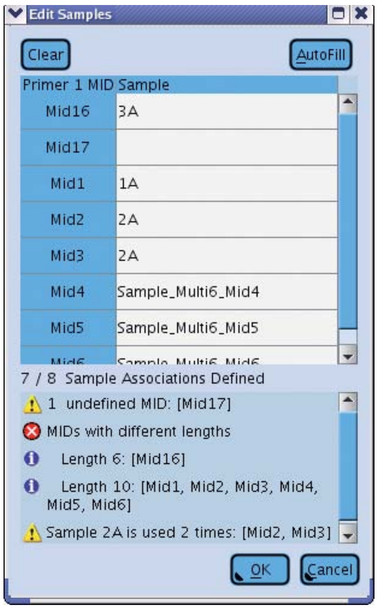
Sample 2A is used 2 times: [Mid2, Mid3]

3 New Samples: [Sample_Multi6_Mid4, Sample_Multi6_Mid5, Sample_Multi6_Mid6]

Figure 9–37: The Edit Samples window, for Primer 1 MIDs encoding.

A functional Multiplexer must specify at least one Sample assignment, but it is not formally necessary to fill all cells of the Table. This can be useful if a subset of the selected MIDs have not been used in the experiment, but were known to have been used in a previous experiment: specifying them allows the system to search for these MIDs as potential contaminants and prevent them from being misinterpreted as an erroneous version of one of the MIDs actually used in the Project to demultiplex Samples. On the other hand, it is possible to assign the same Sample to multiple MIDs (but not the opposite). Since this could be a legitimate experimental set up, it does not elicit an error message; however, a warning is displayed to draw the user's attention to this unusual assignment (Figure 9–37**B**).

In a manner analogous to the Edit Primer 1 MIDs or Edit Primer 2 MIDs windows (section 9.3.2.7.2), a summary of the assignment scheme is provided at the bottom of the Edit Samples window, including information on the number of MID-Sample associations defined and the total number that can be defined with the MIDs selected (Figure 9–37). Any errors and warnings associated with the MIDs are also shown here, to alert the user that action must be taken to complete or correct the MID definitions or the Sample assignments (Figure 9–38).



Primer 1 MID	Sample
Mid16	3A
Mid17	
Mid1	1A
Mid2	2A
Mid3	2A
Mid4	Sample_Multi6_Mid4
Mid5	Sample_Multi6_Mid5

7 / 8 Sample Associations Defined

⚠ 1 undefined MID: [Mid17]

✖ MIDs with different lengths

ℹ Length 6: [Mid16]

ℹ Length 10: [Mid1, Mid2, Mid3, Mid4, Mid5, Mid6]

⚠ Sample 2A is used 2 times: [Mid2, Mid3]

Figure 9–38: The Edit Samples window for Primer 1 MID encoding, showing one error and two warnings regarding either the MIDs currently selected for this Multiplexer, or the Sample assignments. The MIDs are listed in sorted order based on their MID Group (not displayed, but visible in the tooltip that will appear if the user mouses over the MID name) and then their MID name. MIDs without an associated MID Group appear before those with an MID Group. In this example, Mid16 and Mid17 were not assigned to an MID Group and so appear before Mid1. Had they been part of the same MID group as Mid1 they would have appeared later in the list, as expected.

9.3.2.7.3.2 Sample Assignment with "Both" Encoding

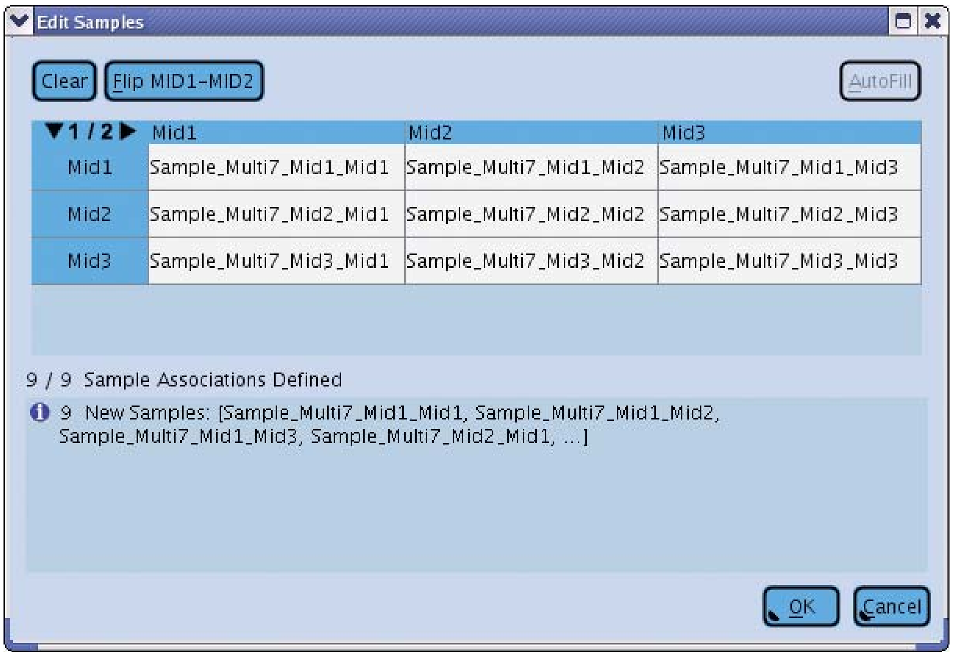
With the "Both" encoding scheme, two MIDs must be specified for each Sample, one attached to Primer 1 and one to Primer 2. In this case, therefore, the Edit Samples window displays a two-dimensional Table where the MIDs selected for the Primer 1 side occupy one dimension (rows or columns) and those for Primer 2 occupy the other (Figure 9–39). In a manner analogous to the single-MID Sample encoding seen above, Sample assignment is done by selecting the Sample name from the drop down menu (or by typing it) in the cell at the intersection of the two encoding MID names.

If the drop down menus are too narrow to display the full Sample names, you can widen the Edit Samples window, making more room for the 'Sample' columns. You can also resize individual columns by dragging on the column header separators to the left or right, until the column of interest has the proper width to allow the display of the full Sample names.

The other features of this window (can have empty cells, shortcut buttons, summary and error/warning reporting, *etc.*) are the same as for the Primer 1 MID or Primer 2 MID encoding, described above. For "Both" encoding, the names of the Autofill Samples are of the form:

Sample_<Multiplexer name>_<Primer 1 MID name>_<Primer 2 MID name>

It is important to be aware of the directionality of the Amplicons when assigning the Samples to MID pairs: MIDs are selected separately for the Primer 1 and Primer 2 sides to support this directionality. In the Edit Samples window, the side corresponding to the two selected MID sets are identified by a “1” and a “2” with arrowheads, on the top-left corner of the Table. This is illustrated on Figure 9–39: in this example, the table of the Edit Samples window has been populated using the Autofill button; the Primer 1 – MID1: Primer 2 – MID2 pair encodes Sample Sample_Multi7_Mid1_Mid2, while the Primer 1 – MID2: Primer 2 – MID1 pair encodes Sample Sample_Multi7_Mid2_Mid1. For convenience, a Flip Table button can transpose the table.



▼1 / 2▶	Mid1	Mid2	Mid3
Mid1	Sample_Multi7_Mid1_Mid1	Sample_Multi7_Mid1_Mid2	Sample_Multi7_Mid1_Mid3
Mid2	Sample_Multi7_Mid2_Mid1	Sample_Multi7_Mid2_Mid2	Sample_Multi7_Mid2_Mid3
Mid3	Sample_Multi7_Mid3_Mid1	Sample_Multi7_Mid3_Mid2	Sample_Multi7_Mid3_Mid3

9 / 9 Sample Associations Defined

9 New Samples: [Sample_Multi7_Mid1_Mid1, Sample_Multi7_Mid1_Mid2, Sample_Multi7_Mid1_Mid3, Sample_Multi7_Mid2_Mid1, ...]

Figure 9–39: The Edit Samples window for the “Both” encoding scheme, showing Samples that were assigned using the Autofill button.

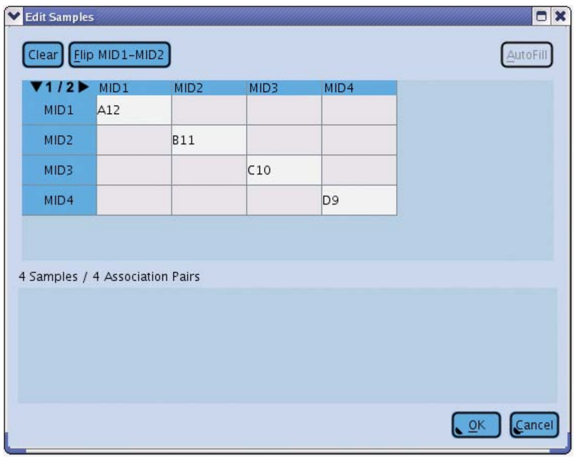
9.3.2.7.3.3 Sample Assignment with “Either” Encoding

With the “Either” encoding, Amplicons also have two MIDs so the Edit Samples Table is two-dimensional as well. However, contrary to the situation with Both encoding, the MID from only one end is used to assign any given read to the proper Sample. This situation is akin to the Primer 1 MID and Primer 2 MID encoding cases, and likewise, each MID (at a given end) can encode only one Sample.

To help in this, the software grays out cells that become ineligible as Samples are assigned to Primer 1 MID – Primer 2 MID pairs. In the simplest case, the libraries are designed such that the same MIDs are placed at both ends of each Amplicon (Figure 9–40**A**). For an Either encoded Multiplexer, the Autofill function is only enabled for this type of symmetric design (AutoFill expects to make sample assignments along the diagonal where the same MID is used on each end of the read: Mid1-Mid1, Mid2-Mid2, *etc.*).

However, asymmetric designs are also legitimate. The software flags this with a warning in case the asymmetry was unintended (Figure 9–40**B**). Even if the same set of MIDs are selected for both the Primer 1 MIDs and the Primer 2 MIDs series (a symmetrical design), the Sample assignment does not have to be along the diagonal in the grid (Mid1-Mid1, Mid2-Mid2, *etc.*) as it would be with an AutoFill. As long as no MID at either end is assigned to more than one Sample, and every MID on one side that has a Sample assignment has some corresponding MID on the other side with the same Sample assignment, the design is still valid. Again, mis-assignment is prevented by graying out the ineligible cells (Figure 9–40**C**).

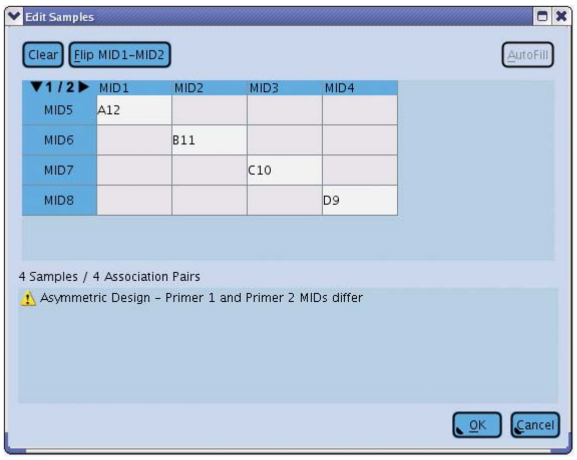
A



▼1 / 2▶	MID1	MID2	MID3	MID4
MID1	A12			
MID2		B11		
MID3			C10	
MID4				D9

4 Samples / 4 Association Pairs

B

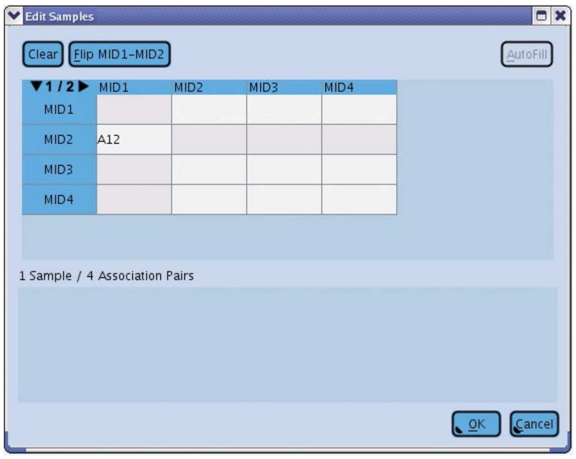


▼1 / 2▶	MID1	MID2	MID3	MID4
MID5	A12			
MID6		B11		
MID7			C10	
MID8				D9

4 Samples / 4 Association Pairs

⚠ Asymmetric Design - Primer 1 and Primer 2 MIDs differ

C



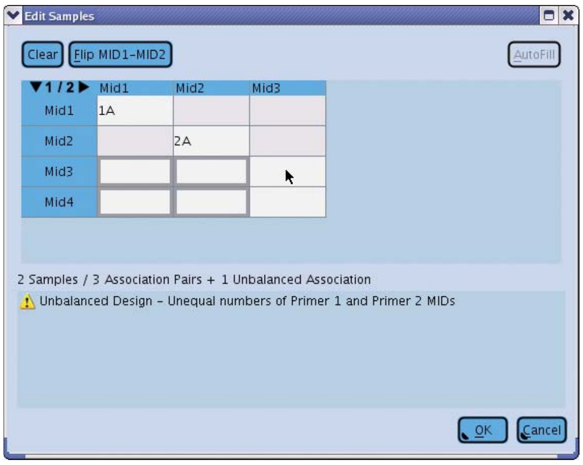
▼1 / 2▶	MID1	MID2	MID3	MID4
MID1				
MID2	A12			
MID3				
MID4				

1 Sample / 4 Association Pairs

Figure 9–40: The Edit Samples window, for Either encoding. (A) Symmetrical design with Sample assignment down the diagonal. (B) Asymmetrical design with different sets of Primer1 MIDs and Primer 2 MIDs and a warning about the asymmetry. (C) Symmetrical design but with a Sample assignment deviating from the diagonal.

In fact, it is even possible to have a different number of MIDs selected on the Primer 1 and Primer 2 sides. When this kind of design is used, the software displays a warning that there are unequal numbers of Primer 1 and Primer 2 MIDs, and specifies the number of unbalanced associations (Figure 9–41). In this special case, one or more MIDs will have to be used more than once, yet the constraint that a given MID at a given end of the Amplicons must specify a single Sample (to allow for unambiguous assignment of the reads) must be respected. To accomplish this, the AVA software restricts the Sample choices in cells that may receive such secondary assignments (highlighted with a thicker gray border), to Samples already specified for a Primer 1 MID or a Primer 2 MID. Some of the specific circumstances one might encounter are illustrated in Figure 9–41.

A

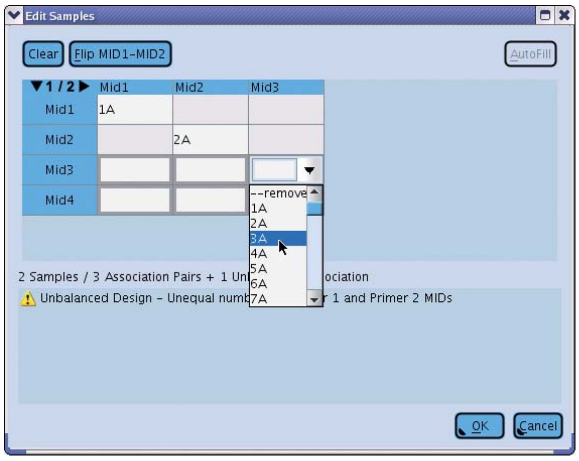


▼1 / 2▶	Mid1	Mid2	Mid3
Mid1	1A		
Mid2		2A	
Mid3			
Mid4			

2 Samples / 3 Association Pairs + 1 Unbalanced Association

⚠ Unbalanced Design - Unequal numbers of Primer 1 and Primer 2 MIDs

B

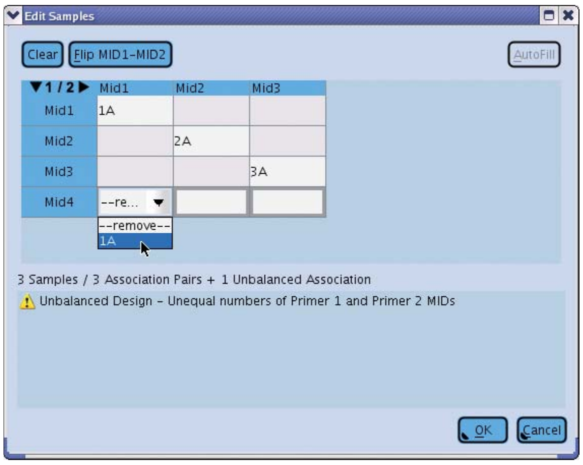


▼1 / 2▶	Mid1	Mid2	Mid3
Mid1	1A		
Mid2		2A	
Mid3			
Mid4			

2 Samples / 3 Association Pairs + 1 Un ociation

⚠ Unbalanced Design - Unequal numb r 1 and Primer 2 MIDs

C

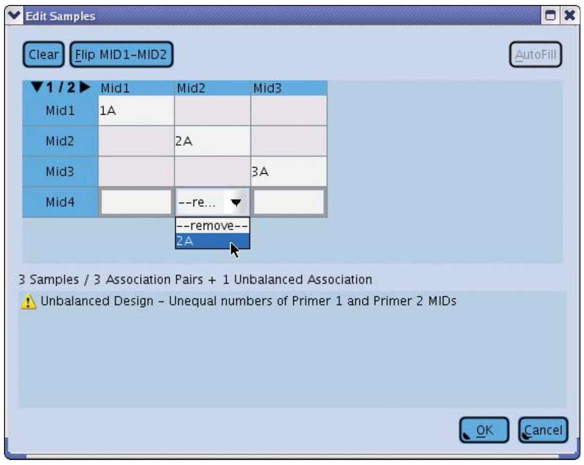


▼1 / 2▶	Mid1	Mid2	Mid3
Mid1	1A		
Mid2		2A	
Mid3			3A
Mid4	--re...		

3 Samples / 3 Association Pairs + 1 Unbalanced Association

⚠ Unbalanced Design - Unequal numbers of Primer 1 and Primer 2 MIDs

D



▼1 / 2▶	Mid1	Mid2	Mid3
Mid1	1A		
Mid2		2A	
Mid3			3A
Mid4		--re...	

3 Samples / 3 Association Pairs + 1 Unbalanced Association

⚠ Unbalanced Design - Unequal numbers of Primer 1 and Primer 2 MIDs

Figure 9–41: Edit Samples window for an Either-encoded Multiplexer with an unbalanced design. There are 4 Primer 1 MIDs and 3 Primer 2 MIDs. (A) With two sample assignments already made, the only cells that are not constrained at all (Mid3-Mid3 and Mid4-Mid3) are shown as totally white. Cells that have a thicker gray border are available for selection but are constrained to a single choice by previous Sample assignments. (B) The Mid3-Mid3 cell has an unconstrained Sample list and Sample 3A is being selected. (C) After the Sample assignment, the Mid4 row consists of three constrained cells with thick gray borders. The Mid4-Mid1 cell only allows Sample 1A to be selected because the Primer 2 MID Mid1 is already being used to encode that Sample. (D) Similarly, The Mid4-Mid2 cell only allows Sample 2A to be selected because the Primer2 MID Mid2 is already being used to encode Sample 2A.

Again the other features of the Edit Samples window for "Either" encoding (can have empty cells, shortcut buttons, summary and error/warning reporting, *etc.*), are the same as for the Primer 1 MID, Primer 2 MID, or Both encoding, described above. Autofill Samples are handled the same way as for Both encoding.

As above, if the drop down menus are too narrow to display the full Sample names, you can widen the Edit Samples window, making more room for the 'Sample' columns. You can also resize individual columns by dragging on the column header separators to the left or right, until the column of interest has the proper width to allow the display of the full Sample names.

9.3.2.7.4 Using Multiplexers for more than one Read Data

Once a Multiplexer has been created and defined, it can be used on any number of Read Data Sets in the Project, as long as these Read Data Sets share the same MIDs, encoding, and MID-to-Sample associations. Furthermore, each Multiplexer-Read Data Set pairing can even be used to demultiplex distinct sets of Amplicons, as defined by the Sample-Amplicon-Read Data Set triad associations (Figure 9–42).

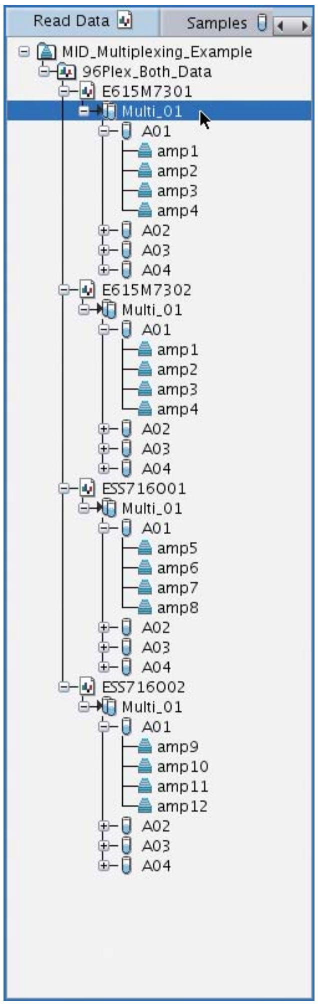


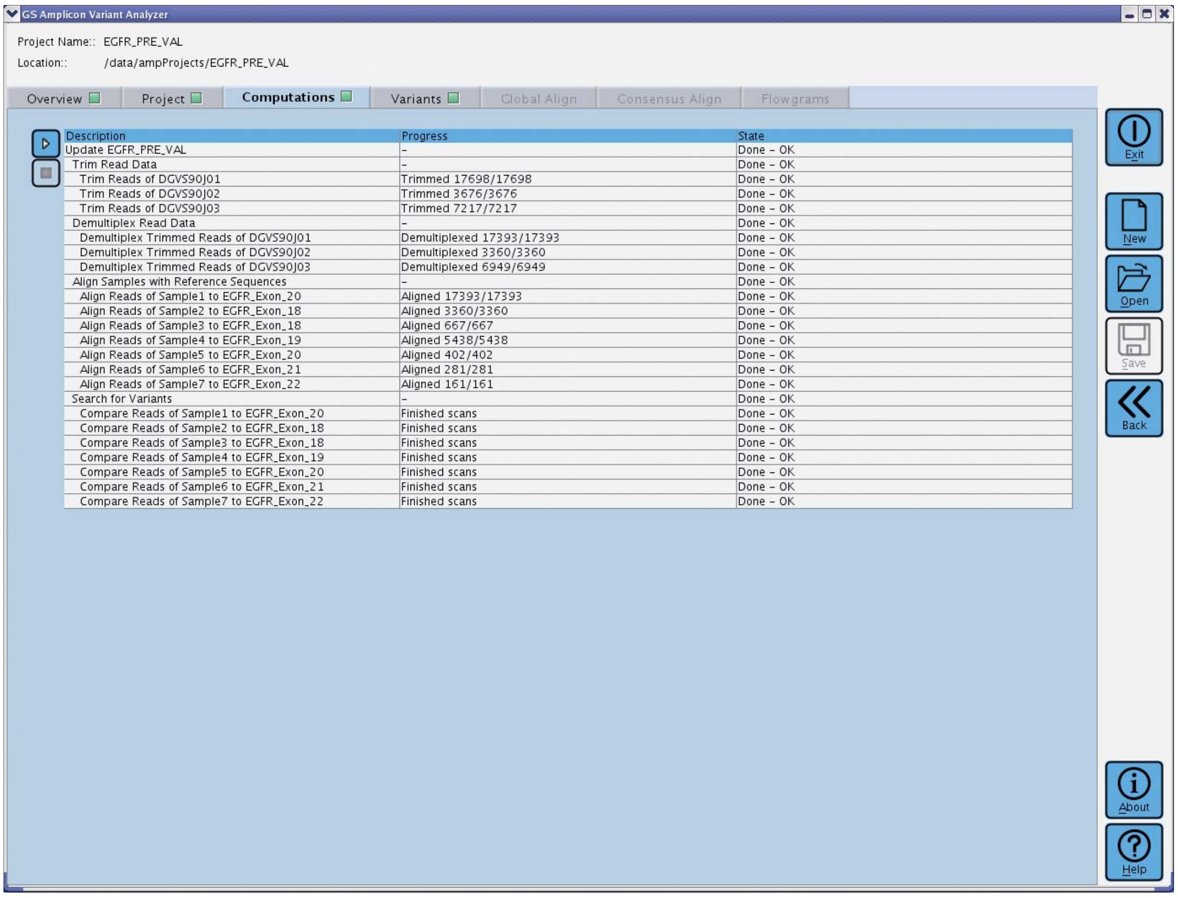
Figure 9–42: A single Multiplexer (Multi_01) is associated with 4 different Read Data Sets in the Read Data Tree. In the context of the first two Read Data Sets, the same set of Amplicons is being measured (amp1-amp4), but different Amplicons are being measured for each of the remaining Read Data Sets (amp5-amp8 for the third Read Data Set and amp9-amp12 for the fourth Read Data Set).

The AVA software also allows for Multiplexers to be duplicated. Although it is not necessary to define multiple exact copies of Multiplexers within a Project, as just discussed, duplication may be useful if multiple Multiplexers need to be defined, that share common baseline features such as the encoding scheme and specific MIDs on each side of their Amplicons. This is done using the "Duplicate item" button on the left margin of the Project tab (see section 9.3.2): a copy of a Multiplexer created this way retains the encoding and MID settings of the original.

The "Select Amplicons associated with item" button can also provide a very useful shortcut when a given set of Amplicons is to be measured by multiple Read Data Set – Multiplexer pairs. This button is also located on the left margin of the Project tab, and its functionality is described in section 9.3.1. Selecting a large number of disparate Amplicons from the Amplicons Definition Table, to associate them to a Multiplexer, can be laborious and painstaking; if many Multiplexers require the same (or similar) Amplicon associations, you need to create only the first of these Multiplexers manually, then select it in the Read Data Tree and click the "Select Amplicons associated with item" button; the software will switch to the Amplicons Definition Table sub-tab, and the subset of the Amplicons that are associated with the original Multiplexer will be selected, ready to be dragged to another Multiplexer in the Tree.

9.4 The Computations Tab

The Computations tab has only one function: carry out the computations on the Amplicon Project, with the elements currently defined and the "active" Read Data Set(s). This requires that the Project has been "set up", including the definition of the various elements that constitute it (Reference Sequences, Amplicons, Read Data Sets/Read Groups, Samples, Variants, and, optionally, MIDs / MID Groups and Multiplexers), and their associations as appropriate. (See sections 9.3 on the Project tab, and the example in section 10.2, for details on how to set up a Project before computation.)



Project Name:: EGFR_PRE_VAL
Location:: /data/ampProjects/EGFR_PRE_VAL

Description	Progress	State
Update EGFR_PRE_VAL	–	Done – OK
Trim Read Data	–	Done – OK
Trim Reads of DGVS90J01	Trimmed 17698/17698	Done – OK
Trim Reads of DGVS90J02	Trimmed 3676/3676	Done – OK
Trim Reads of DGVS90J03	Trimmed 7217/7217	Done – OK
Demultiplex Read Data	–	Done – OK
Demultiplex Trimmed Reads of DGVS90J01	Demultiplexed 17393/17393	Done – OK
Demultiplex Trimmed Reads of DGVS90J02	Demultiplexed 3360/3360	Done – OK
Demultiplex Trimmed Reads of DGVS90J03	Demultiplexed 6949/6949	Done – OK
Align Samples with Reference Sequences	–	Done – OK
Align Reads of Sample1 to EGFR_Exon_20	Aligned 17393/17393	Done – OK
Align Reads of Sample2 to EGFR_Exon_18	Aligned 3360/3360	Done – OK
Align Reads of Sample3 to EGFR_Exon_18	Aligned 667/667	Done – OK
Align Reads of Sample4 to EGFR_Exon_19	Aligned 5438/5438	Done – OK
Align Reads of Sample5 to EGFR_Exon_20	Aligned 402/402	Done – OK
Align Reads of Sample6 to EGFR_Exon_21	Aligned 281/281	Done – OK
Align Reads of Sample7 to EGFR_Exon_22	Aligned 161/161	Done – OK
Search for Variants	–	Done – OK
Compare Reads of Sample1 to EGFR_Exon_20	Finished scans	Done – OK
Compare Reads of Sample2 to EGFR_Exon_18	Finished scans	Done – OK
Compare Reads of Sample3 to EGFR_Exon_18	Finished scans	Done – OK
Compare Reads of Sample4 to EGFR_Exon_19	Finished scans	Done – OK
Compare Reads of Sample5 to EGFR_Exon_20	Finished scans	Done – OK
Compare Reads of Sample6 to EGFR_Exon_21	Finished scans	Done – OK
Compare Reads of Sample7 to EGFR_Exon_22	Finished scans	Done – OK

Figure 9–43: The Computations tab

To carry out the computations, simply click the "Start Computation" button. The Computations Status Table will be populated as the computations progress. The Table comprises 3 columns: Description (of each computational step); Progress; and Status. You cannot enter any information into this Table.

The main steps of a Project's computations are as follows:

1. Trim Read Data: for each Read Data Set, the Primer sequences and MIDs, if used, of all the reads are identified for demultiplexing purposes, and the trim points are noted delineating the "Target" sequences. (As mentioned in section 9.1.1.3, trimming the Primers is important because any variations found therein would have no biological significance, and therefore should not be reported by the AVA software.)

2. Demultiplex Read Data: for each Read Data Set, each read is identified as belonging to one defined Amplicon and assigned to the appropriate Sample, taking into account any relevant MID information. If MIDs are not used, the Amplicon must be associated with one specific Sample for the Read Data Set, and the read's Sample is so established. If MIDs are used, the Amplicon is used to determine the relevant Multiplexer associated with the Read Data Set, and then the MIDs found within the read, in conjunction with the MID encoding of Samples defined by the Multiplexer, are used to determine the read's Sample. Demultiplexing may involve:

   a. splitting the reads of a Read Data Set over multiple Samples, *e.g.* if the experiment was set up such that one or more Amplicons (which are associated with different Samples either directly, or through the use of MIDs) were present in a PicoTiterPlate device region of a sequencing Run; and/or

   b. joining of the reads from multiple Read Data Sets into any given Sample, *e.g.* if the experiment was set up such that multiple regions of a PicoTiterPlate device and/or sequencing Runs contributed reads to the Amplicons that are associated with the Sample.

3. Align Samples with Reference Sequences: the reads of each Sample are multiply aligned to the Reference Sequences corresponding to the Amplicons with which the Samples are associated. Initially, the reads are aligned with their primers included, but after the reads find their place in the alignment, the primer regions get trimmed off. Using the primers in this way can provide more alignment context and produce more sensitive and accurate alignments at the edges of amplicons (particularly amplicons where there is a sizeable deletion near the target sequence boundaries). Similar Individual reads are grouped into Consensus reads, and the multiple alignments of Individual and Consensus reads are constructed for later viewing in the Global Align and the Consensus Align tabs.

4. Search for Variants: For each defined Variant, the multiple alignments of the previous step are scanned to determine which Individual and Consensus reads span all the positions of the Variant's Pattern and, of those that do span all these positions, which reads satisfy all the constraints specified by the Pattern. Statistics are calculated for forward and reverse reads separately, and then pooled together, in order to provide estimates of Variant frequencies. The results of this search are reported in the Variants tab.

   In addition, the AVA software automatically searches for potential Variants not explicitly entered into the system. These Auto-Detected "Putative" Variants receive "Intelligent Names" that are unique and compact, yet descriptive (see section 12.2); these Auto-Detected Variants are not automatically loaded into the Project, but can be manually loaded based on selection criteria on the main Variants tab (see section 9.5.2.6). The current version of the AVA software automatically searches for substitution (SNP) and block deletion Variants. (The software does not automatically detect Insertion Variants at this time, so manual browsing of the alignments may be needed if insertions are anticipated.) Combined with the ability to selectively load and subsequently sort and filter these Variants based on their frequencies and read-orientation support, the vast majority of interesting Variants can be easily discovered and evaluated from the view of the data provided in the Variant tab's Sample – Variants Table. By providing the ability to both edit the Variant Status, and filter by that Status, the AVA software provides a simple Discovery Workflow to determine which Variants have been evaluated and what the outcome of that evaluation was. See section 9.5.2.7 for more details on the proposed Variant Discovery Workflow process.

If you modified some information in the Project after computing it (*e.g.* imported new Read Data Sets, defined new Variants, *etc.*), you must re-compute it to update the results reported in all the tabs. When you re-compute a Project, the AVA software uses cached results, if possible, for any step that has not changed (except for demultiplexing, which is brief and is always carried out). This can save a lot of time in what would otherwise be needless repetition of calculations.

The Computations tab also has a "Stop computation" button that you can use to abort calculations, *e.g.* if you decide to make further changes to your Project set up while calculations are on the way. This can be useful for very large Projects, where the calculations can take some time. When you do this, the AVA software accepts the results that have already been re-computed, but it also keeps the results from the previous computation that have not yet been altered by the re-computation. The reason for this is that Amplicon Projects are incremental, and re-calculations are often done after Read Data Set(s) or Sample(s) are added to the Project; therefore, much of the previously computed results is still valid. On the other hand, the results that were in the process of being re-computed at the time of the interruption could truly be corrupted; the results for these computations, caught in an intermediate state, are removed from the computation's output (such as in the Variants tab) and the navigation elements that would be used to load these results are disabled (such as those used to load multiple alignments in the Global Align tab).

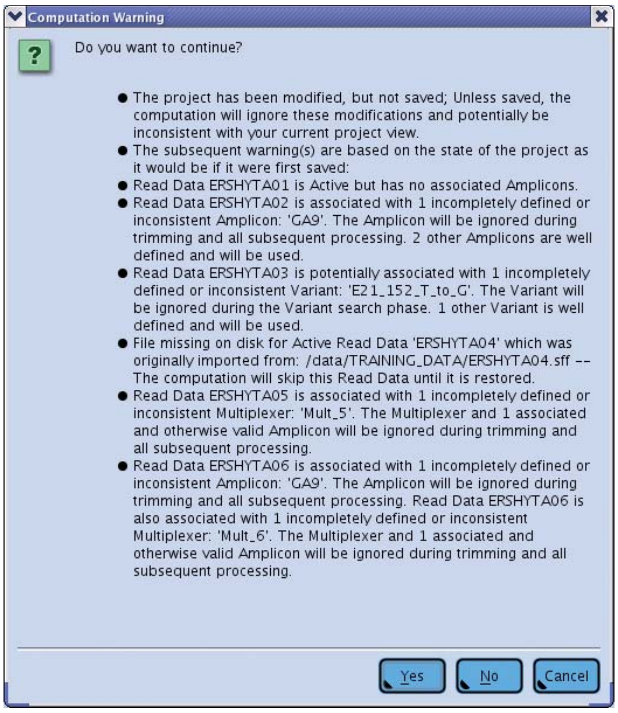
Interrupted computations: If a computation (or re-computation) is interrupted, part of the output may not match the state of the saved Project. While the AVA software withholds the potentially corrupted results from the data that was being processed at the time of the interruption, it also maintains the results from previous computations that had not yet been altered at the time of the interruption. Be aware that those older results may not be consistent with more recent updates to the Project. The outcome of this is similar to the case described in the "Caution" at the end of section 9.3 (editing a Project in a manner that is germane to previously computed results). If you find that the data in these tabs does not reflect the current state of the Project, try re-computing it. The only way to be completely sure that the Project is consistent is to allow the computation to run to completion, without interruption.

Before starting the actual computations, the AVA software validates the computationally relevant aspects of the Project setup. This includes most Project elements' definitions and their associations, such as a Variant pattern and its relationship to the Reference Sequence, or the particular pairings of Samples and Amplicons on particular Read Data Sets. If the software finds any problems, it displays warning messages, giving the user a chance to address them prior to running the computation. The user may elect to ignore the warnings and proceed; the computation will still run, and shouldn't throw any errors, but the results may be incomplete because the computation will skip the problematic elements.

Figure 9–44 shows the different kinds of warning messages that can occur:

- The warning that the Project has been modified, but not saved, so the computation might produce results that are out of sync with the Project's current state.
- The message indicating that further messages in the Computation Warning window are based on the Project in its current state, *i.e.* as computation would see it if it were saved; this gives you advance warning of problems in a Project even before you save it to disk. If your Project is up to date on the disk, the other messages below may still occur but they would concern the saved Project.
- A warning that a Read Data Set is active in the Project, but has no associated Amplicons (because no Sample-Amplicon pairs have been assigned to it). This may be an oversight on the part of the user, in which case a large part of the expected output might be missing if the computation were carried out. This warning also appears if a valid Multiplexer is associated with the Read Data Set but no Amplicons are associated with the Multiplexer (and there are no other Amplicons associated with the Read Data Set directly via Samples or indirectly via another Multiplexer).

- A warning that one or more Amplicons that are associated with some Read Data Set have problems. Such problems may include:
  - the Amplicon is incompletely defined (no primers, or target start/end not specified)
  - the Amplicon appears to be inconsistent with its Reference Sequence (the Start/End of the Target are outside the range of the Reference Sequence; this may occur if the user edited the Reference Sequence after assigning the Start/End of the associated Amplicon)
  - the Reference Sequence itself is incompletely defined (*e.g.* it was given a name, but no actual sequence; in this case the Amplicon wouldn't likely have any Target Start/End set either).
- A warning that one or more Variants that are potentially associated with a Read Data Set (the Variant location on its Reference Sequences is spanned by an Amplicon that is associated with the Read Data Set) have problems. Possible Variant definition problems include:
  - the Variant pattern is inconsistent with the Reference Sequence [*e.g.* a substitute constraint specifies that the substituted nucleotide is the same as the one already in the Reference Sequence (should be a match constraint instead)]
  - some or all the positions of the Variant Pattern are not in the Reference Sequence (as above, this may occur if the user edited the Reference Sequence after defining the Pattern of the associated Variant)
- A warning that the file for an active Read Data Set is missing. This warning would be triggered if a Read Data Set was imported as a symbolic link, and the file that the link points to has been moved, deleted, or become corrupted. The warning message tells where the link was expecting to find the file so that it can be restored to the proper location.
- A warning that an inconsistent Multiplexer is associated with a Read Data Set. Problems with a Multiplexer may include:
  - The Multiplexer has not been completely defined (the Encoding, MIDs, and/or Samples have not been specified).
  - There is a problem with the set of MIDs on either the Primer 1 or Primer 2 sides being used by the Multiplexer to encode Samples. Examples of MID problems are:
    - An MID is undefined.
    - At least 2 MIDs are defined and they are not of uniform length (at least one of them has a different sequence length than another).
    - An MID in the set has an identical sequence to another MID in the set.



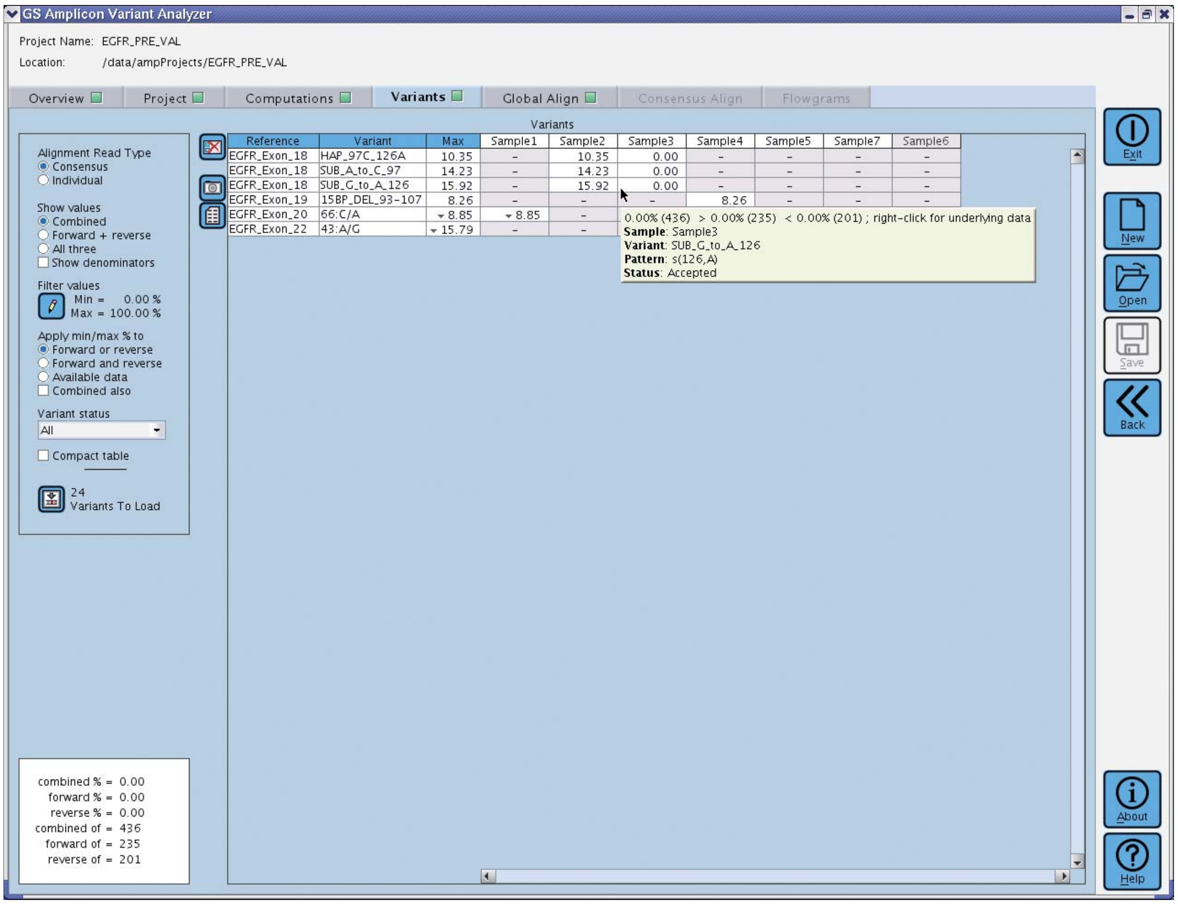
Do you want to continue?

- The project has been modified, but not saved; Unless saved, the computation will ignore these modifications and potentially be inconsistent with your current project view.
- The subsequent warning(s) are based on the state of the project as it would be if it were first saved:
- Read Data ERSHYTA01 is Active but has no associated Amplicons.
- Read Data ERSHYTA02 is associated with 1 incompletely defined or inconsistent Amplicon: 'GA9'. The Amplicon will be ignored during trimming and all subsequent processing. 2 other Amplicons are well defined and will be used.
- Read Data ERSHYTA03 is potentially associated with 1 incompletely defined or inconsistent Variant: 'E21_152_T_to_G'. The Variant will be ignored during the Variant search phase. 1 other Variant is well defined and will be used.
- File missing on disk for Active Read Data 'ERSHYTA04' which was originally imported from: /data/TRAINING_DATA/ERSHYTA04.sff -- The computation will skip this Read Data until it is restored.
- Read Data ERSHYTA05 is associated with 1 incompletely defined or inconsistent Multiplexer: 'Mult_5'. The Multiplexer and 1 associated and otherwise valid Amplicon will be ignored during trimming and all subsequent processing.
- Read Data ERSHYTA06 is associated with 1 incompletely defined or inconsistent Amplicon: 'GA9'. The Amplicon will be ignored during trimming and all subsequent processing. Read Data ERSHYTA06 is also associated with 1 incompletely defined or inconsistent Multiplexer: 'Mult_6'. The Multiplexer and 1 associated and otherwise valid Amplicon will be ignored during trimming and all subsequent processing.

Figure 9–44: A Computation Warning message showing several of the types of message that can occur. The final warning message illustrates that more than one warning for a single Read Data Set can get merged together into a single message.

9.5 The Variants Tab

The Variants tab shows the frequency at which all the Variants defined in the Project were observed, for each Sample defined in the Project, at the time of the last computation (Figure 9–45). This data is presented in a convenient Table that allows the user to compare the frequencies of the Variants across Samples. This tab is not to be confused with the Variants sub-tab of the Project tab, which is used to store and edit the definitions of Variants. In addition to the Variants Frequency Table, the Variants tab also contains controls to modify the content and format in which the Variant frequency data is displayed. The customary Mouse Tracker, on the left, is also present to provide additional details as you mouse over the cells and headers of the Variants Frequency Table.



Project Name: EGFR_PRE_VAL
Location: /data/ampProjects/EGFR_PRE_VAL

Alignment Read Type
[✓ Consensus]
Individual

Show values
[✓ Combined]
Forward + reverse
All three
Show denominators

Filter values
Min = 0.00 %
Max = 100.00 %

Apply min/max % to
[✓ Forward or reverse]
Forward and reverse
Available data
Combined also

Variant status
All

Compact table

24
Variants To Load

Variants

Reference	Variant	Max	Sample1	Sample2	Sample3	Sample4	Sample5	Sample7	Sample6
EGFR_Exon_18	HAP_97C_126A	10.35	–	10.35	0.00	–	–	–	–
EGFR_Exon_18	SUB_A_to_C_97	14.23	–	14.23	0.00	–	–	–	–
EGFR_Exon_18	SUB_G_to_A_126	15.92	–	15.92	0.00	–	–	–	–
EGFR_Exon_19	15BP_DEL_93-107	8.26	–	–	–	8.26	–	–	–
EGFR_Exon_20	66:C/A	▾ 8.85	▾ 8.85	–					
EGFR_Exon_22	43:A/G	▾ 15.79	–	–					

0.00% (436) > 0.00% (235) < 0.00% (201) ; right-click for underlying data
Sample: Sample3
Variant: SUB_G_to_A_126
Pattern: s(126,A)
Status: Accepted

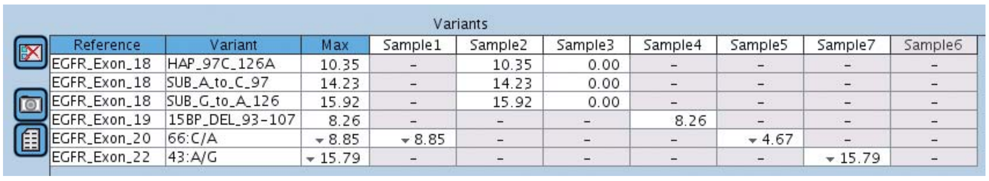
```
combined % = 0.00
 forward % = 0.00
 reverse % = 0.00
combined of = 436
 forward of = 235
 reverse of = 201
```

Figure 9–45: The Variants tab

9.5.1 The Variants Frequency Table

9.5.1.1 General organization

The Variants Frequency Table shows the results from one Variant per row and one Sample per column (Figure 9–46). Initially, cells that contain data are white and cells that contain no data are grayed-out (this can happen, for example, if a Sample has no associated Amplicons whose sequence covers the Variant; see below). When an entire row or column is grayed-out, it is moved to the bottom or right of the Table, respectively. This grayed-out scheme is also used by various display features whereby you can filter out the data according to selected criteria, leaving the data of most interest in white cells, in the upper-left area of the Table. (The 'Compact table' option from the Variant data display controls can then remove from view all completely grayed-out rows and columns; see section 9.5.2.4.)



Variants

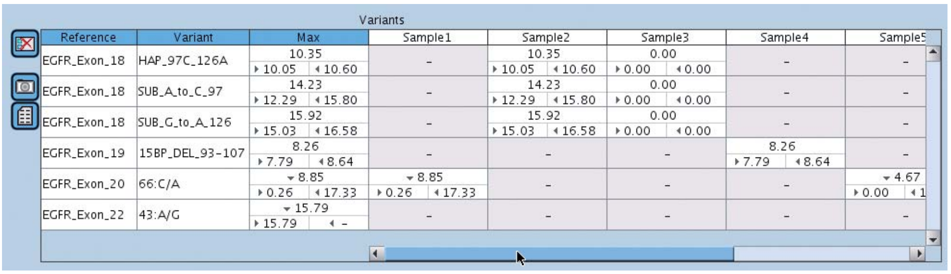
Reference	Variant	Max	Sample1	Sample2	Sample3	Sample4	Sample5	Sample7	Sample6
EGFR_Exon_18	HAP_97C_126A	10.35	–	10.35	0.00	–	–	–	–
EGFR_Exon_18	SUB_A_to_C_97	14.23	–	14.23	0.00	–	–	–	–
EGFR_Exon_18	SUB_G_to_A_126	15.92	–	15.92	0.00	–	–	–	–
EGFR_Exon_19	15BP_DEL_93-107	8.26	–	–	–	8.26	–	–	–
EGFR_Exon_20	66:C/A	▾ 8.85	▾ 8.85	–	–	–	▾ 4.67	–	–
EGFR_Exon_22	43:A/G	▾ 15.79	–	–	–	–	–	▾ 15.79	–

Figure 9–46: The Variants Frequency Table

With respect to columns, the Table is divided into two main parts:

- The first three columns (with blue Header cells) act as "Headers" to the Variant rows:
  - The 'Reference' column gives the names of the Reference Sequences to which the Variants (in the second column) are associated. The rows are initially sorted from top to bottom in alphabetical order of the Reference Sequence names (this applies separately to rows that contain at least one white cell and to the grayed-out rows that appear at the bottom of the Table).
  - The 'Variants' column gives the names of the Variants whose occurrence frequency for each Sample are given in each row. If two or more Variants are associated with any given Reference Sequence, the Variant names are used for second level sorting in the initial display.
  - The 'Max' column displays the maximum frequency observed for each Variant across all the Samples displayed in the Table. This can be a convenient indicator of whether the software detected a given Variant at sufficient frequency in the computed data to warrant further examination in the individual 'Sample' columns, to the right.
- All other columns give the occurrence frequency observed for each Variant for one Sample (identified in the column Header), one column for each Sample. These columns (excluding the first three) are initially sorted from left to right in alphabetical order of the Sample names (this applies separately to columns that contain at least one white cell and to the grayed-out columns that appear at the right end of the Table).

While the first three columns are always visible, you can scroll through the Sample columns using the scroll bar located at the bottom-right of the Table [see Figure 9–47, which displays the same data as Figure 9–46 but in a more expanded display (see section 9.5.2), that shows the scroll bar]. As you scroll to the right, the leftmost Sample columns appear to slide behind the first three rows, so you may end up in situations where you display a partial column just after the 'Max' column.



Variants

Reference	Variant	Max	Sample1	Sample2	Sample3	Sample4	Sample5
EGFR_Exon_18	HAP_97C_126A	10.35 ▸ 10.05 ◂ 10.60	–	10.35 ▸ 10.05 ◂ 10.60	0.00 ▸ 0.00 ◂ 0.00	–	–
EGFR_Exon_18	SUB_A_to_C_97	14.23 ▸ 12.29 ◂ 15.80	–	14.23 ▸ 12.29 ◂ 15.80	0.00 ▸ 0.00 ◂ 0.00	–	–
EGFR_Exon_18	SUB_G_to_A_126	15.92 ▸ 15.03 ◂ 16.58	–	15.92 ▸ 15.03 ◂ 16.58	0.00 ▸ 0.00 ◂ 0.00	–	–
EGFR_Exon_19	15BP_DEL_93-107	8.26 ▸ 7.79 ◂ 8.64	–	–	–	8.26 ▸ 7.79 ◂ 8.64	–
EGFR_Exon_20	66:C/A	▾ 8.85 ▸ 0.26 ◂ 17.33	▾ 8.85 ▸ 0.26 ◂ 17.33	–	–	–	▾ 4.67 ▸ 0.00 ◂ 1
EGFR_Exon_22	43:A/G	▾ 15.79 ▸ 15.79 ◂ –	–	–	–	–	–

Figure 9–47: The Variants Frequency Table showing the same data as in Figure 9–46, but in a more expanded form, showing the scrolling feature that applies to the 'Sample' columns.

Since Variants are defined in the context of a Reference Sequence, and Samples are associated with Amplicons (which are in turn defined in the context of a Reference Sequence), there may be Samples in your Project that are not valid candidates for a particular Variant scan. This happens if all the Amplicons associated with the Sample are defined relative to a different Reference Sequence than the Variant; or even if they are from the same Reference Sequence, if they do not cover regions of the Reference Sequence where the Variant is defined. It may even be that the Sample was associated with an Amplicon that should have covered the region of variation, but for some reason no actual reads were sequenced that covered the variant (as distinguished from the case where some reads cover the region of variation, but none of them satisfy the constraints given by the Variant's Pattern). In these cases, the corresponding Sample-Variant cell in the Table will be grayed-out and contain a single dash ('-') character.

The Variants tab also has various common features such as a Mouse Tracker and the "Save table snapshot to .png file" and "Save table to .xls (spreadsheet) file" buttons. Note that since each cell can contain up to 6 values (frequency in forward, reverse, and combined reads, and the three associated denominators of those frequencies; see section 9.5.2.2), the spreadsheet may have multiple columns for each sample to accommodate all these values.

Another useful feature is that if you pause the mouse over any cell of the Table, a screen tip will open providing relevant information about the content of that cell, as follows:

- Column header cell
  - Instructions on the right-click options (see section 9.5.1.2)
- Reference cell
  - Name of the Reference Sequence
  - Beginning of its DNA sequence
  - Reference Sequence Annotation
- Variant cell
  - Name of the Variant
  - "Pattern" of the Variant, in the Variant Definition Syntax (see section 9.3.2.5.2)
  - Variant Annotation
  - "Status" of the Variant (see section 9.3.2.5.3)
- All "Frequency" cells ('Max' or 'Sample' columns; shown in Figure 9–45):
  - Frequency (and number of reads) for the combined orientations and for each orientation; and instructions on the right-click options (see section 9.5.1.2)
  - Name of the Sample
  - Name of the Variant
  - "Pattern" of the Variant, in the Variant Definition Syntax (see section 9.3.2.5.2)
  - "Status" of the Variant (see section 9.3.2.5.3)

9.5.1.2 Organizing Data in the Variants Frequency Table

The Variants Frequency Table contains the summary results of your Amplicon Project provided by the AVA software: the frequency at which each defined Variant was observed in each Sample, in the Read Data Set(s) analyzed; this frequency is expressed as a percentage of the number of reads included in the calculation. As you examine these results, however, it may be convenient to sort the data or to bring the focus on only certain Variants or Samples at a time. This would be especially true in a large Project with many defined Variants and/or many Samples.

To help with this, right-clicking on any column or row "header" cell opens a contextual menu that offers several sorting or filtering options (remember that the cells in the first three columns are all "row headers"; see section 9.5.1.1):

- If you right-click on the Reference or Variant column header, the contextual menu will include show/ignore options that apply to rows along with reversion options but no column sort options are available (Figure 9–48**A-B**).
- If you right-click on the 'Max', column header, the contextual menu will contain hybrid options: the sort options will apply to the given column, but the show and ignore options will apply to rows (Figure 9–48**C)**.
- If you right-click on a Sample column header, the options in the contextual menu will apply to that column, with additional options that apply to all the other Sample columns collectively; the sort options sort the values found in the column, effectively reordering the rows of Variant data (Figure 9–48**D**).
- If you right-click on a cell in the column underneath the Reference label, there will be options that apply to all rows collectively and also to that subset of rows that are associated with the specific Reference Sequence: *i.e.*, all those rows that have data for Variants associated with that Reference Sequence (Figure 9–48**E**).
- If you right-click on a cell in the columns underneath the Variant or Max headers, the options in the contextual menu will apply to that row or to all rows of Variants collectively; the sort options sort the values found in the row, effectively reordering the columns of Sample data. The menu also has a Variant Status option that pops up a set of radio buttons for Status selection (Figure 9–48**F**).
- If you right-click on a Sample/Variant intersection cell, the contextual menu contains options to view the Global Alignment from which the frequency data for that cell was derived, and to edit the Variant Status or remove the Variant from the project (Figure 9–48**G**).
- If you right-click on a cell in the columns underneath the Variant or Max headers or if you right-click on a Sample/Variant intersection cell, there is also a "Define Haplotype" option (Figure 9–48 **F** and **G**). This option is inactive unless rows for two or more Variants from the same Reference are selected. The "Define Haplotype" option is described in section 9.5.1.4.

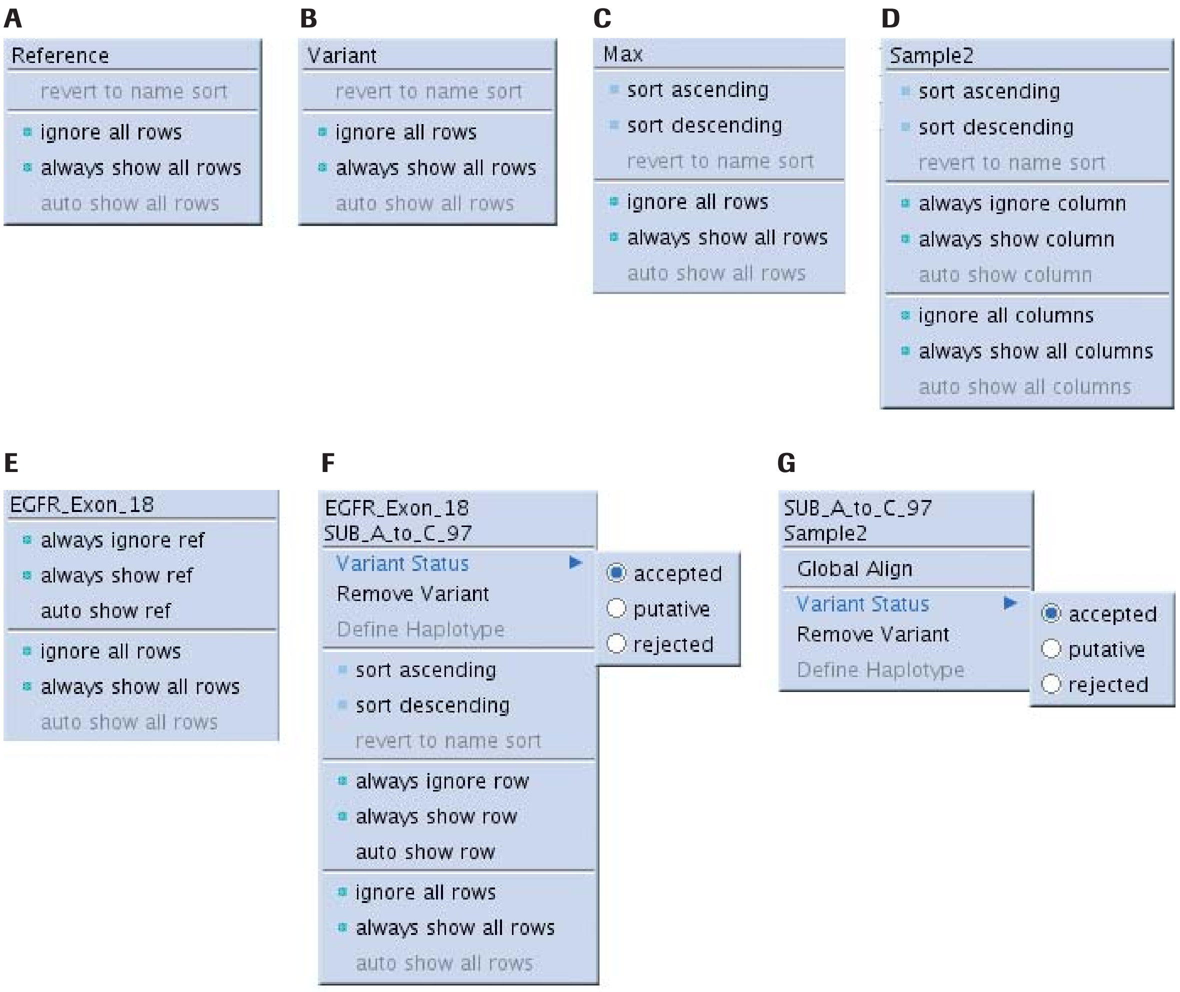


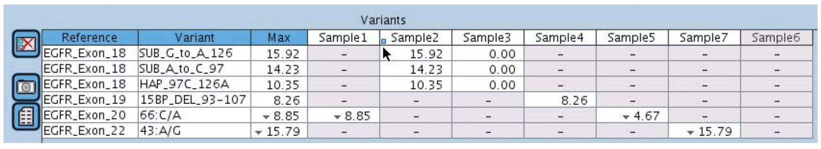
Figure 9–48: The contextual menus available in the Variants tab (see description above)

The data organization tools offered in these contextual menus include sorting, ignore filters, show filters, and option reversions. These are described below.

9.5.1.2.1 Sort options

- sort ascending
- sort descending

These options sort the columns/rows according to the 'Combined' Variant frequency in the column/row on which you right-clicked. A blue marker appears in the lower-left corner of the header cell (the 'Max' header for rows) for the column/row that is the basis of the sort. The 'Max' column header can be sorted and marked just like the Sample columns. You may apply a sort to only one row and only one column at a time, but you can sort the Table according to both a row and a column simultaneously.



Variants

Reference	Variant	Max	Sample1	Sample2	Sample3	Sample4	Sample5	Sample7	Sample6
EGFR_Exon_18	SUB_G_to_A_126	15.92	-	15.92	0.00	-	-	-	-
EGFR_Exon_18	SUB_A_to_C_97	14.23	-	14.23	0.00	-	-	-	-
EGFR_Exon_18	HAP_97C_126A	10.35	-	10.35	0.00	-	-	-	-
EGFR_Exon_19	15BP_DEL_93-107	8.26	-	-	-	8.26	-	-	-
EGFR_Exon_20	66:C/A	▾ 8.85	▾ 8.85	-	-	-	▾ 4.67	-	-
EGFR_Exon_22	43:A/G	▾ 15.79	-	-	-	-	-	▾ 15.79	-

Figure 9–49: The Variants Frequency Table, sorted (descending) according to the Sample2 column. Note the blue marker in the lower-left corner of the Sample2 column header; compare with Figure 9–46.

9.5.1.2.2 Ignore filters

- always ignore column/row
- ignore all columns/rows

As indicated above (section 9.5.1.1), the software grays-out cells that contain no data and shifts rows and columns that contain only gray cells to the bottom or right ends of the Table. This focuses the cells of interest (white) to the upper-left area of the Table. With the 'always ignore column/row' options, you can gray-out any columns/rows (which moves them to the right/bottom of the Table) to help you focus on data of current interest. If you want to gray-out most columns/rows, you can use the 'ignore all' option to gray them all first, and then apply the 'show' filter (see below) to re-focus on only the ones for which you have a current interest. (Cells will also be grayed-out if they fail the Min/Max filter, from the Variant data display controls; see section 9.5.2.3.)

A blue marker appears in the upper-left corner of the header cell (the 'Max' header for rows) in the column/row that you chose to ignore, to remind you that this filter was applied manually. You can ignore any number of columns or rows. You can also ignore a column or a row that was used for sorting; in this case, the corresponding header cell will have both blue markers, in its upper- and lower-left corners.

9.5.1.2.3 Show filters

- always show column/row
- always show all columns/rows

These filters have the opposite effect of the 'ignore' filters (including moving the columns/rows to the upper-left area of the Table). They use the same upper-left blue marker in the header cell as the 'ignore' filters, to indicate that the show filter was applied manually to the column/row. They will override the application of an 'ignore' filter, as described above, and can be used to force a Sample or Variant into the display even if it has failed the Min/Max filters (see section 9.5.2.3). This can be particularly useful for forcing the inclusion of negative controls in the display (which would typically fail a minimum filter).

9.5.1.2.4 Option reversions

The right-click 'revert' options below can be used to selectively undo any of the option choices you have made.

- 'revert to name sort' removes the numerical sort on the values of column or row (reverts to the default alphabetical order of the column or row header labels), and removes the lower-left blue marker in the header cell of the column/row that had been used to sort. You don't have to click on the specific column/row that was the object of the sorting to get this option; the 'revert to name sort' from another column/row would have the same effect.
- 'auto show column/row' reverts any show or ignore filters that may have been applied to the column or row on whose header cell you right-clicked. If that column/row was also used for sorting, sorting is not effected.
- 'auto show all columns/rows' reverts all show or ignore filters that may have been applied to any column or row (without removing any sorting that may have been applied to the table).

If you want to undo all the sorts and filters you applied (*i.e.* restore all defaults), use the 'Reset table' button.

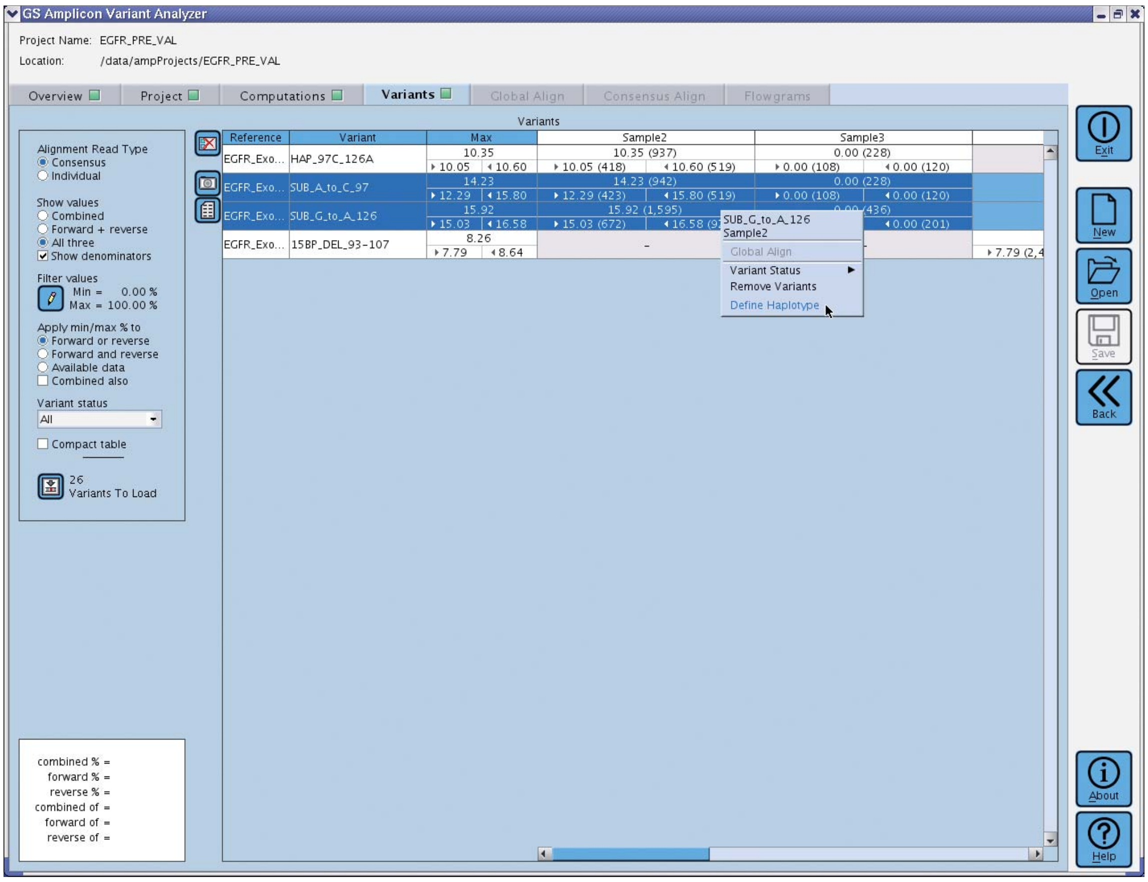
The 'Reset table' button removes all sorting and ignore/show filters that have been applied to the table and restores the table to its state prior to the filters. Note that this does not affect any of the table formatting options from the Variant data display controls (see section 9.5.2), including the Min/Max filter settings (section 9.5.2.3).

9.5.1.3 Populating the Global Align Tab from the Variants Tab

As described above, right-clicking on a column or row header cell in the Variants Frequency Table opens a contextual menu that includes the sorting and filter options. By contrast, right-clicking on a cell in the 'body' of the Table opens a contextual menu that allows you to populate the Global Align tab with the multi-alignment of the reads belonging to the Sample-Variant combination specified by that cell. This is the best way to initiate your exploration of the underlying reads if you are specifically interested in results for a given Variant, as it will selectively include reads from all Amplicons that cover that Variant (and that are associated with that Sample). Such a selection could not be done directly from any of the tree views (see sections 9.3.1 and 9.6.1) because these only allow you to choose a single Sample-Amplicon pair. When you navigate to the Global Align tab from a specific Variant on the Variants tab, the sof highlights the leftmost position of the Variant Pattern, to help you visually key in on the variation.

9.5.1.4 Defining a Haplotype from the Variants Tab

Right-clicking on a cell in the columns underneath the Variant or Max headers, or right-clicking on a Sample/Variant intersection cell, reveals a "Define Haplotype" option (Figure 9–48 **F** and **G**) in the contextual menu. This option allows you to select multiple rows of individual Variants in the Variant Frequencies Table to propose a new Variant that requires that all the selected Variants be found together as a haplotype. This option is grayed-out and inactive unless rows for two or more Variants from the same Reference are selected (selecting multiple Variant rows from mixed References will inactivate the option even if any of the selected Variants share a Reference). After selecting two or more such Variant rows, you can right-click over any cell in the selection except for a Reference cell, to access the "Define Haplotype" option (Figure 9–50).



Project Name: EGFR_PRE_VAL
Location: /data/ampProjects/EGFR_PRE_VAL

Variants

Reference	Variant	Max		Sample2		Sample3		
EGFR_Exo...	HAP_97C_126A	10.35		10.35 (937)		0.00 (228)		
		▸ 10.05	◂ 10.60	▸ 10.05 (418)	◂ 10.60 (519)	▸ 0.00 (108)	◂ 0.00 (120)	
EGFR_Exo...	SUB_A_to_C_97	14.23		14.23 (942)		0.00 (228)		
		▸ 12.29	◂ 15.80	▸ 12.29 (423)	◂ 15.80 (519)	▸ 0.00 (108)	◂ 0.00 (120)	
EGFR_Exo...	SUB_G_to_A_126	15.92		15.92 (1,595)		0.00 (436)		
		▸ 15.03	◂ 16.58	▸ 15.03 (672)	◂ 16.58 (9		◂ 0.00 (201)	
EGFR_Exo...	15BP_DEL_93-107	8.26		-		-		
		▸ 7.79	◂ 8.64					▸ 7.79 (2,4

Alignment Read Type
Consensus [✓]
Individual

Show values
Combined
Forward + reverse
All three [✓]
Show denominators [✓]

Filter values
Min = 0.00 %
Max = 100.00 %

Apply min/max % to
Forward or reverse [✓]
Forward and reverse
Available data
Combined also

Variant status
All

Compact table

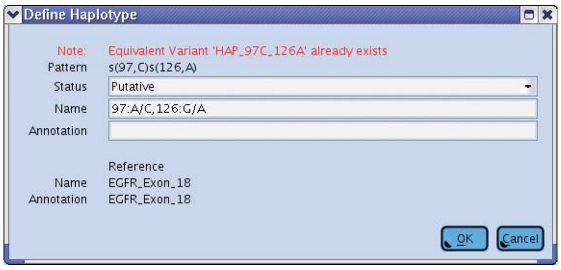
26
Variants To Load

combined % =
forward % =
reverse % =
combined of =
forward of =
reverse of =

Figure 9–50: The Variants Frequency Table with two individual Variant rows selected. Right-clicking over a non-Reference cell in the selection provides a contextual menu with an active "Define Haplotype" option that can be used to propose a new Variant that requires the selected Variants to be found together as a haplotype.

Choosing the option will bring up the "Define Haplotype" window (Figure 9–51). This window functions similarly to the "Approve new variant" window used by the "Declare project variant" function of the Global and Consensus Alignment tabs (see section 9.6.3.3, Figure 9–61, and Figure 10–36, below). The main difference between the use of "Define Haplotype" and "Approve new variant" is a matter of context. The "Approve new variant" function is triggered in the context of a multiple alignment where the linkage of Variants in a haplotype can be visually verified. The "Define Haplotype" function is triggered from the main Variant Tab, where similar Variant frequencies may be suggestive of a haplotype, though the similarity may also simply be coincidental. Accordingly, the "Define Haplotype" window defaults to creating the haplotype with a "Putative" Status.

If the Variants selected in the haplotype have any contradictory elements (*e.g.*, specifying two different SNPs for the same position or specifying both a SNP and a deletion for the same position), then a window pops up to explain the problem, just like when the user enters erroneous variant patterns (Figure 9–28). The user can then edit the resulting pattern and create a semantically correct haplotype definition; more likely, however, one would select the Cancel button since any identified errors would actually disprove the coincidental existence of the Variants as a valid haplotype.



Note: Equivalent Variant 'HAP_97C_126A' already exists
Pattern s(97,C)s(126,A)
Status Putative
Name 97:A/C,126:G/A
Annotation

Reference
Name EGFR_Exon_18
Annotation EGFR_Exon_18

Figure 9–51: The "Define Haplotype" window with a Pattern built from the Variant selections made in Figure 9–50. The Status defaults to Putative but can be edited, and a red warning is given to indicate that a variant with the same pattern already exists.

The Auto-Detected Variant system does not propose haplotypes (beyond the special case of multiple base pair deletions), so haplotypes must be entered into the system by the "Define Haplotype" or "Approve new variant" methods, or they must be defined from scratch in the Variant Definition Table. In large projects it may become difficult to keep track of all the proposed haplotypes that have already been entered into the system, so if an attempt is made to use the "Define Haplotype" or "Approve new variant" functions to propose a haplotype that already exists, a red warning message stating the redundancy will appear in the window (Figure 9–51).

9.5.1.5 Editing/Removing Variants from the Variants Tab

Right-clicking on a cell in the "body" of the Variants Frequency Table, at a Sample/Variant intersection (as above in section 9.5.1.3 and as shown in Figure 9–48**G**), provides a contextual menu that allows you to edit the Variant Status or remove Variants from the project. The same functions can be carried out using the Variants Definition Table in the Variants sub-tab of the Project Tab (see section 9.3.2.5). The Variants Frequency Table is a convenient place to decide on the Status of a Variant because the incidence frequency across Samples is available for examination. You can use the shift or control keys to select multiple rows and you can delete them or adjust their Status as a bulk operation. If removing Variants, you are provided with a "Yes to All" confirmation window (similar to the case described in section 9.3.2 and Figure 9–15). A bulk Status edit occurs without any confirmation step.

- The Status is applied to the Variant and not a Sample. You can't mark the Status for a Variant to "Accepted" for one Sample and "Putative" for another; the Status is applied globally to the Variant regardless of the Sample(s) in which it is found.

- If you remove a Variant from the Project it gets removed from every Sample column in the Variants Frequency Table, not just from the Sample corresponding to the cell on which you clicked. If the Variants you remove happen to be Auto-Detected Variants, they could be re-imported the next time you load the Auto-Detected Variants (depending on your filter settings for the load). For this reason, it is usually better to mark the Variants' Status to "Rejected" rather than remove them from the Project, until you are sure that you will not be loading any more Auto-Detected Variants into the Project.

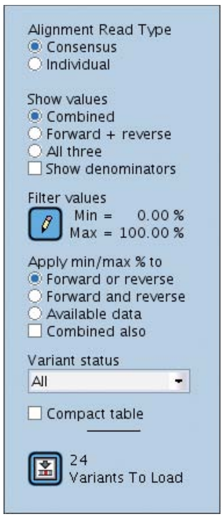
9.5.1.6 The Mouse Tracker

The Mouse Tracker on the Variants tab is slightly more complex than on other tabs because the information it displays is highly context-sensitive:

- If the mouse is over a header cell in the Variants Frequency Table, the Mouse Tracker gives general information about how many different items (Reference Sequences for the 'Reference' column; Variants for all other columns) are present in the column, and how many of these comprise data that meet the current Min/Max filter settings.
- If the mouse is over a specific 'Reference' cell, the Mouse Tracker shows the number of Variants and Samples associated with that Reference Sequence, as well as a count of how many of those Samples comprise data that meet the current Min/Max filter settings.
- If the mouse is over a specific 'Variant' cell, the Mouse Tracker shows how many Samples are in the table for that Variant, as well as a count of how many of those Samples comprise data that meet the current Min/Max filter settings.
- If the mouse is over a specific Sample-Variant cell, the Mouse Tracker shows a set of frequency statistics for that Sample-Variant combination including the frequencies at which that Variant occurred in this Sample among reads in the forward, reverse, and combined read orientations; and corresponding denominators (number of reads covering the Variant position) used in these frequency calculations. All these values are shown in the Mouse Tracker even if the setting of the 'Show values' option (see section 9.5.2.2) is more restrictive; this way, you have access to all the information even if you choose a more compact Table view.
- Finally, if the mouse is over a specific 'Max' cell, the Mouse Tracker behaves as if you were mousing over the actual Sample-Variant cell that contains those maximum values. Note that 'Max' cells in the Table do not display denominators even when the 'Show denominators' option is chosen (section 9.5.2.2), but the Mouse Tracker does.

9.5.2 Variant Data Display Controls

A box located in the top-left corner of the Variants tab contains various display option tools that allow you to control the display of the Variant data in the Variants Frequency Table (Figure 9–52).



Alignment Read Type
[✓ Consensus]
Individual

Show values
[✓ Combined]
Forward + reverse
All three
Show denominators

Filter values
Min = 0.00 %
Max = 100.00 %

Apply min/max % to
[✓ Forward or reverse]
Forward and reverse
Available data
Combined also

Variant status
All

Compact table

24
Variants To Load

Figure 9–52: The Variant data display control tools

9.5.2.1 The Alignment Read Type Controls

The 'Alignment Read Type' radio buttons allow you to select 'Consensus' or 'Individual'. Consensi are a collapsed representation of multiple similar reads (see section 9.6.4.2) and have a single coverage value over their entire length. Creating consensus reads simplifies data analysis and eliminates noise. However, there are sometimes discrepancies in read length within the consensus, making the coverage non-uniform. If a Variant is located in one of the regions of the consensus with lower actual coverage, the Variant frequencies reported with the 'Consensus' option can be misleading. Similarly, if a true variation is misinterpreted as noise, it might be eliminated from all the constructed consensi and a Variant of interest might go unnoticed. Looking at Variant frequencies based on individual reads rather than consensi gives more literal values. It is good to look at and compare Variant frequencies from both types of reads. If the numbers are in close agreement, they bolster one another, but if they are significantly different from each other, you may need to dig into the consensi to get a better understanding of the situation.

9.5.2.2 The Show Values Controls

The 'Show values' set of radio buttons control how to include orientation-specific Variant frequency data in the Table. Variants can potentially be found in reads or consensi of both orientations, but there may be some situations in which the design of your Amplicon libraries or the combination of Amplicon length and read length is such that certain regions of the Reference Sequence are only covered by reads of a single orientation, and Variants defined in those regions are likewise limited to single orientation coverage. Even when both orientations are available to scan for Variants, there can sometimes be discrepancies between Variant frequencies in one orientation versus the other.

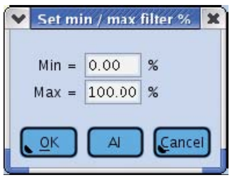
- Choosing the 'Combined' option merges the forward and reverse results together; the Sample-Variant cells in the Table each contain a single Variant frequency value. The Variant frequencies for the reads in each orientation are still calculated, however, and if the AVA software detects a significant difference between the combined value and the individual forward and reverse values, a small down-pointing triangle appears to the left of the value to alert you.
- Choosing 'Forward + reverse' alters the format of the table so that the Sample-Variant cells are divided into two side-by-side sub-cells. The sub-cells show the Variant frequency values broken out by orientation (identified by arrowheads) rather than showing a single combined value.
- Choosing the 'All three' option results in a more complex table format arrangement where each Sample-Variant cell gets subdivided into three sub-cells: the two side-by-side orientation-specific sub-cells are surmounted by the third, 'Combined' sub-cell (see Figure 9–47, above). Again, a small down-pointing triangle appears to the left of the combined value if the AVA software detects a significant difference between the combined Variant frequency value and that of either orientation.

The 'Show denominators' checkbox adds read counts to the Sample-Variant cells (or sub-cell) as a number in parentheses following the Variant frequency values. This allows you to judge the reliability of Variant frequencies based on sample size and can also be of assistance when comparing Variant frequencies by orientation. If one orientation is much more highly represented than the other, you may choose to ignore the value from the underrepresented orientation. All these values (Variant frequencies and number of reads) can also be seen in the Mouse Tracker when the mouse is over a Sample-Variant cell of the Table.

9.5.2.3 The Min / Max Filters

The 'Filter values' controls allow you to set a minimum and maximum Variant frequency on which you want to focus in the Variant Frequency Table: Sample-Variant cells that do not meet the min and max filters are grayed-out, and if any rows or columns are entirely grayed-out, they get moved to bottom or right of the Table; the 'Compact table' option can remove grayed-out rows and columns from view in the Table (see section 9.5.2.4).

- Clicking the 'Change min/max %' button opens the 'Set min / max filter %' window where you can set (to 2 decimal places) the minimum and maximum Variant frequency values (percentage) to use as filters (Figure 9–53). Values must be within the range '0.00' to '100.00', and the maximum value must be greater than or equal to the minimum value. Click 'OK' to accept and apply your min/max filter selections. The 'All' button resets the values to a minimum of 0.00 and a maximum of 100.00 (thus All Variant frequency values pass filter).



Min = 0.00 %
Max = 100.00 %

Figure 9–53: The 'Set min / max filter %' window

- The behavior of the minimum and maximum filters is modified by the 'Apply min/max % to' set of radio buttons, located beneath the 'Change min/max %' button. These controls are used to determine what set of Variant frequencies (forward, reverse, and/or combined) the software applies to each Sample-Variant cell when deciding if the cell survives the min/max filters.
  - 'Forward or reverse' causes the min/max settings to be applied only to the orientation-specific Variant frequency values. *Either* the forward *or* the reverse Variant frequency (or both) must meet both the minimum and the maximum filters for the Sample-Variant cell to survive the filters and remain in the table as a cell with a white background (rather than being grayed-out for failing the filter).
  - 'Forward and reverse' requires that the forward and reverse Variant frequency values *both* meet the minimum and maximum filters independently. If one orientation fails, the cell does not survive the filter and is grayed-out.
  - 'Available data' is a more sophisticated version of the 'Forward and reverse' option. In some cases, you may have intentionally sequenced only a single orientation, or the length of your Amplicon may be such that at the read length provided by the sequencing Run, the forward and reverse reads cannot provide double-orientation coverage in the region where your Variant is located. In those cases, you may not want to penalize a Variant for being represented by a single orientation when it was impossible for it to be represented in both. The 'Available data' option checks to see if there is read coverage from both orientations at the Variant position. If the coverage is all of one orientation, the min and max filters are applied to the Variant frequency value for that orientation. If coverage of the variant position comes from both orientations, both the forward and the reverse frequencies must independently survive the min and max filters as with the 'Forward and reverse' option.
- The 'Combined also' checkbox instructs the software to also take the 'Combined' Variant frequency value into consideration when applying the min/max filters. If you have chosen the 'Forward or reverse' option, the 'Combined' frequency is treated like another value to be added to the 'or' logic, so if any of the three values meets the min/max criteria, the cell survives. When using 'Combined' with the 'Forward and reverse' or the 'Available data' options, the 'Combined' value becomes another value to be added to the 'and' logic, and all must pass the min/max filter or the cell fails and gets grayed-out.

9.5.2.4 The Variant Status Filter

The "Variant Status" filter is presented as a drop down menu, and its action is cumulative with that of the other filters, described above. The choices available are "All", "Accepted or Putative", "Accepted", "Putative", and "Rejected" (Figure 9–54). Rows that do not meet the Variant Status criteria get grayed-out and moved to bottom of the Table. This filter can be useful as part of a workflow process for user verification of the data (section 9.5.2.7).

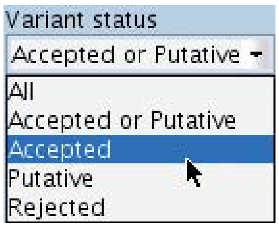


Figure 9–54: The Variant Status filter drop down menu

9.5.2.5 The Compact Table Checkbox

The 'Compact table' checkbox is used to temporarily hide columns or rows that are entirely grayed-out due to a combination of lack of data, the min/max and Variant Status filter criteria, and individual row and column 'ignore' filter settings. This gives a less cluttered view of the data that is a better candidate for the snapshot or spreadsheet options. You can uncheck the 'Compact table' option to reveal the rows/columns that have been hidden. In a large project that has accumulated many Samples and Variants, the judicious use of filters combined with the Compact table checkbox allow you to focus the table contents onto a meaningful domain, *e.g.* for preparing a report.

9.5.2.6 The Auto-Detected Variant Load Button

The Auto-Detected Variant Load button is the last control at the bottom of the Variant Display Control box. When you run a computation on the Project, the AVA software attempts to automatically detect potential substitution (SNP) and block deletion variations in the Samples that are processed (see section 9.4). It then creates a list of Variant definitions (without duplicating any Variants already saved in the Project). Click the Load button to load this queued list of potential Variants into the Project.

The Load button obeys all the other filters in the Variant Display Control box except the Variant Status filter. The Min/Max filter values are inclusive, so if the Min is set to zero and the Max is set to 100, pressing the Load button would accept all the Auto-Detected Variants surviving the other filters.

The number of unloaded Variants that meet the collective filter criteria is indicated on the right-hand side of the Load button (Figure 9–55**A**). If no Auto-Detected Variants that meet the current filter criteria are in the queue (*e.g.* Variants that did may have alredy been loaded), the button is grayed out and the text to the right of the button states this (Figure 9–55**B**). If the Project was last computed with a version of the AVA software that did not support the Auto-Detected Variants feature (versions 1.1.01 and prior), the button is also grayed out and the text to the right states that the Project must be recomputed Figure 9–55**C**).

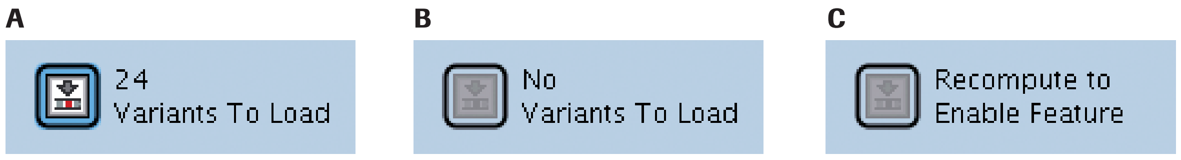


Figure 9–55: The "Variants Load" button in its 3 states: (A) Twenty-four detected Variants currently meet the Variants tab filter settings, but are not yet loaded into the Project; (B) all Variants that currently meet the Variants tab filter settings (if any) have already been loaded into the Project; and (C) the current Project was computed using an earlier version of the AVA software that did not support Auto-Detected Variants (the Project must be recomputed with the current software in order to enable the Load button).

If the number of Variants to load is high, you can do a partial load by making the filter criteria more restrictive and then deal with the remainder later. For instance, you might set the filters to include Variants with a minimum frequency of 5% with support in both the Forward and Reverse Reads and press the Load button. This would give you the subset of the Variants most likely to hold up to scrutiny. A default Status of "Putative" is assigned to all the Auto-Detected Variants that get loaded into the Project.

Note that the filters in the Variant Display Control box select for Variants that meet the chosen criteria in *some* Sample, not ones that meet the criteria in *all* Samples. Thus, even with a minimum frequency setting of 5%, a Variant will be loaded if it appears at 5% or greater in one Sample, even if it is not observed at all in any of the other Samples.

No progress indicatorappears when Auto-Detected Variants are being loaded. If the filters are set to liberal values such that a very large number of Variants are being loaded, the interface may appear to pause for few seconds during the loading process.

The loading of Auto-Detected Variants into the system is not permanent until you click the "Save" button for the Project. If you close your Project after a Variant load without clicking "Save", the Auto-Detected Variants won't be lost, but they will move back into the queue of the Load button and be available for import again when you reopen the Project. The full set of Auto-Detected Variants that don't have the same pattern as any existing Project Variant is updated every time the Project is computed. In addition, the Load queue is maintained when the Project or the AVA application are closed, so it is not necessary to immediately load Variants after a computation completes.

9.5.2.7 Variant Discovery Workflow

The AVA software provides features that allow you to manage a Discovery Workflow, which will help you identify and evaluate meaningful variations. The key components of this process are the ability to load automatically detected Variants; the ability use a right-click menu to easily set the Status of one or more Variants when rows have been selected in the Variants Frequency Table of the main Variants tab; and the ability to filter the content of the Variants Frequency Table based on Variant Status.

This constellation of features allows the main Variants tab to be the hub of operations for Discovery Workflow. One can choose to load Auto-Detected Variants into the Project with the click of a single "Load" button on the Variants tab. If the volume of Variants that are available to load, as displayed to the right of the "Load" button, is large, you can prevent Project clutter by selecting certain Variants to load, via the filters associated with the Variants Frequency Table. For example, you could choose settings such as "Consensus" for "Alignment Read Type", with a Min/Max of 5.00%/100.00% applied to "Forward and reverse" reads. This would allow you to load the subset of Auto-Detected Variants most likely to withstand scrutiny.

For Discovery Workflow purposes the status options have the following intended meanings:

- Accepted – a Variant that is expected to be found in at least one Sample; or a Variant that has previously been found and verified by a user
- Putative – a Variant that is known from the literature but may or may not be found in a Project Sample; or a Variant that has been automatically computed but not verified by a user
- Rejected – a Variant that has been flagged as invalid because a user has determined that it was detected due to some type of artifact, such as from Sample processing or an alignment problem.

Despite these definitions, the AVA software simply treats the Variant Status as a tag that can be used for data filtering. Thus, you can choose to interpret the status values differently, if necessary, to better meet your needs.

Loaded Auto-Detected Variants are automatically assigned an initial Status of "Putative". Any other Variants that are manually defined (section 9.3.2.5.2) or declared via filter selections on Global and Consensus alignments (see sections 9.6.3.3 and 9.7.3) will have defaulted to a Status of "Accepted". Presuming that "Accepted" Variants have already been validated, one can set the "Variant Status" filter of the Variants Frequency Table to "Putative" and click on the "Compact Table" box. This causes any "Accepted" or "Rejected" Variant rows to be grayed-out and demoted to the bottom of the table. The "Compact Table" option then hides those rows so that the only visible rows are those that have a Status of "Putative".

With a Project set up as above, one can begin validating the "Putative" Variants. If you right-click on a Sample-Variant intersection cell in the Table, you can use the "Global Align" link to load the Global Align tab with the alignment of Read Consensi that cover the region of the corresponding Variant. After exploring the underlying alignments and flowgrams to determine if the Variant appears to be legitimate, one can return to the Variants tab and change the Status of the "Putative" Variant to either "Accepted" or "Rejected". This is done via the "Variant Status" submenu that appears when right-clicking on a Sample-Variant intersection cell. Once the Status has been changed from "Putative", the Variant row will no longer meet the "Variant Status" filter that applies to the table; since the "Compact Table" option is active, the row will automatically be hidden. Variants judged as invalid should be marked as "Rejected" rather than deleted entirely. This will both prevent the Variant from being added back to the Load queue, and prevent the automatic Variant detection mechanism from re-proposing the same Variant after the completion of the next computation cycle (which would force you to re-evaluate this Variant each time Variants are loaded).

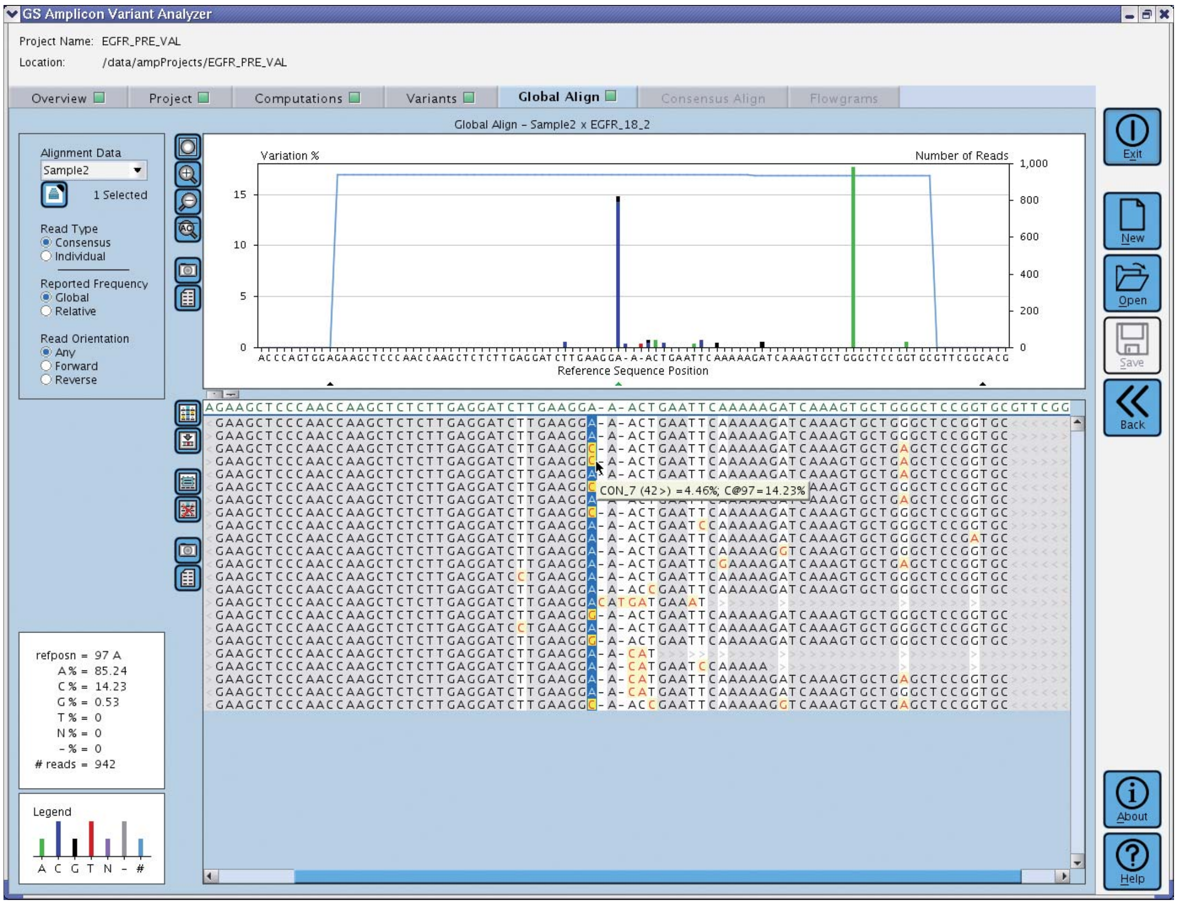
This method provides a shrinking pool of "Putative" Variants to work with. Eventually, after all Variants have been evaluated, the Table will be empty. If one starts the process with a partial Variant Load, you can use looser filter settings for the Variants Frequency Table to see if the "Load" button indicates any additional Variants to load. For instance, one might keep the current filters but just change the "Forward and reverse" option to "Available data". This would pick up Variants in regions of Amplicons that have coverage only from reads of a single orientation. Or one might try switching the "Alignment Read Type" from "Consensus" to "Individual" to catch any cases where variations are hidden because they were distributed over several Consensi at low enough levels that their changes were not incorporated into the Consensus sequences. By manipulating the filters in such a way, one can fine-tune a new set of "Putative" Variants to load and then load and validate them as before.

Eventually, the filters will be set to as restrictive values as one is willing to go, or to where there are no Variants left to load. To determine if there are any available Variants at all, one can choose the loosest settings, *i.e.* with "Alignment Read Type" set to "Individual", Min/Max set to 0.00%/100.00%, and with the Min/Max applied to "Forward or reverse" reads. At any point, one can reset the "Variant Status" filter to see "All" the Variants, just those "Accepted", just those "Accepted or Putative", or even those "Rejected", and then later switch back to the "Putative" only view to continue working through the list of Putative Variants.

Keep in mind that the automatic Variant detection does not currently report insertions or monomer deletions shorter than 3 bases (one must manually define those types of Variants if their statistics are to appear in the Variants Frequency Table). Also, remember that a Variant load and any changing of Variant Status are not permanent until the Project is saved. This can provide a useful form of "undo" if a Project is accidentally cluttered with an errant load. In such a case, simply re-open the Project without a "Save" in order to restore the Project's previous list of Variants.

9.6 The Global Align Tab

The Global Align tab allows you to view the underlying alignment information that is used in the calculation of Variant frequencies. It is divided into two results panels (Figure 9–56): the top panel contains a stacked histogram / depth of coverage plot of all the variations observed between the reads included in the last computation and their Reference Sequence; while the bottom panel contains the multiple alignment of those reads to the Reference Sequence. In addition, a set of display option tools (for data navigation and filtering) as well as a "Mouse Tracker" display (section 9.1.3.3.3) and color legend for the Variation Frequency Plot are available on the left-hand side of the tab. The Global Align tab can only display data for one Sample-Reference Sequence combination at a time, but it can display data for multiple Amplicons together, provided that they are all associated with both this Sample and this Reference Sequence.



Project Name: EGFR_PRE_VAL
Location: /data/ampProjects/EGFR_PRE_VAL

Alignment Data: Sample2
1 Selected

Read Type: [✓ Consensus] Individual
Reported Frequency: [✓ Global] Relative
Read Orientation: [✓ Any] Forward Reverse

```
refposn = 97 A
    A % = 85.24
    C % = 14.23
    G % = 0.53
    T % = 0
    N % = 0
    - % = 0
# reads = 942
```

```
ACCCAGTGGAGAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-ACTGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGCGTTCGGCACG
```

```
AGAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-ACTGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGCGTTCGG
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-ACTGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-ACTGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGC-A-ACTGAATTCAAAAAGATCAAAGTGCTGAGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGC-A-ACTGAATTCAAAAAGATCAAAGTGCTGAGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-ACTGAATTCAAAAAGATCAAAGTGCTGAGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGC CON_7 (42>) =4.46%; C@97=14.23% AAAGTGCTGGGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA                                AAAGTGCTGAGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGC-A-ACTGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-ACTGAATCCAAAAAGATCAAAGTGCTGGGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-ACTGAATTCAAAAAGATCAAAGTGCTGGGCTCCGATGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-ACTGAATTCAAAAAGGTCAAAGTGCTGGGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-ACTGAATTCGAAAAGATCAAAGTGCTGAGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCCTGAAGGA-A-ACTGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-ACCGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGACATGATGAAAT
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGG-A-ACTGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCCTGAAGGA-A-ACTGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGG-A-ACTGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-CAT
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-CATGAATCCAAAAA
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-CATGAATTCAAAAAGATCAAAGTGCTGAGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-CATGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGC
 GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGC-A-ACCGAATTCAAAAAGGTCAAAGTGCTGAGCTCCGGTGC
```

Figure 9–56: The Global Align tab

9.6.1 Populating the Global Align Tab

When you open an Amplicon Project in the AVA software, the Global Align tab has no content and is grayed-out. To populate it from this state, you must use a "Global Align" action from one of the following two sources:

- A Sample-Amplicon pair, from any of the 3 Project Tree views, on the Project tab [see sections 9.3.1.1, 9.3.1.2, and 9.3.1.3; note that the Amplicon must be fully defined (section 9.3.2.2) and the computation must have been carried out (section 9.4)]. Right-clicking on a Sample (in the References Tree) or an Amplicon (in the Read Data Tree or the Samples Tree) opens a contextual menu that includes a "Global Align" option; choosing this will populate the Global Align tab with the multi-alignment of the reads for the Sample-Amplicon pair on that branch of the tree.
- A Sample-Variant pair, from the Variants Table, on the Variants tab [see section 9.5.1.3; note that the Variant must be fully defined (section 9.3.2.5) and the computation must have been carried out (section 9.4)]. Right-clicking on an appropriate cell of the Variants Table opens a contextual menu that includes a "Global Align" option; choosing this will populate the Global Align tab with the multi-alignment of the reads of all the Amplicons that cover this Variant on the Reference Sequence and that are associated with this Sample.

Once the Global Align tab is populated, you can still use the right-click method above, and replace the multi-alignment displayed with another one. But from inside the Global Align tab, you have another, more powerful option: this tab has two "Alignment data" controls in its upper-left corner that allow you to browse through and navigate all the alignments of your Project without leaving this tab, and even to view the data for multiple Amplicons together (for a given Sample-Reference Sequence pair). These controls are described in detail in section 9.6.4.1.

9.6.2 The Variation Frequency Plot

The Variation Frequency Plot (Figure 9–57) located in the top panel of the Global Align tab shows graphically (a) all the variations (relative to the Reference Sequence) that were observed in the reads included in the last computation and associated with the Sample-Reference Sequence combination and the Amplicon(s) selected, and (b) the depth of coverage at each position of the multi-alignment.

- The horizontal axis represents the Reference Sequence, gapped as needed to accommodate any insertions in the reads.
- The left vertical axis shows the percentage frequency of the variations, whereby individual variants for each position of the Reference Sequence are represented as colored bars keyed to the legend at the bottom-left of the tab; if more than one variation occurs at any given position, the bars are stacked vertically.
- The right vertical axis shows the depth of coverage for each Reference Sequence position, in number of reads; this data is shown on the plot as a light blue line.

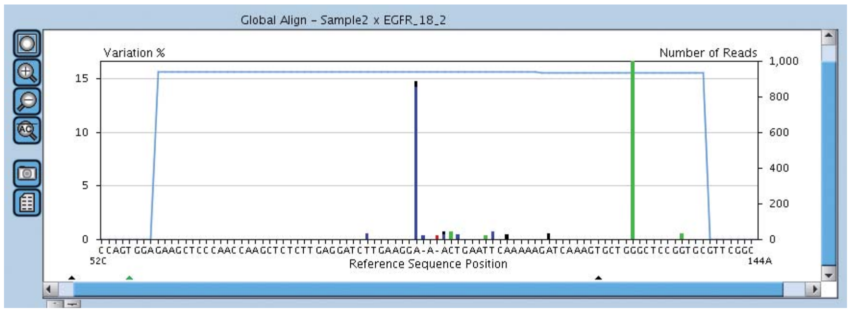

Global Align - Sample2 x EGFR_18_2
Variation %
Number of Reads
15
10
5
0
1,000
800
600
400
200
0
CCAGTGGAGAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGA-A-ACTGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGCGTTCGGC
52C
144A
Reference Sequence Position

Figure 9–57: The Variation Frequency Plot. In this example, the plot was zoomed in to show the Reference Sequence along the horizontal axis.

The plot has all the standard navigation features (scrollbars, zoom buttons, mouse tracker, *etc.*) described in section 9.1.3.3. In addition, the Variation Frequency Plot and the multiple alignment of the bottom panel are reciprocally linked, in the following ways:

- Three small triangles (two black and one green) located at the bottom of the plot panel have the following meanings:
  - The black triangles indicate the boundaries for the subset of the plot that is visible in the multi-alignment panel.
  - The green triangle shows the position in the plot that corresponds to the highlighted nucleotide in the alignment.
- Clicking in either the plot or the multi-alignment panel centers the other panel on the position clicked.
  - If you clicked on the plot, the nucleotide column in the multi-alignment is also highlighted.
  - If you clicked in the multi-alignment, the position of the tracking triangles on the Variation Frequency Plot is also adjusted accordingly.

9.6.3 The Multiple Alignment Display

The multiple alignment display (Figure 9–58) located in the bottom panel of the Global Align tab shows the alignment of all the selected reads to the Reference Sequence. These reads may be grouped into consensi and/or selected for certain observed variations (see below). Scrollbars appear when necessary, and the Mouse Tracker and Screen Tip features (9.1.3.3.3) are also active in the multiple alignment display.

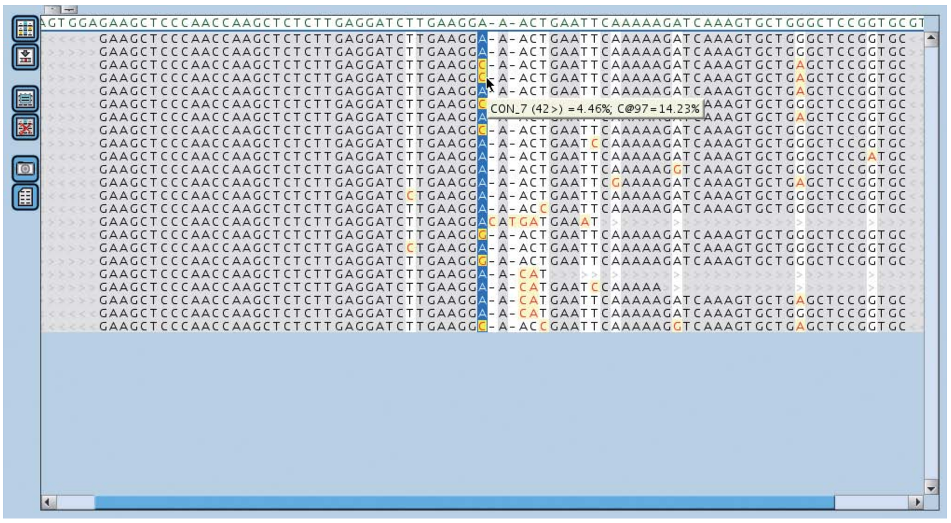


Figure 9–58: The multiple alignment display of the Global Align tab

9.6.3.1 The Reference Sequence

The Reference Sequence runs along the top strip of the multiple alignment displayand is shown in green characters. Pausing the mouse over the Reference Sequence displays a screen tip showing the position of the nucleotide under the pointer.

Gaps (‘-’) in the Reference Sequence indicate virtual positions where one or more aligned reads have insertions. These insertion positions are numbered with decimals based on the Reference Sequence position to the left of the insertion (*e.g.* a two nucleotide insertion between Reference Sequence positions 362 and 363 will be labeled positions 362.1 and 362.2). Do not confuse the decimal positions used in the multiple alignments and those used for specifying insertions in Variant Patterns (section 9.3.2.5.2). In Variant Patterns, the location of an insertion of any length between two specific Reference Sequence positions *p* and *p+1* is always designated “*p.5*”. In a multiple alignment, by contrast, each inserted nucleotide is given its own “virtual” decimal position identifier.

Note that only gaps that correspond to inserted nucleotides *in the reads or consensi displayed in the multi-alignment below* are shown in the Reference Sequence. If you apply selection(s) to the reads or consensi (using a right-click ‘Select’ option or the ‘Assemble consistent reads’ button; see section 9.6.3.2, below), some of the inserted nucleotides may not be represented in the remaining reads; those gaps will not be shown in the Reference Sequence. However, their decimal coordinate number will be maintained in the alignment, such that the decimal number of the gaps displayed may not always be consecutive. (This also applies to the display of the reads from a single consensus on the Consensus Align tab, which is another form of read selection; see section 9.7.)

9.6.3.2 The Multi-Alignment

Below the Reference Sequence are the aligned reads (or consensi). Initially, the 'Read Type' display control (from the display option tools in the upper-left corner of the tab) is set to 'Consensus', whereby the aligned reads are grouped into consensus representations of reads which are substantially similar to each other. If 'Individual' is chosen instead, all the individual underlying reads are displayed. The Read Type setting is maintained within the session as you navigate from alignment to alignment. See section 9.6.4.2 for a more complete description of these display options.

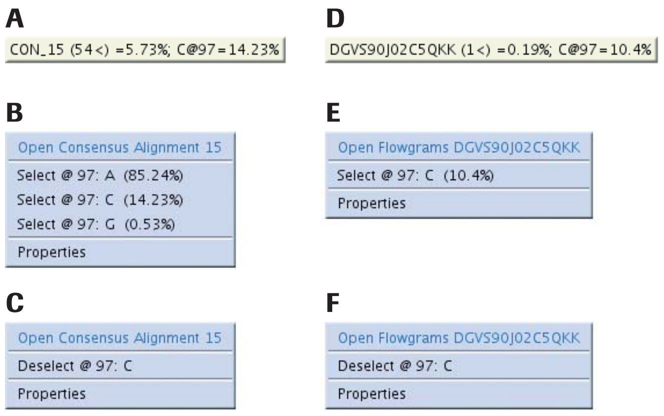
The multiple alignment display has the following functions and features:

- If a read or consensus doesn't start at the first position or end at the last position of the alignment, its beginning and ending will be filled with light gray '>' or '<' characters. These characters are indicators that the sequencing reads are 'forward' or 'reverse', respectively, relative to the Reference Sequence.
- The background color of the alignment columns provide an at-a-glance way to focus on the positions that may be of most interest:
  - Gray columns are tagged as uninteresting because all the reads or consensi match the Reference Sequence at that position.
  - White columns, by contrast, contain at least one read or consensus that differs from the Reference Sequence and are thus worthy of attention.
- In the white alignment columns, the specific nucleotides that do not match the Reference Sequence are shown as eye-catching red letters on yellow background, while the matching ones are black on white.
- Pausing the mouse over a nucleotide in the multi-alignment displays a screen tip (Figure 9–59 **A,D**) that provides:
  - the name of the read or consensus
  - the number of reads represented in the consensus (always 1 if 'Read Type' is set to 'Individual'; see section 9.6.4.2) and its orientation
  - frequency information (subject to the 'Global' or 'Relative' selection made in the 'Reported Frequency' tool; see section 9.6.4.3), as follows:
    - the proportion (as %) of the reads represented by this read or consensus
      - if the 'Read Type' control is set to 'Consensus', the consensi are sorted in decreasing order of the number of reads they comprise
      - if the 'Read Type' control is set to 'Individual', obviously, all the reads will have the same frequency
    - the proportion (as %) of the reads that have this nucleotide at this position
- Left-clicking on a nucleotide in the Reference Sequence or any of the aligned sequences highlights the column. The highlighting switches reference-matching nucleotides to white lettering with a dark blue background; mismatches remain as red letters, but their background switches to yellow with blue left and right edges. As seen above (section 9.6.2), this action also centers the Variation Frequency Plot on the coordinate clicked, and places the green tracking triangle underneath it.
- Right-clicking on a nucleotide in the multi-alignment display opens a contextual menu like the one shown in Figure 9–59**B,E**.
  - The first option will depend on the setting of the 'Read Type' control:
    - If set to 'Consensus', the first item will be 'Open Consensus Alignment' (Figure 9–59**B**); this action will take you to the Consensus Align tab (see section 9.7) and populate it with the multi-alignment of the reads that are included in the consensus on which you clicked.
    - If set to 'Individual', the first item will be 'Open Flowgrams' (Figure 9–59**E**); this action will take you to the Flowgrams tab (see section 9.8) and populate it with the tri-flowgram corresponding to the read on which you clicked and focused on the flow corresponding to the base on which you clicked.

- The option at the bottom of the menu, called "Properties" pops up a new window displaying specific sequence information of the consensus or read on which you right-clicked. The data is presented in a format that allows you to copy information to the clipboard so you can export it to external programs for further analysis. For further details about the data available in the properties menus and for suggestions on how the data can be useful, see section 12.3.
- The rest of the options allow you to restrict the display to the reads or consensi that contain a specific selected nucleotide (or a gap) at the position on which you clicked. When you make a selection, the corresponding nucleotide in the Reference Sequence is highlighted with a cyan background color, and all reads that do not contain the selection are hidden from view. You can make multiple successive selections in a multiple alignment, at one or more positions, further restricting the number of reads or consensi displayed at each selection. This can be useful to explore linkage between variations in the read data.

  If following your selections a given position consists only of gaps (including that position in the Reference Sequence), this gap position will be removed from display; this results in a more compact and more readily understandable alignment. Because of this collapsing of gapped columns, the decimal "virtual" positions in the Reference Sequence, while always increasing, may not always be consecutive in a "selected" multi-alignment display (see section 9.6.3.1 for more details on decimal position numbering in a gapped alignment). This also applies to the display of the reads from a single consensus on the Consensus Align tab (another form of read selection) whose view features these same selection tools (see section 9.7).

- If you right-click on a nucleotide in the multi-alignment display at a position that is already the object of a selection, a contextual menu like the one shown in Figure 9–59**C**,**F** will open.
  - The first option is the same as what is seen when no selection is active
  - The Properties option also is the same as what is seen when no selection is active
  - The second (middle) option indicates the currently active selection(s). If a selection is deactivated, all reads currently hidden by the selection will be reintroduced into the visible multi-alignment and the cyan highlight at the top of the alignment column will be removed. Selections may be removed from an alignment in any order, regardless of the order in which they were added to the alignment.


A
CON_15 (54<) =5.73%; C@97=14.23%

B
Open Consensus Alignment 15
Select @ 97: A (85.24%)
Select @ 97: C (14.23%)
Select @ 97: G (0.53%)
Properties

C
Open Consensus Alignment 15
Deselect @ 97: C
Properties

D
DGVS90J02C5QKK (1<) =0.19%; C@97=10.4%

E
Open Flowgrams DGVS90J02C5QKK
Select @ 97: C (10.4%)
Properties

F
Open Flowgrams DGVS90J02C5QKK
Deselect @ 97: C
Properties

Figure 9–59: Screen tips and contextual menus that can appear in Global Align and Consensus Align tabs. The manus on the left (A-C) are only seen in the Global Align Tab when "Read Type" is set to "Consensus". The menus on the right (D-F) are seen in the Consensus Align tab or in the Global Align Tab when "Read Type" is set to "Individual". (A, D) The screen tip displayed when you pause the mouse over a nucleotide in the multi-alignment. (B, E) The contextual menu that opens when you right-click on a nucleotide in the multi-alignment. (C, F) The contextual menu that opens when you right-click on a nucleotide in a multi-alignment that is already the object of a selection.

9.6.3.3 Special Function Buttons

Various advanced functions can be carried out using the special function buttons located to the left of the multiple alignment. These can help you explore and exploit the reads or consensi displayed in the multi-alignment, in order to identify (or "declare") variations you believe to be valid Variants (see section 10.4 for guidelines and factors to consider when trying to determine whether a Variant is genuine).

Deselect menu – Every time you use a right-click 'Select' option on a nucleotide in the multiple alignment (and also when you use the 'Assemble consistent reads' function, below), your selections are added to a list. Clicking this button opens the "Remove Selections" window showing the list of selections, sorted by reference position, and allows you to remove any or all of the selections (Figure 9–60). As the selections are removed, the sequences hidden by those selections will be added back to the multiple alignment.

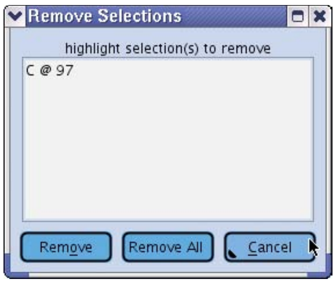


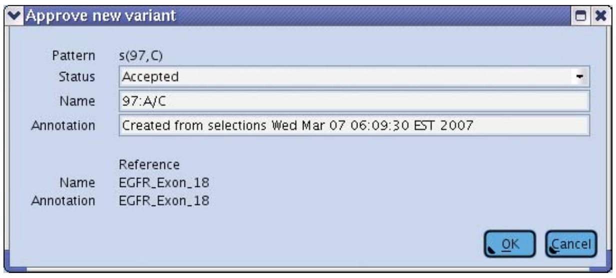
Figure 9–60: The Remove Selections window

The Deselect function is critical to removing selections that correspond to gaps in the Reference Sequence. If some reads contained an insertion relative to the Reference Sequence, and the user selects a '-' at that position, then the reads with the insertion would be eliminated from the display. At that time, the multiple alignment of the remaining reads would all have a gap character at that position. Since the AVA software automatically collapses any columns that consist entirely of gaps, that gap position of the alignment would be removed from view and there would be no cyan indicator at the top of the alignment to indicate that the '-' selection was performed. The only way to remove the '-' selection at that point would be through the Deselect menu.

Declare project variant – Clicking this button takes the current set of 'Select' choices you made on the multi-alignment, and converts it into a new Variant on the Variants sub-tab of the Project tab; the next computation will search for this new Variant, and the frequency results will be reported in the Variants tab. An "Approve new variant" window (Figure 9–61) will first open and show how your 'Select' choices have been converted into a valid 'Pattern' compatible with the Variant scanning function; the 'Reference' Sequence will have been determined based on the alignment you were viewing. The window also has fields in which you can select a Status for the Variant from a drop down (defaults to Accepted) and in which you can enter a Name and an Annotation for the new Variant, before accepting it. If the Pattern is equivalent to that of a Variant already defined in the Project, the window will display a warning to help prevent the incorporation of a redundant Variant.

Keep in mind that during Variant scanning, a read must overlap all the positions involved in the Variant to qualify as containing the Variant, so 'Select' choices should be as compact and succinct as possible before declaring a Variant. Note also that there must be at least one 'Select' choice made prior to clicking the 'Declare project variant' button or the 'Approve new variant' window will not open. In some cases, the current selections may be close to, but not exactly, the Variant Pattern you want to use to define the new Variant. The "Approve new variant" window does not, however, let you edit the synthesized Variant Pattern. In this case, you should approve the addition of the Variant to the Project and subsequently edit it in the Variants sub-tab of the Project tab (section 9.3.2.5).

Since possible Variants are automatically proposed by the AVA software, potentially in quite large numbers, the software attempts to provide meaningful but unique default names (see section 12.2). This also applies to Variants declared manually, via the Approve New Variant window.



Approve new variant

Pattern	s(97,C)
Status	Accepted
Name	97:A/C
Annotation	Created from selections Wed Mar 07 06:09:30 EST 2007
	Reference
Name	EGFR_Exon_18
Annotation	EGFR_Exon_18

Figure 9–61: The Approve new variant window.

Assemble consistent reads – This button provides a means of mining for consistent patterns within the sequences displayed in the multiple alignment. “Consistent reads” means reads that are identical in the portion over which they overlap (*i.e.* overhanging nucleotides due to reads of different lengths do not penalize the “consistency”). This is more useful when the ‘Read Type’ is set to ‘Individual’ as opposed to ‘Consensus’ (in which case the consensus process has already gathered up similar reads). Using the reads on display in the multiple alignment as input (but not those already hidden away by ‘Select’ choices), the assembly process makes a set of automated ‘Select’ choices to identify sets of consistent reads for display.

This is typically used in conjunction with the ‘Remove reads’ button discussed next, as a means to recursively mine for patterns in the alignment. As you identify patterns in the sequence variations, you can discard those sequences temporarily and search through the remaining sequences for additional variation patterns. There isn’t a direct undo operation after a round of assembly, but you can use the ‘Remove all’ option of the ‘Deselect menu’ button (see above) to wipe out the selections made by the assembly process (along with any other selections you may have made prior to the assembly).

When you normally add selections to alignment positions, only those reads that explicitly have the selected nucleotide (or gap) remain visible; in particular, reads that do not extend all the way to the selected column are not given the benefit of the doubt that they might have matched that position, and are therefore hidden from view. Since the “Assemble consistent reads” action only requires agreement in the areas of overlap, the automatically generated ‘Select’ choices behave differently and allow reads to remain so long as they are consistent with the selections in the columns for which the read has coverage. Following an Assembly operation, the selection mechanism stays in this more “inclusive” mode even for additional selections or de-selections made via a right-click by the user. The selection mode is made clear to the user by having an ‘Inclusive Select’ rather than simply a ‘Select’ menu item appear in the right-click contextual menu. The AVA software reverts to the original non-inclusive behavior as soon as all the selections have been removed (either via the “Deselect menu” or by using the “Remove reads, reselect selections” button described next).

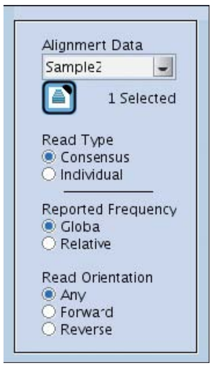
Remove reads, reset selections – Clicking this button takes all the displayed sequences in the currently displayed multiple alignment, and discards them from memory. The full set of alignment position filters is cleared, and any remaining sequences that were hidden by prior selections are revealed in the alignment. This button is typically used in conjunction with the 'Assemble consistent reads' button described above, as a means to recursively mine for patterns in the alignment. Once you click this button you cannot undo the sequence discard, but the sequences are only discarded from memory and not from the underlying multiple alignment. Reopening the global alignment via the Project Trees or the Variants tab, or using the "Alignment data" display control tools (section 9.6.4.1), will reload the original alignment and restore the full complement of reads or consensi. (Similarly, if you are in the Consensus Align tab, you can restore the full complement of reads by returning to the Global Align tab and re-loading the same consensus; see section 9.7.)

Save table snapshot to a .png file – This button saves the visible portion of the currently displayed multiple alignment, as a PNG image file. A file browser window will open, allowing you to assign a name and a destination for the file.

Save the alignment as an ACE file – This button takes the currently displayed multiple alignment and writes it out as an ACE format file so you can import it into a suitable third-party application. A file browser window will open, allowing you to assign a name and a destination for the file. You should maintain the '.ace' suffix on the file name since some applications will expect it when importing the file.

9.6.4 Display Option Tools

The upper-left corner of the Global Align tab contains various navigation and filtering tools that modify the information displayed on the Variation Frequency Plot and the multiple alignment panels of the tab (Figure 9–62).


Alignment Data
Sample2
1 Selected

Read Type
(●) Consensus
( ) Individual

Reported Frequency
(●) Global
( ) Relative

Read Orientation
(●) Any
( ) Forward
( ) Reverse

Figure 9–62: The display option tools of the Global Align tab.

9.6.4.1 Alignment Data

There are two navigation controls, located at the top of the display option tools box, that allow you to select new sets of data to display in the Global Align tab. The first is a drop down menu that contains a list of all the Samples defined in the Project that are associated with at least one of the Amplicons you are viewing in the currently displayed multiple alignment. Selecting a new Sample from the drop down menu will update the 'Global Align' tab with the alignment data for the new Sample, replacing the current data. This allows you to quickly compare various Samples over a single or a given set of Amplicon(s).

The second Alignment data control is the Amplicon selection button, located just below the 'Alignment Data' drop down menu (Figure 9–62). Clicking this button opens the "Choose Alignment Data" window (Figure 9–63).

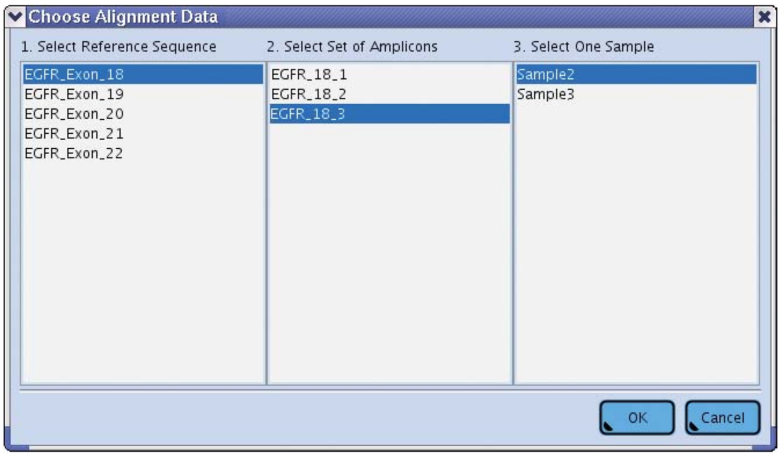


Figure 9–63: The Choose Alignment Data Window

This window allows you to browse over the entire Project and select data for display in the Global Align tab. It is used in three steps:

- Step 1: choose a Reference Sequence for which you want to display the data. This will update the list of available Amplicons, displayed in the second column, to those that are associated with that Reference Sequence and for which there is an alignment computed for at least one Sample (excluding, however, Amplicons associated with the Reference Sequence selected but for which no Read Data Sets have supplied any reads).
- Step 2: select one or more of the available Amplicon(s) of interest. The Global Align tab can display the reads from multiple Amplicons, merged into a single multi-alignment, as long as they all belong to the same Reference Sequence. This selection will update the list of eligible Samples, displayed in the third column, to those that are associated with at least one of the Amplicons selected and for which there are currently computed results.
- Step 3: select one of the eligible Samples, and click "OK" to load the selected alignment in the Global Align tab. The number of Amplicons included in the displayed multi-alignment is indicated to the right of the Amplicon selection button.

If you activate the "Choose Alignment Data" window and click "OK" without changing any of the currently selected choices, the AVA software will freshly reload the currently selected alignment. This can be useful to reset the view of the alignment after using the "Remove reads, reset selections" action (see section 9.6.3.3)

9.6.4.2 Read Type

The 'Read Type' radio buttons (see Figure 9–62) give you the choice to display the reads in one of two ways:

- Consensus: This is the default read type. When this option is selected, reads that are substantially similar are grouped together into consensi, which results in fewer sequence entries in the multi-alignment table. The consensi are listed in the multi-alignment in decreasing order of the number of reads they represent, so the ones near the top have more "weight". This option reduces noise and speeds up navigation.
- Individual: This option displays every read as a separate sequence line in the multi-alignment, even if they are identical to other reads. This can greatly increase the volume of alignment lines and slow navigation, but hides no noise.

It is usually easiest to perform an initial analysis with the default Consensus view, with its lower volume and decreased noise. Delving into the individual reads can be useful if you need to search for a particular variation that may have been erroneously spread amongst several consensi and treated as noise in basecalling, rather than being exposed as a separate variation.

9.6.4.3 Reported Frequency

The next set of radio buttons controls the type of 'Reported Frequency' (see Figure 9–62). This applies to:

- the Variation Frequency Plot left axis (which relates to the histogram bars)
- the information reported in the mouse tracker panel
- the frequency information for a nucleotide, in the screen tips that appear when you pause the mouse over a nucleotide in the multi-alignment panel
- the frequency information for the nucleotide selection options, in the contextual menu that appears when you right-click on a nucleotide in the multi-alignment panel
- the reported read depth in all of the above

The two options are as follows:

- The default 'Global' frequency option uses the coverage from the full data set as the read depth denominator when calculating the frequency of occurrence of a given nucleotide at a given alignment position, regardless of any positional 'Select' filters that may have been applied to hide reads (or consensi) from the multi-alignment display. (This also applies to the display of the reads from a single consensus on the Consensus Align tab, which is another form of read selection; see section 9.7.)
- The 'Relative' option recalculates frequencies using only the visible data, *i.e.* ignoring reads (or consensi) hidden from the multi-alignment display after you applied any selection(s). This can be useful when you have selected reads (or consensi) for a variation at a given coordinate and you want to examine other variations relative to the first selection(s) (now set at 100%); for example, variations linked as a haplotype should show near 100% relative frequency in this situation. (This is also useful when examining the reads from a single consensus on the Consensus Align tab, which is another form of read selection; see section 9.7.)

If you did not make any 'Select' filter choices on alignment positions, the 'Global' and 'Relative' frequencies will be the same (but not so on the Consensus Align tab, where all results displayed are inherently selected for a single consensus; see section 9.7). Once you make selections to focus on a subset of the data, you will notice the difference between the reported frequency types. Note that the reported read depth is that used in the frequency calculations. So, if you want to know how many reads are present amongst the selected reads of the multiple alignment, you must switch to 'Relative' frequencies.

9.6.4.4 Read Orientation

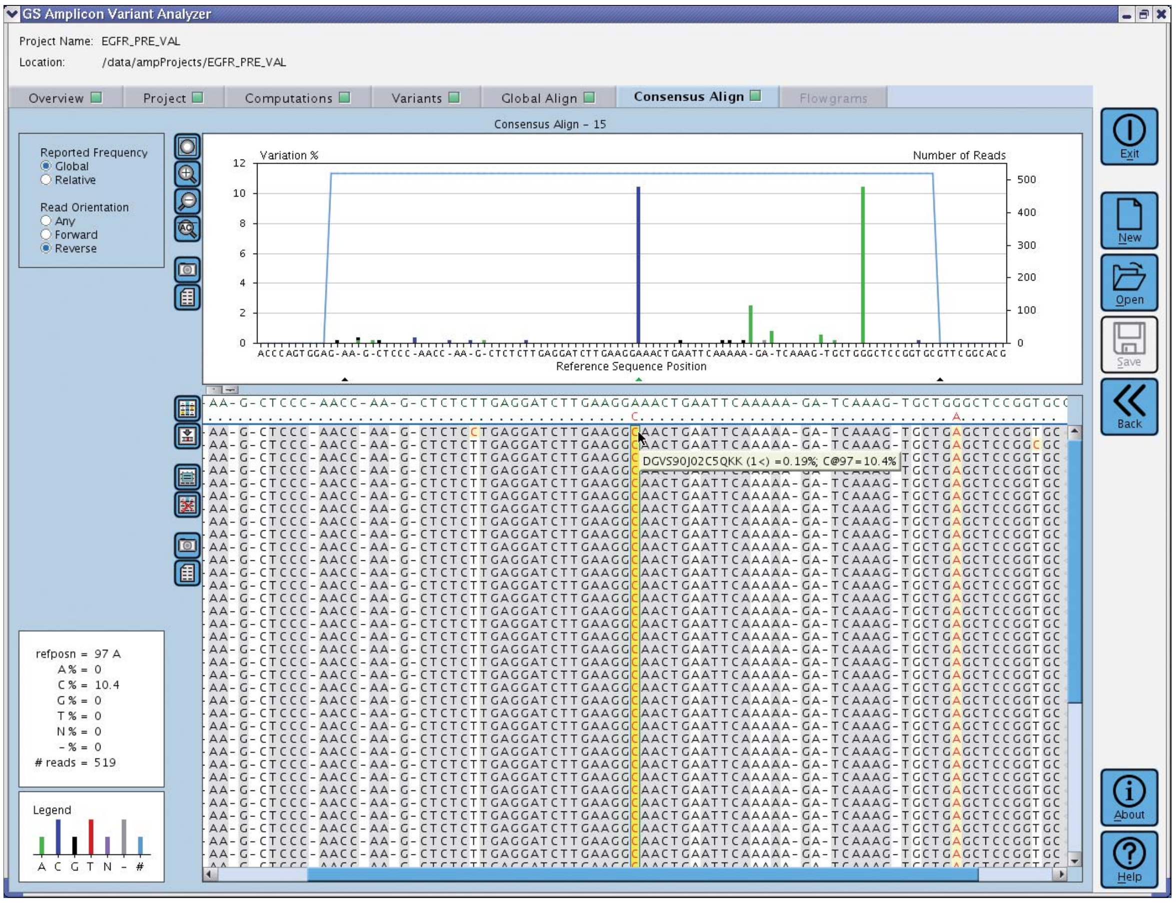
The final set of radio buttons controls the display of the reads or consensi by 'Read Orientation' (see Figure 9–62).

- The default is 'Any' which means both forward and reverse reads are presented in the multi-alignment.
- Choosing 'Forward' or 'Reverse' will restrict the multi-alignment view to display only the reads of the selected orientation. This can be useful when you have coverage of a variation in reads from both orientations; the presence of the variation at a similar frequency in both orientations would be a strong argument that it is "real", whereas its presence in only one orientation (or a large discrepancy in its frequency between the two orientations) would be an indication that the variation might be due to an artifact. (See section 10.5 for guidelines and factors to consider when trying to determine whether a Variant is genuine).

In a restricted orientation view, the behavior of the 'Global' reported frequency option is slightly modified so that the coverage in the denominator still comes from the full data set, but is restricted by orientation. This prevents the global frequencies from deceptively dropping by about 50% when an orientation is chosen. As mentioned in the description of the Variants tab (section 9.5.2.2), failure of a variation to show in one orientation for which a good number of reads are available is an indication of a possible artifact; however, you may want to avoid "penalizing" a variation on this account if it is covered in only one orientation, or if too few reads cover it in one orientation to provide for statistically valid data.

9.7 The Consensus Align Tab

The Consensus Align tab (Figure 9–64) is useful when you need to see the individual reads comprised in a consensus (from the Global Align tab), to evaluate the variations that the software has considered "noise". This removal of "noise" is what allows the Global Align tab to simplify the display of the Project's data by collapsing groups of similar reads into consensi, when its Read Type control is set to "Consensus" (see section 9.6.4.2). The Consensus Align tab, thus, is very similar to the Global Align tab, except that it displays the multi-alignment of the individual reads that comprised a consensus on the Global Align tab.


GS Amplicon Variant Analyzer

Project Name: EGFR_PRE_VAL
Location: /data/ampProjects/EGFR_PRE_VAL

Consensus Align – 15

Reported Frequency
Global [✓]
Relative

Read Orientation
Any
Forward
Reverse [✓]

DGVS90J02C5QKK (1<) =0.19%; C@97=10.4%

refposn = 97 A
A % = 0
C % = 10.4
G % = 0
T % = 0
N % = 0
- % = 0
reads = 519

Figure 9–64: The Consensus Align tab

9.7.1 Populating the Consensus Align Tab

When you open an Amplicon Project in the AVA software, the Consensus Align tab has no content and is grayed-out. To populate it, go to the Global Align tab, make sure that its Read Type control is set to "Consensus", and right-click on the consensus whose reads you want to explore in detail. A contextual menu will appear which will include an "Open Consensus" option. Selecting this option will populate the Consensus Align tab with the multi-alignment of the reads that are grouped in the consensus on which you right-clicked.

Since the purpose of the Consensus Align tab is to "drill down" on the reads of a given consensus, as opposed to viewing data from the whole Project, it lacks the "Alignment data" controls that allow you to browse through all the alignments, on the Global Align tab (see section 9.6.4.1). Also, since it always displays individual reads, it lacks the "Read Type" controls as well, that allow you to choose to display consensi or individual reads, on the Global Align tab (see section 9.6.4.2).

9.7.2 The Variation Frequency Plot

The appearance and all features of the Variation Frequency Plot in the Consensus Align tab are identical to those of the corresponding plot in the Global Align tab, except that this one shows only data from the reads of the consensus selected for display in this tab. See section 9.6.2 for a full description of this plot's features.

9.7.3 The Multiple Alignment Display

The appearance and all features of the multiple alignment display in the Consensus Align tab are very similar to those of the corresponding display in the Global Align tab (see section 9.6.3 for a full description of this display's features). They do, however, have the following differences:

- The Consensus Align tab shows only data from the individual reads of the selected consensus. Therefore, the reads are never grouped into consensi as is possible in the Global Align tab, and there are no 'Read Type' display options.
- The consensus sequence of the aligned reads is shown just below the Reference Sequence, at the top of the panel. Matching positions are displayed as dots ".", whereas the consensus nucleotides are shown explicitly for positions that differ from the Reference Sequence.

9.7.4 Display Option Tools

This aspect of the Consensus Align tab differs from the Global Align tab in that it lacks the 'Alignment Data' and 'Read Type' controls, as these would not apply in this context. The other display controls, for 'Reported Frequency' and 'Read Orientation', are the same as in the Global Align tab (see section 9.6.4 for a full description of these display options features).

Note that consensi (in the Global Align tab) are always constructed from reads of the same orientation and from the same Amplicon; forward and reverse reads, and reads from two Amplicons, even if they overlap, will not be commingled in a single Consensus read. The system can thus automatically select the appropriate 'Read Orientation' option for the Individual reads of the Consensus at the time the Consensus Align tab is populated. This is important so that the estimated frequency of variation be correctly calculated when the Global Reported Frequency is selected, *i.e.* without including the depth of both read orientations in the denominator of the calculated frequency (see section 9.6.4.4 for more details on this).

9.8 The Flowgrams Tab

The Flowgrams tab allows you to view the flow-by-flow signals of any individual read included in the Project, highlighting any departure from the signal intensities that would be expected of the Reference Sequence to which that read (Amplicon) is associated. The display is designed to help you evaluate the significance of differences between an individual read and a Reference Sequence. To this end, the tab does not simply display the raw flowgram of the read, but rather a computationally processed version of it. In particular, flow cycle-shifts may be introduced into one or both flowgrams in order to optimize their alignment; and the flowgram of the read may be computationally reverse-complemented in order that it is always shown in the 5'→3' orientation of the Reference Sequence. Finally, the flowgram only displays the subset of flows relevant to the read's sequence alignment, as displayed in the Global Align or Consensus Align tabs.

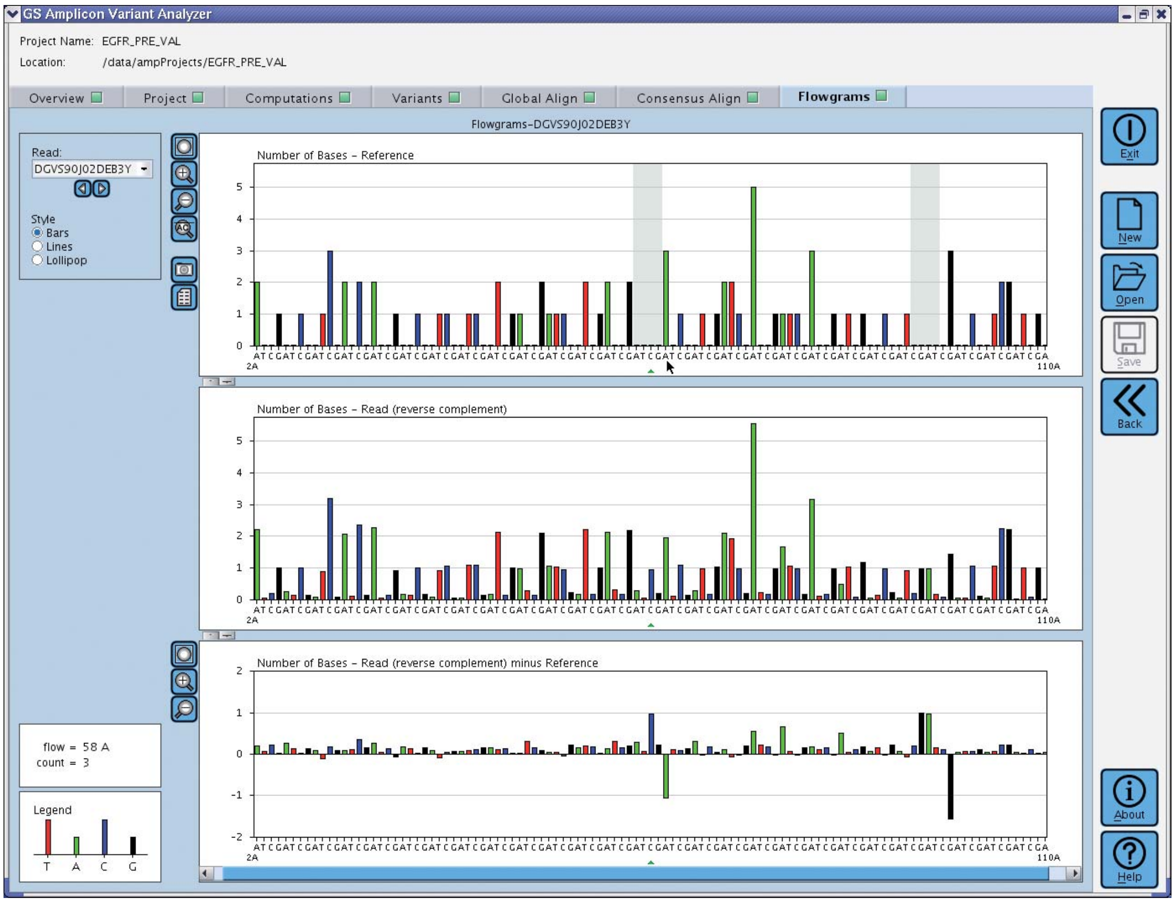
The Flowgrams tab's main feature is a "tri-flowgram " plot showing (Figure 9–65):

- an aligned, idealized flowgram for the Reference Sequence.
- an aligned, (possibly reverse complemented) flowgram of the read.
- a difference flowgram (read minus reference), where any variation from the Reference Sequence is seen as a non-zero value. (Extra signal in the read, relative to the Reference Sequence, shows up as positive differences in this panel.)

In addition to the tri-flowgram plot, the Flowgrams tab contains a small set of display option and navigation tools in the upper-left corner, and the usual "Mouse Tracker" and color-code legend, in the lower-left corner.

Examining flowgrams can be useful when trying to assess whether a variation may be genuine or due to an artifact. For example, the flowgram of a mononucleotide from the Reference Sequence called as a dinucleotide repeat in a given read may show that the signal was barely over the threshold for calling a two-nucleotide incorporation, casting doubt on the second base of the call. Conversely, variations that induce a cycle shift in the flow alignment are particularly compelling since such shifts would not be expected as a result of simple overcalling or undercalling during signal processing, nor would they result from sequencing artifacts such as incomplete extension or carry forward. If the flowgram indicates that a variation appears genuine, the user should still consider whether it also occurs in other, overlapping reads, especially in the opposite orientation, or in a replicate experimental Sample. Alternatively, could the variation simply be due to a PCR artifact introduced early in the sample preparation process? Anticipating these types of questions should play a large role in the experimental design of an Amplicon sequencing experiment.

A brief explanation of the information contained in a flowgram is provided along with a full description of the tri-flowgram display, in section 9.8.2. For more details on flowgrams and on the processing of data that generates them, see sections 5.2.8.4 and 6.2.9.5.


GS Amplicon Variant Analyzer

Project Name: EGFR_PRE_VAL
Location: /data/ampProjects/EGFR_PRE_VAL

Read: DGVS90J02DEB3Y
Style: [✓ Bars] Lines Lollipop

flow = 58 A
count = 3

Flowgrams-DGVS90J02DEB3Y
Overview
Project
Computations
Variants
Global Align
Consensus Align
Flowgrams
Number of Bases – Reference
Number of Bases – Read (reverse complement)
Number of Bases – Read (reverse complement) minus Reference
2A
110A
Legend
T
A
C
G
Exit
New
Open
Save
Back
About
Help

Figure 9–65: The Flowgrams tab

The flowgram alignment algorithm works only by introducing cycle shifts, with the goal of minimizing the sum of the absolute values from the signals in the difference plot while simultaneously attempting to minimize the number of cycle shifts introduced. The alignment algorithm does not attempt to split larger individual flow values into multiple flows of lesser magnitude, which could allow it to produce results that more closely mimic the alignments one would obtain by working in nucleotide-space (for an example where this situation arises, see section 10.3.2, below Figure 10–30).

9.8.1 Populating the Flowgrams Tab

When you open an Amplicon Project in the AVA software, the Flowgrams tab has no content and is grayed-out. To populate it, you must use the "Open Flowgrams <Name of the Read>" action from the contextual menu that appears when you right-click one of the following two sources:

- A single read on the Global Align tab (make sure that its Read Type control is set to "Individual")
- A single read on the Consensus Align tab (which always displays individual reads)

Once the Flowgrams tab is populated, a small green arrow points to the flow corresponding to the nucleotide upon which you right-clicked. To load the Flowgrams tab with another read, you can again use the right-click method above and replace the displayed tri-flowgram with another one. But if you are in the Flowgrams tab, you have another, more powerful option: this tab has two "Read" controls in its upper-left corner that allow you to browse through and navigate all the reads from the source tab that generated the read currently displayed (the Global Align or the Consensus Align tab). These controls are described in detail in section 9.8.3. As you navigate to other reads with these controls, the AVA software attempts to maintain the focus of the green arrow on the same nucleotide in the source tab. By observing the distribution of signals for a particular nucleotide over many reads, one may obtain increased (or diminished) confidence for a given variation.

9.8.2 The Tri-Flowgram Plot

A flowgram is a graphic representation of the number of nucleotides added to the nascent DNA strands present in a given well of a PicoTiterPlate device during each nucleotide flow of a sequencing Run. Simply put, it shows the succession of nucleotide flows of the sequencing Run on the horizontal axis, and the number of nucleotides incorporated during each flow on the vertical axis (as histogram bars, with the nucleotides color-coded per the legend at the lower-left of the tab).

In Amplicon sequencing, because the software not only knows the data from the reads but also has a Reference Sequence to which each read is to be compared, the AVA application can calculate an "ideal flowgram" of the Reference Sequence corresponding to the read (*i.e.* the Amplicon, in AVA software language), and display the difference. The AVA software presents this to the user in the form of a "tri-flowgram". The three plots of a tri-flowgram show the following:

- The top plot shows the calculated flowgram of the segment from the Reference Sequence corresponding to the Amplicon that produced the read currently displayed. This is simply the even number of bases that would result from the perfect incorporation of all nucleotides along a sequencing template of that sequence, during a sequencing Run on the Genome Sequencer Instrument.
- The middle plot shows the exact signals recorded at each flow of the sequencing Run for the read currently displayed; these signals have been converted into "nucleotide" units. If the read was from the DNA strand opposite the Reference Sequence, the plot will display the signals after calculating the reverse complement of the read sequence, so the read flowgram will align with that of its Refe,rence Sequence.
- The bottom plot is not truly a flowgram; it shows the difference between the two plots above (read minus Reference), such that if the read matches the Reference Sequence at a given flow, the histogram bar height is zero; if the read has a stronger signal than expected from the known Reference, the bar height is greater than zero; and if the read has a weaker signal than expected from the known Reference, the bar height is less than zero. This "difference flowgram" thus conveniently shows all flow-by-flow variations between the read and its Reference Sequence as departures from zero.

The tri-flowgram plot has all the standard navigation features (scrollbars, zoom buttons, mouse tracker, *etc.*) described in section 9.1.3.3. A separate set of zoom buttons is available for the difference flowgram because it is sometimes convenient to examine it at a different scale than the Reference and read flowgrams. Also, the "Save plot snapshot to .png file" and "Save plot data to .xls (spreadsheet) file" buttons will save all three plots together, in a single file. The histogram bars can be displayed in your choice of three styles (Bars, Lines, and Lollipop), by selecting the corresponding radio button near the upper-left corner of the tab. The three plots of the Flowgrams tab can also be resized and collapsed, as described in section 9.1.3.2.

The tri-flowgram plots have another feature that is important to maintain the alignment between the two upper plots: due to the interplay between the nature of a variation and the flow order of the nucleotide cycle during the sequencing Run, a situation sometimes occurs whereby the Reference and read flowgrams fall out of phase by one or more cycles of four flows. If this were allowed to happen, all the flows beyond the cycle shift would be misaligned, and all the values in the difference flowgram would be wrong. Cycle shifts are therefore inserted in either the Reference or the read flowgram, as appropriate, to maintain their synchronicity; these "empty" inserted cycles are marked in gray on the flowgrams (Figure 9–66).

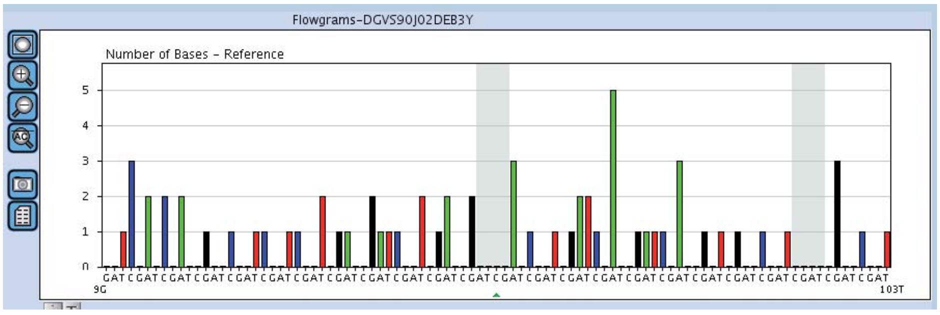

Flowgrams-DGVS90J02DEB3Y
Number of Bases - Reference
5
4
3
2
1
0
GATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGAT
9G
103T

Figure 9–66: Example of a flowgram with inserted nucleotide cycles

9.8.3 Navigation on the Flowgrams Tab

There are two navigation controls located at the upper-left corner of the tab that allow you to select new flowgrams to display in the Flowgrams tab. The first is a drop down menu (see Figure 9–65, above) that contains a list of all the reads that are present in the source tab that generated the read currently displayed (the Global Align or the Consensus Align tab). Selecting a new read from the drop down menu will update the Flowgrams tab with the tri-flowgram for the new read, replacing the current data. This allows you to quickly compare the flowgrams of various reads over the data group available (*e.g.* all the reads of a Sample, for a single or a given set of Amplicon(s) as displayed in the Global Align tab; or a sub-list of the reads of a given consensus, as restricted by one or more selection(s) made on the Consensus Align tab).

The second Read display control allows you to scan the set of reads even faster. It consists of a pair of arrow buttons located just below the 'Read' drop down menu. Clicking one of these buttons directly replaces the tri-flowgram currently displayed by the one of the read next to (or preceding) it in the list. This update is very fast and allows you to quickly scan the flowgrams for a large number of reads to see if a variation (most readily seen on the difference flowgram) is rare or, on the contrary, present in many (or most) of them.

10. Example Amplicon Project Design and Analysis

To help better guide the reader through the process of an Amplicon Sequencing experiment, this section provides a fictitious example of the whole procedure, starting with setting the objectives and including the design of the Amplicon libraries, sequencing, and the full analysis of the results through the determination of the frequency of previously known as well as novel Variants observed in the Sample. Emphasis, however, will be on software.

Features supported only under the GS FLX standard chemistry: While the GS Amplicon Variant Analyzer software v. 2.0.00 provides important new features, such as the support for MIDs with Amplicon libraries, those libraries are not currently supported under the GS FLX Titanium chemistry. Therefore, the example Project described in this section applies only to Amplicon libraries prepared using the GS FLX standard chemistry.

This example describes a Project that does not make use of MIDs or Multiplexers. For an example of how these features are used in an Amplicon Project (using the CLI), see section 11.6.

10.1 Experimental Design

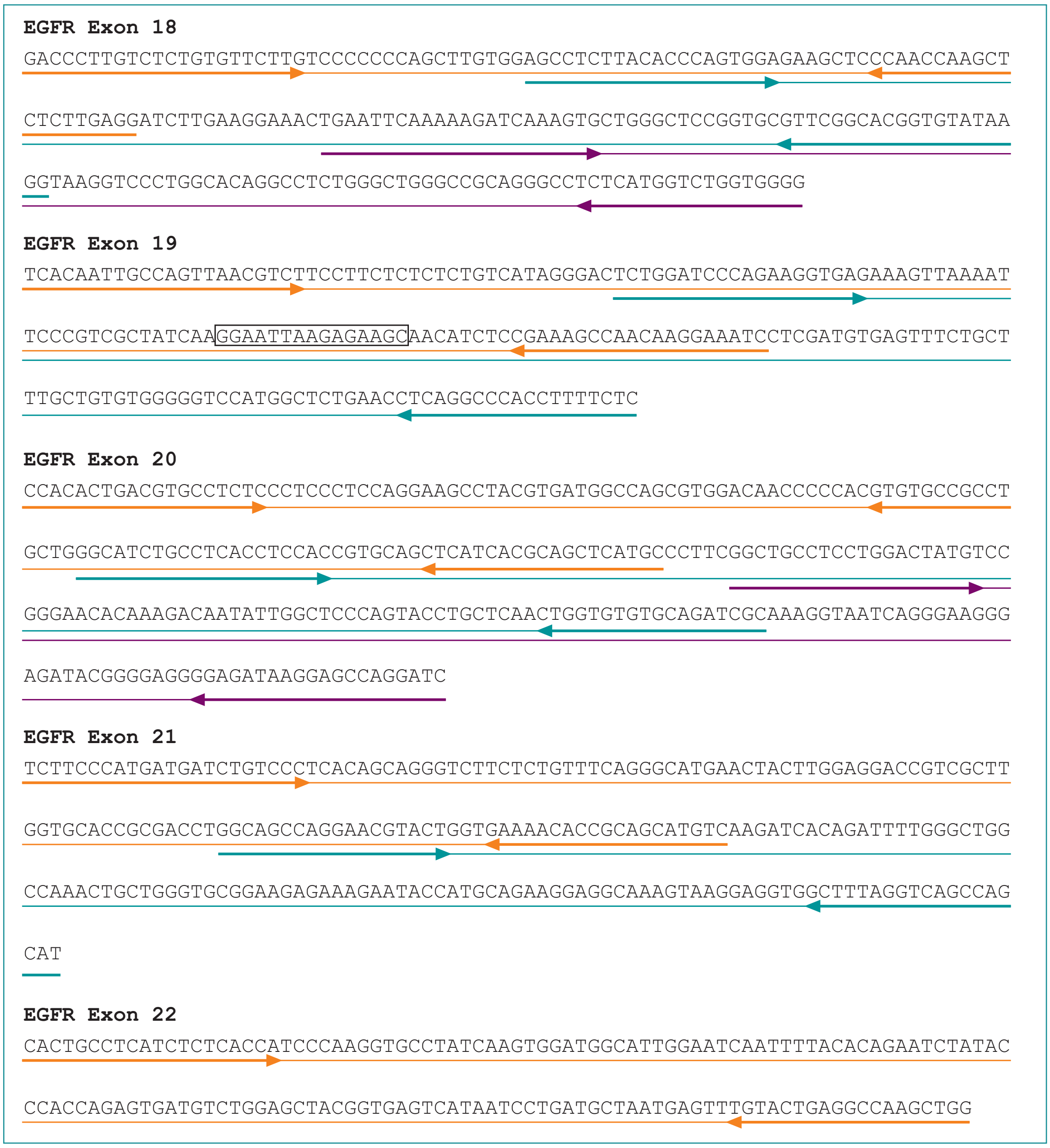
In this example, we will look for Variants in five exons from the human Epidermal Growth Factor Receptor gene, EGFR exons 18 through 22, in a single DNA source. The sequences of the 5 exons are known (*e.g.* from public databases), and are shown in Figure 10–1. We further posit that there is a known Variant in exon 19 that we want to track: a 15 bp deletion at positions 93–107 (inclusive) of the exon.

In order to be able to gather sequencing data from both orientations over the full length of the exons (as much as possible), we define a set of overlapping Amplicons whereby every nucleotide is within about 100 bp from each of two facing primers, providing nearly full coverage of each exon in both orientations [see Figure 10–1. Note that this example was created for the Genome Sequencer 20 System but that the Genome Sequencer FLX System features read lengths of over 200 bp for the GS FLX standard chemistry (Amplicon libraries are not currently supported by the GS FLX Titanium chemistry); this would allow a GS FLX System user to design longer Amplicons than those used in this example]. A Primer Design software can assist in this task.

Overall, we define 11 Amplicons, as listed in Table 10–1. To ensure proper representation of all Amplicons in our experiment, we will generate 11 single-peak Amplicon libraries (as opposed to attempting a multiplex amplification). As described in the *GS FLX Amplicon DNA Library Preparation Method Manual*, Amplicon libraries are made using Fusion Primers; for our experiment, all forward primers ("Primer1" from Table 10–1) are fused to Primer A, in the configuration 5'-PrimerA-Primer1-3'; and all reverse primers ("Primer2" from Table 10–1) are fused to Primer B, in the configuration 5'-PrimerB-Primer2-3'. Since the experiment comprises only one Sample, we do not need to use MIDs; the software will recognize the Amplicon to which each read belongs by looking at the Primer 1 or Primer 2 sequence that it will see at the beginning of the read.

Amplicon Name	Primer1 (Forward; 5'→3')	Primer2 (Reverse; 5'→3')
EGFR_18_1	GACCCTTGTCTCTGTGTTCTTG	CCTCAAGAGAGCTTGGTTGG
EGFR_18_2	AGCCTCTTACACCCAGTGGA	CCTTATACACCGTGCCGAAC
EGFR_18_3	TGAATTCAAAAAGATCAAAGTG	CCCCACCAGACCATGAGA
EGFR_19_1	TCACAATTGCCAGTTAACGTCT	GATTTCCTTGTTGGCTTTCG
EGFR_19_2	TCTGGATCCCAGAAGGTGAG	GAGAAAAGGTGGGCCTGAG
EGFR_20_1	CCACACTGACGTGCCTCTC	GCATGAGCTGCGTGATGAG
EGFR_20_2	GCATCTGCCTCACCTCCAC	GCGATCTGCACACACCAG
EGFR_20_3	GGCTGCCTCCTGGACTATGT	GATCCTGGCTCCTTATCTCC
EGFR_21_1	TCTTCCCATGATGATCTGTCCC	GACATGCTGCGGTGTTTTC
EGFR_21_2	GGCAGCCAGGAACGTACT	ATGCTGGCTGACCTAAAGC
EGFR_22_1	CACTGCCTCATCTCTCACCA	CCAGCTTGGCCTCAGTACA

Table 10–1: Names of the Amplicons defined for the EGFR experiment, and the Primers used to create them



EGFR Exon 18

```
GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACACCCAGTGGAGAAGCTCCCAACCAAGCT

CTCTTGAGGATCTTGAAGGAAACTGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGCGTTCGGCACGGTGTATAA

GGTAAGGTCCCTGGCACAGGCCTCTGGGCTGGGCCGCAGGGCCTCTCATGGTCTGGTGGGG
```

EGFR Exon 19

```
TCACAATTGCCAGTTAACGTCTTCCTTCTCTCTCTGTCATAGGGACTCTGGATCCCAGAAGGTGAGAAAGTTAAAAT

TCCCGTCGCTATCAAGGAATTAAGAGAAGCAACATCTCCGAAAGCCAACAAGGAAATCCTCGATGTGAGTTTCTGCT

TTGCTGTGTGGGGGTCCATGGCTCTGAACCTCAGGCCCACCTTTTCTC
```

EGFR Exon 20

```
CCACACTGACGTGCCTCTCCCTCCCTCCAGGAAGCCTACGTGATGGCCAGCGTGGACAACCCCCACGTGTGCCGCCT

GCTGGGCATCTGCCTCACCTCCACCGTGCAGCTCATCACGCAGCTCATGCCCTTCGGCTGCCTCCTGGACTATGTCC

GGGAACACAAAGACAATATTGGCTCCCAGTACCTGCTCAACTGGTGTGTGCAGATCGCAAAGGTAATCAGGGAAGGG

AGATACGGGGAGGGGAGATAAGGAGCCAGGATC
```

EGFR Exon 21

```
TCTTCCCATGATGATCTGTCCCTCACAGCAGGGTCTTCTCTGTTTCAGGGCATGAACTACTTGGAGGACCGTCGCTT

GGTGCACCGCGACCTGGCAGCCAGGAACGTACTGGTGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCTGG

CCAAACTGCTGGGTGCGGAAGAGAAAGAATACCATGCAGAAGGAGGCAAAGTAAGGAGGTGGCTTTAGGTCAGCCAG

CAT
```

EGFR Exon 22

```
CACTGCCTCATCTCTCACCATCCCAAGGTGCCTATCAAGTGGATGGCATTGGAATCAATTTTACACAGAATCTATAC

CCACCAGAGTGATGTCTGGAGCTACGGTGAGTCATAATCCTGATGCTAATGAGTTTGTACTGAGGCCAAGCTGG
```

Figure 10–1: DNA sequence of the five human EGFR exons in which we will be searching for Variants. The figure shows the location of the regions (colored underlines) and Primers (colored arrows) used to generate Amplicon libraries for sequencing. The box indicates the known deletion Variant in exon 19.

With these Fusion Primers on hand (and the initial DNA sample), we can proceed with the preparation of the 11 Amplicon libraries. Proper amounts of each library are subjected to the emPCR amplification process using both emPCR kits II and III, so that we will have reads that start from both the Primer A and Primer B ends of each Amplicon. Library preparation is described in detail in the *GS FLX Amplicon DNA Library Preparation Method Manual*; the emPCR amplification procedure is described in the *GS FLX emPCR Method Manual*.

Preparing for the sequencing proper, we calculate that, with an expected minimum of 33,000 high quality reads per medium region of a PicoTiterPlate device (on the Genome Sequencer 20 System), we can pool all 11 Amplicon libraries, and load them together in a single region and carry out a single sequencing Run: we can then expect approximately 3000 reads for each Amplicon (about 1500 in each orientation), not counting for overlaps, a depth of coverage sufficient to detect and reasonably quantitate Variants down to a frequency of approximately 3–5% (this depends on the nature of the variation, the sequence environment, and other factors), which is sufficient for our purpose. (As above, this example was created for the Genome Sequencer 20 System; in the Genome Sequencer FLX System, one would expect 80,000 reads or more per medium region of a PicoTiterPlate device for a good quality library.) For details about sequencing Runs and the operation of the Genome Sequencer Instrument, see the *GS FLX Sequencing Method Manual*.

10.2 Project Setup in the AVA Software

For our EGFR experiment example, let's posit that a 4-region sequencing Run generated four SFF files, one of which, named "DGVS90J03.sff", contains all the reads of our combined 11 Amplicon libraries and is now available on the local file-system.

Also, while each exon could be used individually as Reference Sequences in the Variant analysis, we decide that it would be more convenient to report on all the Variants found in all five exons together. To simplify the analysis, therefore, we create an "artificial" Reference Sequence by concatenating the sequences of the 5 exons, with strings of 20 'N' characters to separate them. The resulting single Reference Sequence is shown in Table 10–2.

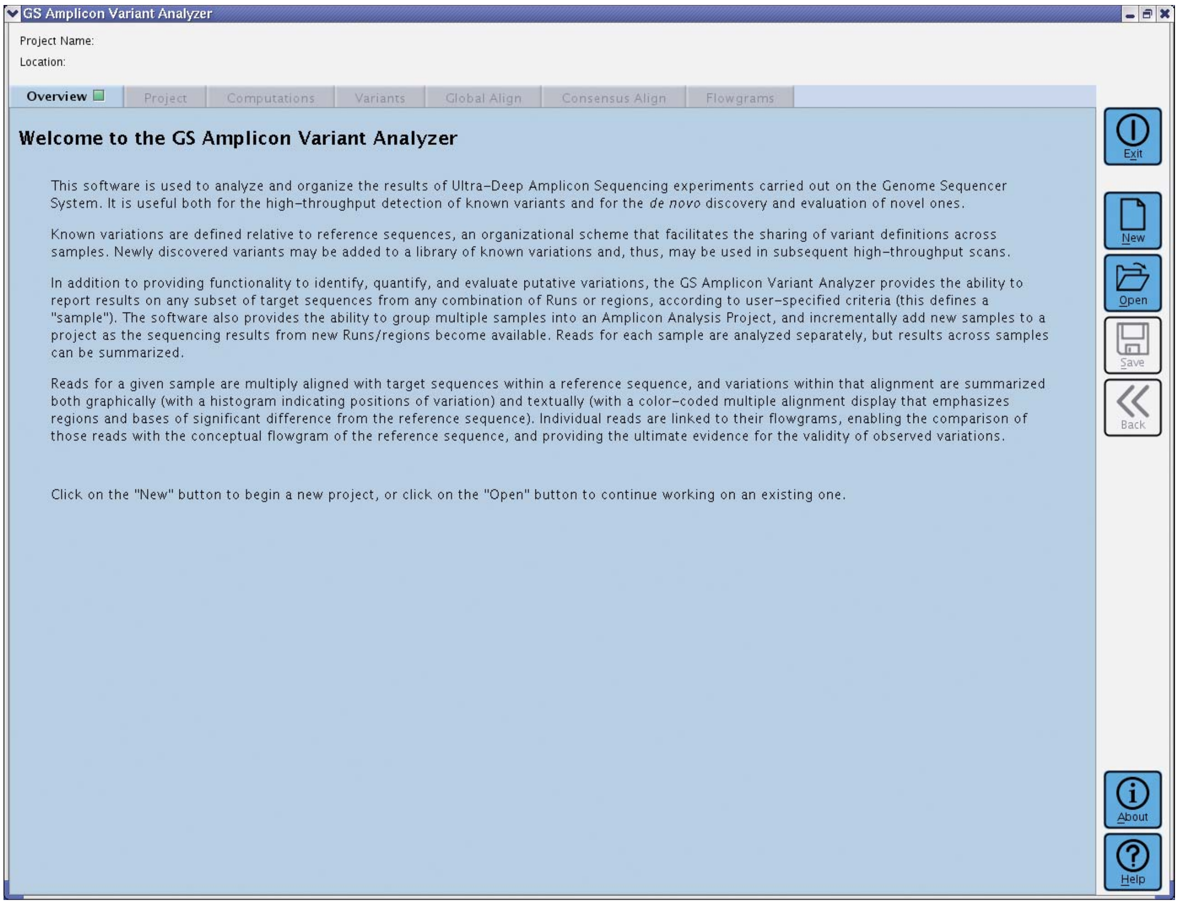
```
EGFR Exons 18-22
GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACACCCAGTGGAGAAGCTCCCAACCAAGCT
CTCTTGAGGATCTTGAAGGAAACTGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGCGTTCGGCACGGTGTATAA
GGTAAGGTCCCTGGCACAGGCCTCTGGGCTGGGCCGCAGGGCCTCTCATGGTCTGGTGGGGNNNNNNNNNNNNNNNN
NNNNTCACAATTGCCAGTTAACGTCTTCCTTCTCTCTCTGTCATAGGGACTCTGGATCCCAGAAGGTGAGAAAGTTA
AAATTCCCGTCGCTATCAAGGAATTAAGAGAAGCAACATCTCCGAAAGCCAACAAGGAAATCCTCGATGTGAGTTTC
TGCTTTGCTGTGTGGGGGTCCATGGCTCTGAACCTCAGGCCCACCTTTTCTCNNNNNNNNNNNNNNNNNNNNCCACA
CTGACGTGCCTCTCCCTCCCTCCAGGAAGCCTACGTGATGGCCAGCGTGGACAACCCCCACGTGTGCCGCCTGCTGG
GCATCTGCCTCACCTCCACCGTGCAGCTCATCACGCAGCTCATGCCCTTCGGCTGCCTCCTGGACTATGTCCGGGAA
CACAAAGACAATATTGGCTCCCAGTACCTGCTCAACTGGTGTGTGCAGATCGCAAAGGTAATCAGGGAAGGGAGATA
CGGGGAGGGGAGATAAGGAGCCAGGATCNNNNNNNNNNNNNNNNNNNNTCTTCCCATGATGATCTGTCCCTCACAGC
AGGGTCTTCTCTGTTTCAGGGCATGAACTACTTGGAGGACCGTCGCTTGGTGCACCGCGACCTGGCAGCCAGGAACG
TACTGGTGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCTGGCCAAACTGCTGGGTGCGGAAGAGAAAGAA
TACCATGCAGAAGGAGGCAAAGTAAGGAGGTGGCTTTAGGTCAGCCAGCATNNNNNNNNNNNNNNNNNNNNCACTGC
CTCATCTCTCACCATCCCAAGGTGCCTATCAAGTGGATGGCATTGGAATCAATTTTACACAGAATCTATACCCACCA
GAGTGATGTCTGGAGCTACGGTGAGTCATAATCCTGATGCTAATGAGTTTGTACTGAGGCCAAGCTGG
```

Table 10–2: "Artificial" Reference Sequence comprising exons 18 through 22 of EGFR, concatenated and with separating strings of 20 N characters

Many of the operations described through the end of section 10.4 can be done in more than one way. For example, most actions that can be performed by clicking a button can also be accessed via a contextual menu and/or by double-clicking in a field in a Table. Only one way is given in the example below; for more information, see the detailed description of each tab in section 9.

10.2.1 Launching the AVA Application

The next step is to launch the GS Amplicon Variant Analyzer (AVA) application, which is done from the command line, using the "`gsAmplicon`" command (see section 9.1.2); the AVA software splash screen appears briefly, while the application is launching, and then the AVA main window is displayed with its Overview tab showing the AVA introduction text and the 7 main buttons, to the right of the window; the 'Project Name' and 'Location' fields at the top left are initially blank and all the other tabs are grayed-out, since no project is open (see Figure 10–2).



Welcome to the GS Amplicon Variant Analyzer

This software is used to analyze and organize the results of Ultra-Deep Amplicon Sequencing experiments carried out on the Genome Sequencer System. It is useful both for the high-throughput detection of known variants and for the *de novo* discovery and evaluation of novel ones.

Known variations are defined relative to reference sequences, an organizational scheme that facilitates the sharing of variant definitions across samples. Newly discovered variants may be added to a library of known variations and, thus, may be used in subsequent high-throughput scans.

In addition to providing functionality to identify, quantify, and evaluate putative variations, the GS Amplicon Variant Analyzer provides the ability to report results on any subset of target sequences from any combination of Runs or regions, according to user-specified criteria (this defines a "sample"). The software also provides the ability to group multiple samples into an Amplicon Analysis Project, and incrementally add new samples to a project as the sequencing results from new Runs/regions become available. Reads for each sample are analyzed separately, but results across samples can be summarized.

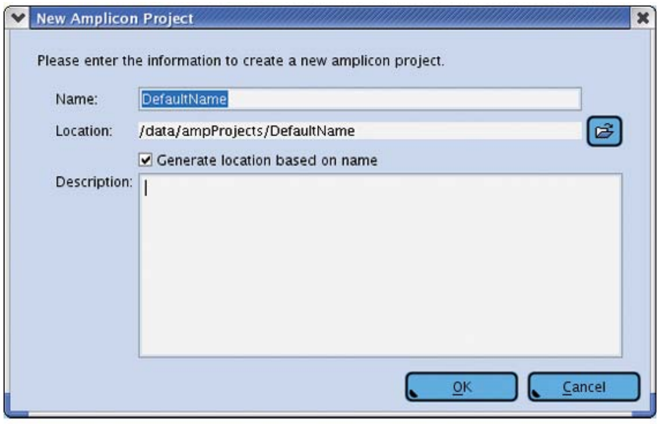
Reads for a given sample are multiply aligned with target sequences within a reference sequence, and variations within that alignment are summarized both graphically (with a histogram indicating positions of variation) and textually (with a color-coded multiple alignment display that emphasizes regions and bases of significant difference from the reference sequence). Individual reads are linked to their flowgrams, enabling the comparison of those reads with the conceptual flowgram of the reference sequence, and providing the ultimate evidence for the validity of observed variations.

Click on the "New" button to begin a new project, or click on the "Open" button to continue working on an existing one.

Figure 10–2: The AVA main window at start up

10.2.2 Creating a New Project

Since we want to create a new Project, we click the "New" button. (The "Open" button can be used to load a pre-existing Project.) The "New Amplicon Project" window opens, where we can specify the 'Name', 'Location', and 'Description' for the new Project (Figure 10–3). Note that since the 'Generate location based on name' box is checked, the 'Name' and 'Location' fields are linked, so when we type a name to replace the 'DefaultName' in the 'Name' field, the 'DefaultName' portion of the 'Location' will dynamically update to match the content of the 'Name' box; this is an easy way to ensure that the same name is used for the Project and for the folder that contains it (the "Location"), which is usually what one would want to do.

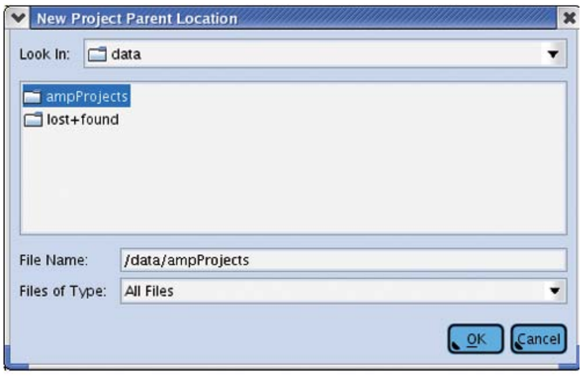


Please enter the information to create a new amplicon project.

Name: DefaultName
Location: /data/ampProjects/DefaultName
☑ Generate location based on name
Description:

Figure 10–3: The New Amplicon Project window

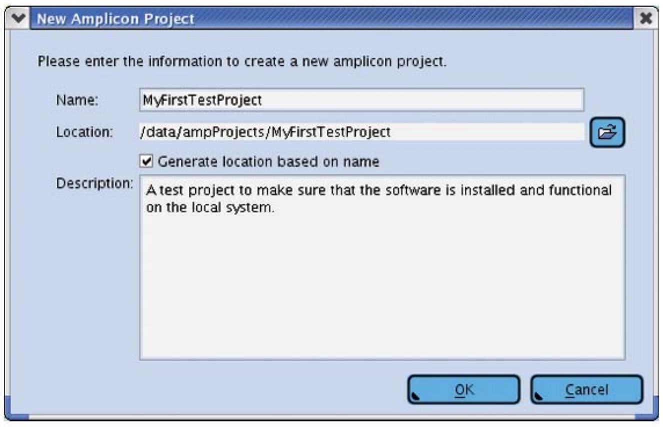
To keep things simple, we initially leave the 'Name' field alone and first select the 'Location' we want. The folder icon to the right of the 'Location' field opens the "New Project Parent Location" window which allows us to navigate the file-system easily (Figure 10–4). The object is to identify a parent location where we want to store Project directories (as opposed to the full path to this particular new Project directory), providing a standard base of operations. It is important to choose a directory where we have both read and write permissions.



Look In: data
ampProjects
lost+found
File Name: /data/ampProjects
Files of Type: All Files

Figure 10–4: The New Project Parent Location window

To pursue our example, let's assume that we have read and write permissions in the '/data/ampProjects' directory on our local system, and that we chose 'ampProjects' as the parent location directory (Figure 10–4). The path to this directory is used to form a 'File Name'. Clicking 'OK' returns us to the "New Amplicon Project" window, where the path we just chose is reflected in the 'Location' field: '/data/ampProjects/DefaultName'. Editing the contents of the 'Name' field to 'myFirstTestProject' (with the 'Generate location based on name' box checked), provides a full path for the Location of the new Project (Figure 10–5). This Figure also shows a short annotation entered in the 'Description' field.



Please enter the information to create a new amplicon project.

Name: MyFirstTestProject

Location: /data/ampProjects/MyFirstTestProject

☑ Generate location based on name

Description: A test project to make sure that the software is installed and functional on the local system.

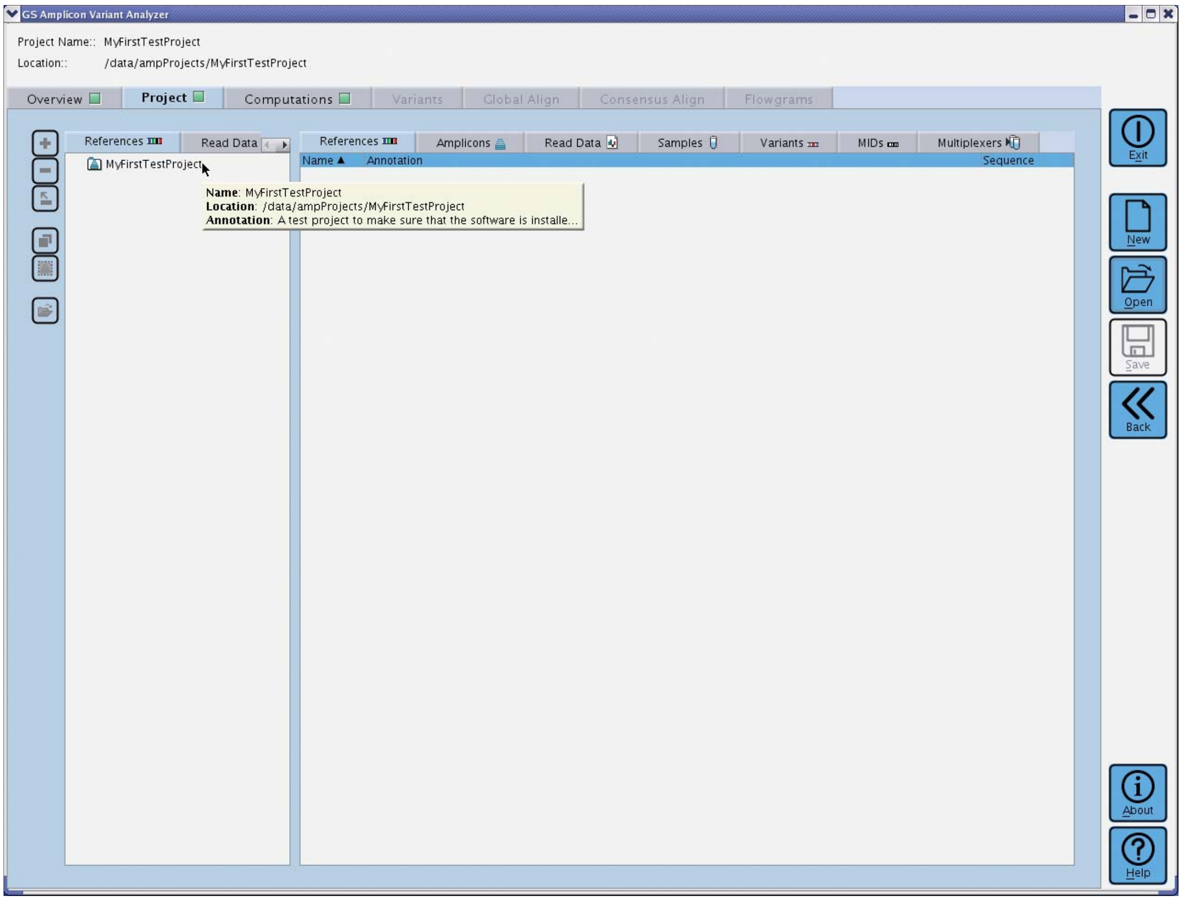
Figure 10–5: The New Amplicon Project window, with the Name, Location and Description of the new project

Clicking 'OK' at this point closes the "New Amplicon Project" window, creates the Project and the Location (including a proper subdirectory structure for the functioning of the Project computation and result storage), and opens the new Project in the AVA main window, in its "Project" tab (see next section).

Since this Project does not use MIDs, the 454Standard MID set that is automatically loaded when a new Project is created (see section 12.4), has been manually removed to simplify the display of the Project.

10.2.3 Defining the Reference Sequence

Figure 10–6 shows the AVA main window with its Project tab in the front, showing the new (empty) Project. The 'Project Name' and 'Location' fields at the top left of the window match the new Project information we just entered. The 'Project' tab is in bold black lettering on blue background with a green square icon, indicating it is ready to be used to setup the Project. The other two accessible tabs are 'Overview' and 'Computations' (black type on gray background and green square icons), while the remaining tabs don't yet have any content and remain grayed-out. The References Tree (left panel of the Project tab) contains a folder representing the Project, and nothing else. The 6 context-sensitive buttons in the upper left hand margin of the "tree" panel ('Add', 'Remove from project', 'Remove association and remain in project', 'Duplicate item', 'Select Amplicons associated with item', and 'Import data') start out inactive and grayed-out until we select something that can give those buttons context as to what needs to be added or removed (*e.g.* we might select an object type tab in the right table-view area).


GS Amplicon Variant Analyzer

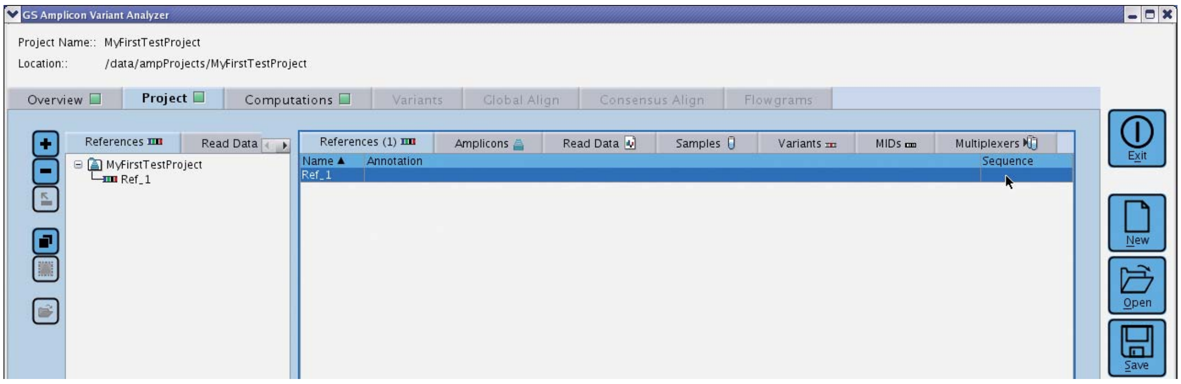
Project Name:: MyFirstTestProject

Location:: /data/ampProjects/MyFirstTestProject

Name: MyFirstTestProject
Location: /data/ampProjects/MyFirstTestProject
Annotation: A test project to make sure that the software is installe...

Figure 10–6: The AVA main window ('Project' tab) of a newly created Project

For the example Project, we want to enter a Reference Sequence first, so we click on the right panel's 'References' sub-tab, or References Definition Table. This enables the 'Add' button on the left margin (the button turns blue rather than gray) and a blue outline appears around the Definition Table Panel (indicating that the active buttons on the left can operate on the selected item in the Definition Table). Clicking the 'Add' button creates a new Reference Sequence entry in the References Definition Table and creates a corresponding reference node in the References Tree (Figure 10–7).



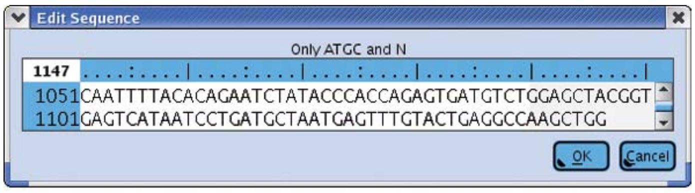
Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

MyFirstTestProject
Ref_1

Name ▲	Annotation	Sequence
Ref_1		

Figure 10–7: The Project tab, with the References Tree and References Definition Table displayed in the left and right panels, respectively. Note that the 'Add' button is enabled; it was used to create a new Reference Sequence entry.

To define the Reference Sequence, we double-click on the fields in its row, in the References Definition Table. In our example, we begin by changing the default Name "Ref_1" to "EGFR_Exons_18-22". Note that the tree and table views are linked and editing the Reference Sequence name in the table also changes its name in the tree. Annotations are optional; double-clicking in the Annotations filed for our Reference Sequence opens a text entry window (not shown). Finally, we double-click in the 'Sequence' field of our Reference Sequence; an 'Edit Sequence' window appears in which we paste the previously prepared "artificial" Reference Sequence covering all five exons, prepared before (Figure 10–8).



```
1147
1051CAATTTTACACAGAATCTATACCCACCAGAGTGATGTCTGGAGCTACGGT
1101GAGTCATAATCCTGATGCTAATGAGTTTGTACTGAGGCCAAGCTGG
```

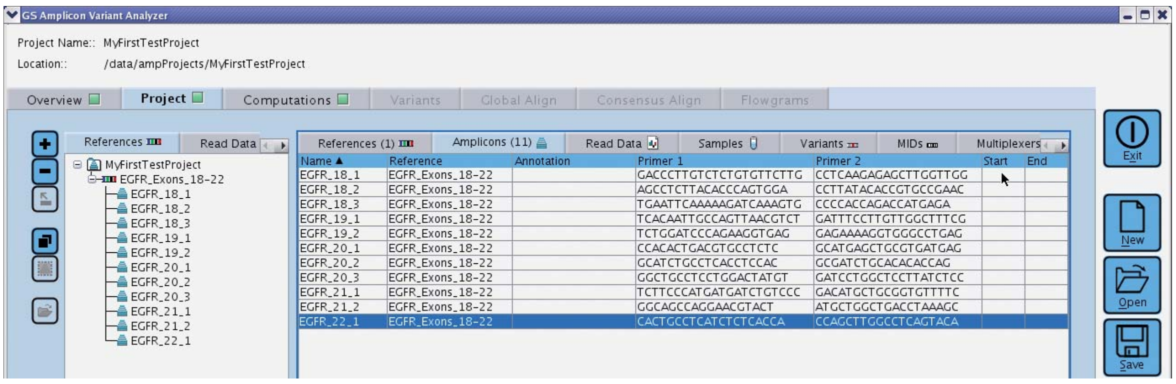
Figure 10–8: The Edit Sequence window, in which we pasted the 1146 nt "artificial" Reference Sequence covering exons 18-22 of EGFR

10.2.4 Defining the Amplicons

Now that we have a Reference Sequence, we can enter our Amplicons. To do so, we click on the 'Amplicons' sub-tab of the table-view. The 'Add' button on the left margin is now capable of creating new Amplicons, as this is its new context. For our EGFR Project, we need to add 11 Amplicons, so we click the 'Add' button 11 times, which adds 11 rows in the Amplicons Definition Table, with generic Amplicon Names. As before, we enter the information by double-clicking into the various fields for each Amplicon; the fields are: 'Name', 'Reference', 'Annotation' (optional), 'Primer 1', 'Primer 2', 'Start', and 'End').

- Since we have only one Reference Sequence, we can associate all our Amplicons with it at once by multi-selecting all the rows from the Amplicons Definition Table and dragging the selection to the proper (in our case, only) node in the References Tree, on the left. This adds a branch of Amplicon nodes hanging off the Reference node; and the 'Reference' fields on the Amplicons Definition Table are updated so that this sequence now appears in the 'Reference' field for each Amplicon.
- Primer 1 and Primer 2 are those used to prepare the Amplicon library (or libraries); for our example, these are listed in Table 10–1 (above), along with the Amplicon Names. We enter the sequence of all the Primers by double-clicking in each of the 'Primer' fields; this opens a sequence editor window into which the sequence can be typed or pasted (always 5'→3'; the software will compute the reverse complement of Primer 2 to align it to the Reference Sequence). For the Amplicon Names, we double-click in the 'Names' fields and type or paste the Amplicon Names in the Table cells.
- We choose not to enter any Annotations for our Amplicons.

After adjusting the width of the column (by dragging of the separation lines between the headers) so that all the fields are completely readable, the application view looks as shown in Figure 10–9.

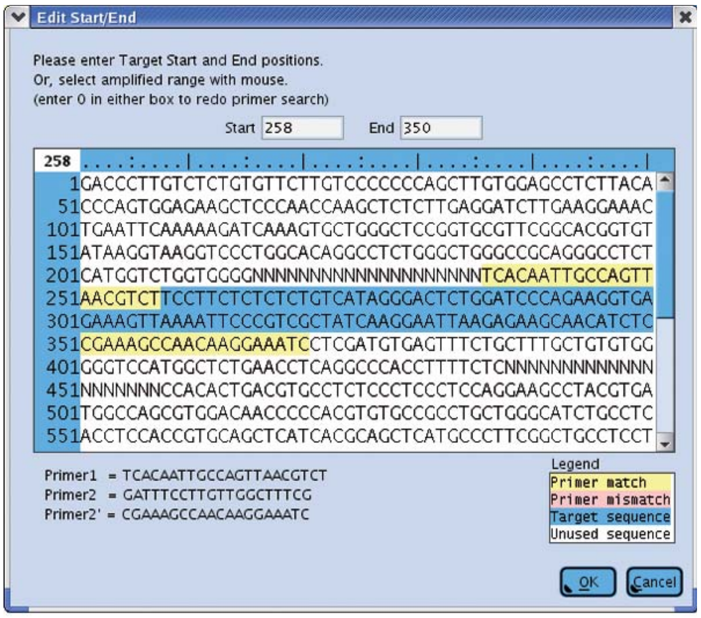


Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Name ▲	Reference	Annotation	Primer 1	Primer 2	Start	End
EGFR_18_1	EGFR_Exons_18-22		GACCCTTGTCTCTGTGTTCTTG	CCTCAAGAGAGCTTGGTTGG		
EGFR_18_2	EGFR_Exons_18-22		AGCCTCTTACACCCAGTGGA	CCTTATACACCGTGCCGAAC		
EGFR_18_3	EGFR_Exons_18-22		TGAATTCAAAAAGATCAAAGTG	CCCCACCAGACCATGAGA		
EGFR_19_1	EGFR_Exons_18-22		TCACAATTGCCAGTTAACGTCT	GATTTCCTTGTTGGCTTTCG		
EGFR_19_2	EGFR_Exons_18-22		TCTGGATCCCAGAAGGTGAG	GAGAAAAGGTGGGCCTGAG		
EGFR_20_1	EGFR_Exons_18-22		CCACACTGACGTGCCTCTC	GCATGAGCTGCGTGATGAG		
EGFR_20_2	EGFR_Exons_18-22		GCATCTGCCTCACCTCCAC	GCGATCTGCACACACCAG		
EGFR_20_3	EGFR_Exons_18-22		GGCTGCCTCCTGGACTATGT	GATCCTGGCTCCTTATCTCC		
EGFR_21_1	EGFR_Exons_18-22		TCTTCCCATGATGATCTGTCCC	GACATGCTGCGGTGTTTTC		
EGFR_21_2	EGFR_Exons_18-22		GGCAGCCAGGAACGTACT	ATGCTGGCTGACCTAAAGC		
EGFR_22_1	EGFR_Exons_18-22		CACTGCCTCATCTCTCACCA	CCAGCTTGGCCTCAGTACA		

Figure 10–9: The AVA window after associating all 11 Amplicons to the single 'EGFR_Exons_18-22' Reference Sequence, and entering the Amplicons' names and Primer sequences

Next, we define the 'Targets' by specifying their Start and End (*i.e.* by positioning the Primers along the Reference Sequence), for each Amplicon. To do this, we double-click in either the 'Start' or the "End' field of each Amplicon. This opens the Edit Start/End window for this Amplicon and carries out an automatic search for its Primer 1 and the reverse-complement of its Primer 2 along its Reference Sequence (Figure 10–10). As long as we made no errors when entering the Primer sequences, each Primer in our EGFR example will find an exact unique match and be displayed with a yellow background, and the Start and End of the Target will appear in the corresponding fields in the Amplicons Definition Table.



Please enter Target Start and End positions.
Or, select amplified range with mouse.
(enter 0 in either box to redo primer search)

Start 258 End 350

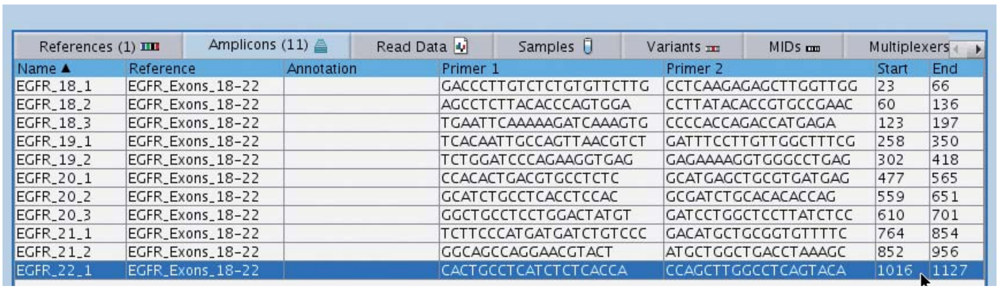
```
258 ....:....|....:....|....:....|....:....|....:....|
  1GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACA
 51CCCAGTGGAGAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGAAAC
101TGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGCGTTCGGCACGGTGT
151ATAAGGTAAGGTCCCTGGCACAGGCCTCTGGGCTGGGCCGCAGGGCCTCT
201CATGGTCTGGTGGGGNNNNNNNNNNNNNNNNNNNNNTCACAATTGCCAGTT
251AACGTCTTCCTTCTCTCTCTGTCATAGGGACTCTGGATCCCAGAAGGTGA
301GAAAGTTAAAATTCCCGTCGCTATCAAGGAATTAAGAGAAGCAACATCTC
351CGAAAGCCAACAAGGAAATCCTCGATGTGAGTTTCTGCTTTGCTGTGTGG
401GGGTCCATGGCTCTGAACCTCAGGCCCACCTTTTCTCNNNNNNNNNNNNN
451NNNNNNNCCACACTGACGTGCCTCTCCCTCCCTCCAGGAAGCCTACGTGA
501TGGCCAGCGTGGACAACCCCCACGTGTGCCGCCTGCTGGGCATCTGCCTC
551ACCTCCACCGTGCAGCTCATCACGCAGCTCATGCCCTTCGGCTGCCTCCT
```

Primer1 = TCACAATTGCCAGTTAACGTCT
Primer2 = GATTTCCTTGTTGGCTTTCG
Primer2' = CGAAAGCCAACAAGGAAATC

Legend
Primer match
Primer mismatch
Target sequence
Unused sequence

Figure 10–10: The Edit Start/End window for Amplicon EGFR_19_1, showing the two Primers with yellow background and the Target with blue background

After setting the Start and End points for the Targets within all Amplicons, the Amplicons Definition Table of out Project looks as shown in Figure 10–11.



Name ▲	Reference	Annotation	Primer 1	Primer 2	Start	End
EGFR_18_1	EGFR_Exons_18-22		GACCCTTGTCTCTGTGTTCTTG	CCTCAAGAGAGCTTGGTTGG	23	66
EGFR_18_2	EGFR_Exons_18-22		AGCCTCTTACACCCAGTGGA	CCTTATACACCGTGCCGAAC	60	136
EGFR_18_3	EGFR_Exons_18-22		TGAATTCAAAAAGATCAAAGTG	CCCCACCAGACCATGAGA	123	197
EGFR_19_1	EGFR_Exons_18-22		TCACAATTGCCAGTTAACGTCT	GATTTCCTTGTTGGCTTTCG	258	350
EGFR_19_2	EGFR_Exons_18-22		TCTGGATCCCAGAAGGTGAG	GAGAAAAGGTGGGCCTGAG	302	418
EGFR_20_1	EGFR_Exons_18-22		CCACACTGACGTGCCTCTC	GCATGAGCTGCGTGATGAG	477	565
EGFR_20_2	EGFR_Exons_18-22		GCATCTGCCTCACCTCCAC	GCGATCTGCACACACCAG	559	651
EGFR_20_3	EGFR_Exons_18-22		GGCTGCCTCCTGGACTATGT	GATCCTGGCTCCTTATCTCC	610	701
EGFR_21_1	EGFR_Exons_18-22		TCTTCCCATGATGATCTGTCCC	GACATGCTGCGGTGTTTTC	764	854
EGFR_21_2	EGFR_Exons_18-22		GGCAGCCAGGAACGTACT	ATGCTGGCTGACCTAAAGC	852	956
EGFR_22_1	EGFR_Exons_18-22		CACTGCCTCATCTCTCACCA	CCAGCTTGGCCTCAGTACA	1016	1127

Figure 10–11: The completed Amplicons Definitions Table for the EGFR example experiment

10.2.5 Defining the Sample

Our example experiment is the simplest case, in which we have only one Sample: the single DNA source used to prepare the 11 Amplicon libraries covering the five exons from the EGFR gene, which we will search for sequence Variants (see section 10.1). (For more complicated situations, see section 10.6.1). To set up our single Sample in our Project, we click on the 'Sample' sub-tab (Sample Definition Table) and then click on the 'Add' button at the left of the tree view. This adds a Sample called 'Sample_1' to the Sample Definition Table. We will keep this Default Sample Name, and leave the 'Annotation' field empty. The AVA application window is now in the state shown in Figure 10–12.

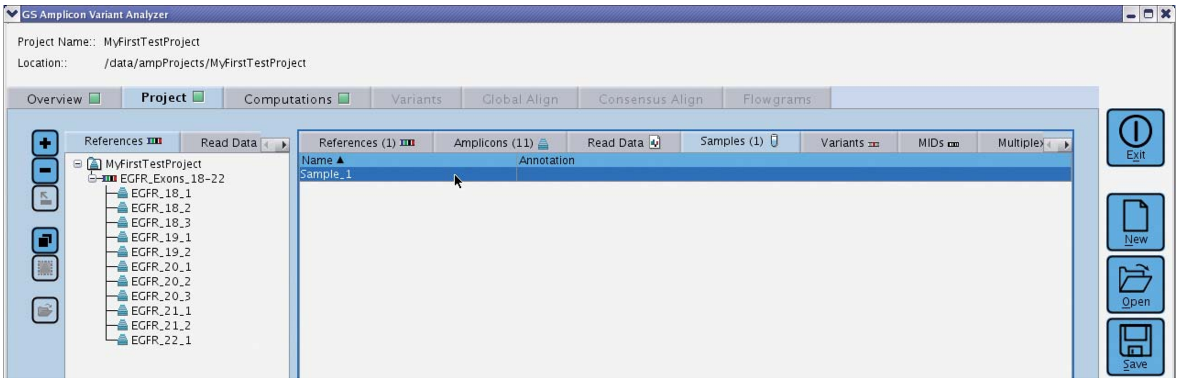
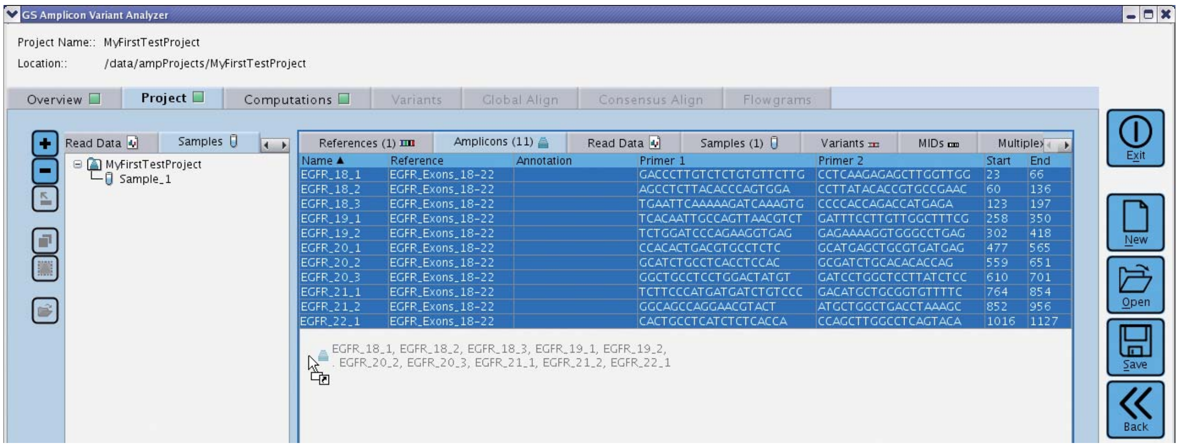


Figure 10–12: The AVA software window after creating a single Sample into the Project

We will next use the Tree sub-tabs to create the associations between the Sample and all 11 Amplicons defined in the Project. To do this, we select the Samples Tree sub-tab on the left panel (showing our single Sample hanging off the main project node), and the Amplicons Definition Table on the right panel. We then multi-select the full set of Amplicons from the Table (using the shift key) and drag them to Sample_1 in the Tree (Figure 10–13). This will associate all our Amplicons with our Sample (Figure 10–14).



Name ▲	Reference	Annotation	Primer 1	Primer 2	Start	End
EGFR_18_1	EGFR_Exons_18-22		GACCCTTGTCTCTGTGTTCTTG	CCTCAAGAGAGCTTGGTTGG	23	66
EGFR_18_2	EGFR_Exons_18-22		AGCCTCTTACACCCAGTGGA	CCTTATACACCGTGCCGAAC	60	136
EGFR_18_3	EGFR_Exons_18-22		TGAATTCAAAAAGATCAAAGTG	CCCCACCAGACCATGAGA	123	197
EGFR_19_1	EGFR_Exons_18-22		TCACAATTGCCAGTTAACGTCT	GATTTCCTTGTTGGCTTTCG	258	350
EGFR_19_2	EGFR_Exons_18-22		TCTGGATCCCAGAAGGTGAG	GAGAAAAGGTGGGCCTGAG	302	418
EGFR_20_1	EGFR_Exons_18-22		CCACACTGACGTGCCTCTC	GCATGAGCTGCGTGATGAG	477	565
EGFR_20_2	EGFR_Exons_18-22		GCATCTGCCTCACCTCCAC	GCGATCTGCACACACCAG	559	651
EGFR_20_3	EGFR_Exons_18-22		GGCTGCCTCCTGGACTATGT	GATCCTGGCTCCTTATCTCC	610	701
EGFR_21_1	EGFR_Exons_18-22		TCTTCCCATGATGATCTGTCCC	GACATGCTGCGGTGTTTTC	764	854
EGFR_21_2	EGFR_Exons_18-22		GGCAGCCAGGAACGTACT	ATGCTGGCTGACCTAAAGC	852	956
EGFR_22_1	EGFR_Exons_18-22		CACTGCCTCATCTCTCACCA	CCAGCTTGGCCTCAGTACA	1016	1127

EGFR_18_1, EGFR_18_2, EGFR_18_3, EGFR_19_1, EGFR_19_2,
. EGFR_20_2, EGFR_20_3, EGFR_21_1, EGFR_21_2, EGFR_22_1

Figure 10–13: The AVA window, in the middle of a multi-select and drag of the Amplicons from their Definition Table to the Sample_1 node in the Samples Tree, to create the associations

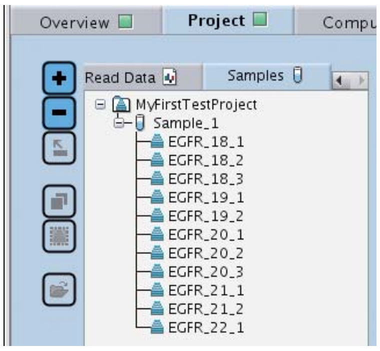
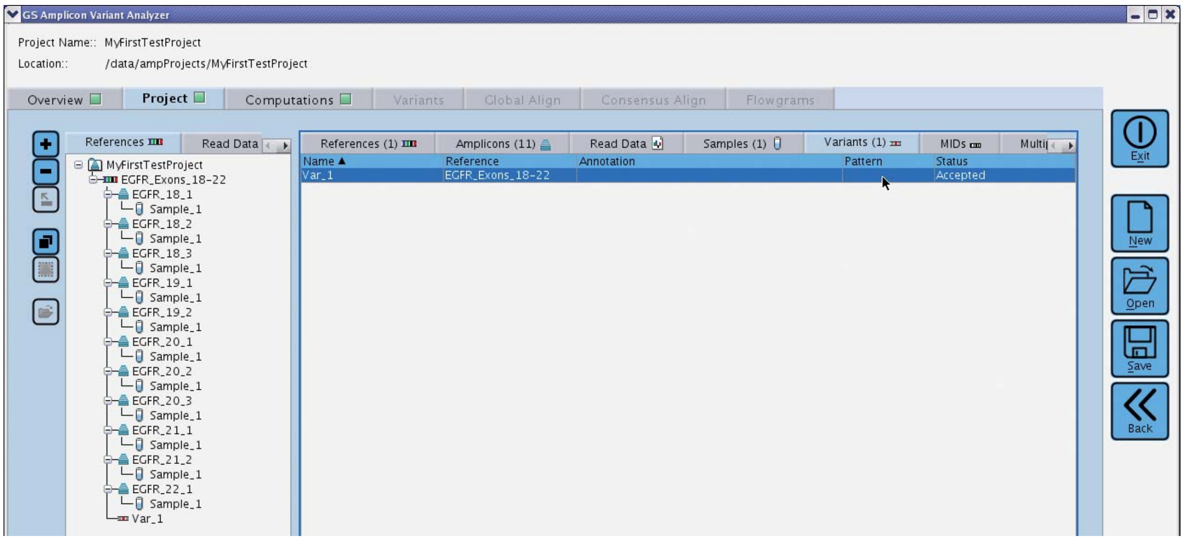


Figure 10–14: The Samples Tree after the 11 Amplicons have been associated with Sample_1

10.2.6 Defining the Known Variant

As mentioned at the beginning of this example (see section 10.1), there is a known 15 bp deletion Variant in exon 19 whose frequency we want to evaluate in our Sample. The corresponding sequence is in the “artificial” ‘EGFR_Exons_18-22’ Reference Sequence we already defined in our Project, so we can proceed with the definition of the Variant.

First, we click on the ‘Variants’ sub-tab on the right-hand panel of the ‘Project’ tab (the Variants Definition Table) and then choose the ‘Add’ button at the left margin. This creates a new entry in the Variant Definition Table. We decide to use the generic name, Var_1, for our Variant and to not enter any ‘Annotation’. We are thus ready to associate the Variant to its Reference Sequence by click-and-dragging it to the ‘EGFR_Exons_18-22’ node on the References Tree. This fills in the ‘Reference’ field for this Variant and sets the Status to “Accepted” in the Definition Table, and also adds the Variant as a sub-node of the References Tree (Figure 10–15).



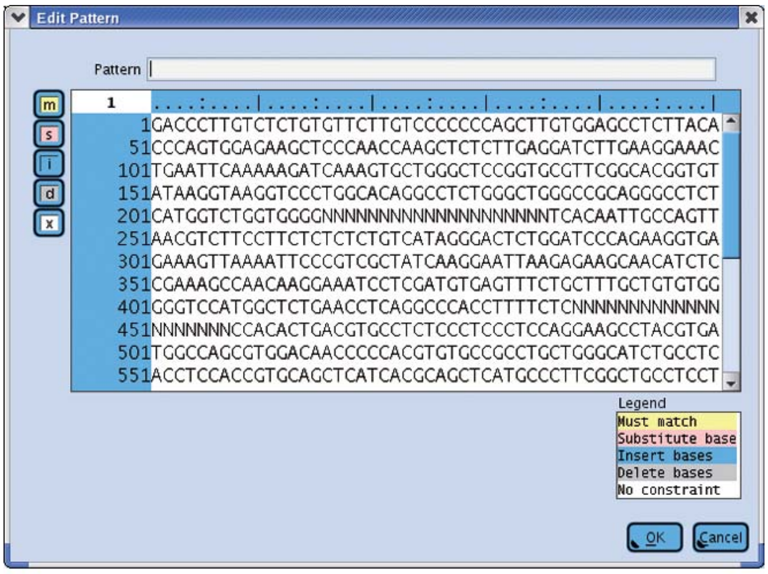
Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

```
MyFirstTestProject
  EGFR_Exons_18-22
    EGFR_18_1
      Sample_1
    EGFR_18_2
      Sample_1
    EGFR_18_3
      Sample_1
    EGFR_19_1
      Sample_1
    EGFR_19_2
      Sample_1
    EGFR_20_1
      Sample_1
    EGFR_20_2
      Sample_1
    EGFR_20_3
      Sample_1
    EGFR_21_1
      Sample_1
    EGFR_21_2
      Sample_1
    EGFR_22_1
      Sample_1
    Var_1
```

Name ▲	Reference	Annotation	Pattern	Status
Var_1	EGFR_Exons_18-22			Accepted

Figure 10–15: The AVA window after creating a Variant and associating it to the ‘EGFR_Exons_18-22’ Reference Sequence

To complete the definition of our Variant, we must enter a ‘Pattern’ of variation for the Variant with respect to the Reference Sequence. To do this, we double-click on the ‘Pattern’ cell for the Variant in the Variants Definition Table. This opens the ‘Edit Pattern’ window, pre-loaded with the Reference Sequence to which the Variant is associated (Figure 10–16).

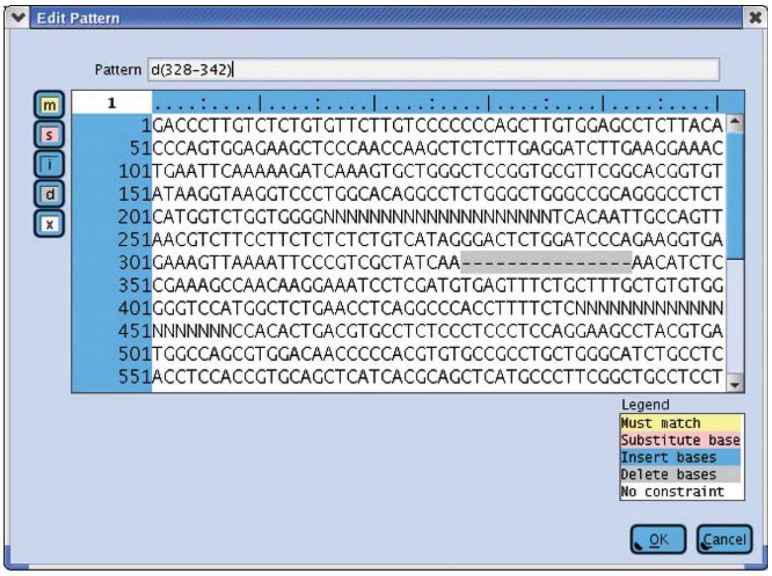


```
  1 ....:....|....:....|....:....|....:....|....:....|
  1GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACA
 51CCCAGTGGAGAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGAAAC
101TGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGCGTTCGGCACGGTGT
151ATAAGGTAAGGTCCCTGGCACAGGCCTCTGGGCTGGGCCGCAGGGCCTCT
201CATGGTCTGGTGGGGNNNNNNNNNNNNNNNNNNNNNTCACAATTGCCAGTT
251AACGTCTTCCTTCTCTCTCTGTCATAGGGACTCTGGATCCCAGAAGGTGA
301GAAAGTTAAAATTCCCGTCGCTATCAAGGAATTAAGAGAAGCAACATCTC
351CGAAAGCCAACAAGGAAATCCTCGATGTGAGTTTCTGCTTTGCTGTGTGG
401GGGTCCATGGCTCTGAACCTCAGGCCCACCTTTTCTCNNNNNNNNNNNNN
451NNNNNNNCCACACTGACGTGCCTCTCCCTCCCTCCAGGAAGCCTACGTGA
501TGGCCAGCGTGGACAACCCCCACGTGTGCCGCCTGCTGGGCATCTGCCTC
551ACCTCCACCGTGCAGCTCATCACGCAGCTCATGCCCTTCGGCTGCCTCCT
```

Legend
Must match
Substitute base
Insert bases
Delete bases
No constraint

Figure 10–16: The Edit Pattern window, pre-loaded with the ‘EGFR_Exons_18-22’ Reference Sequence and ready to receive the ‘Pattern’ for the Var_1 Variant

We decide to type the Pattern directly into the ‘Pattern’ text area of the window, using the AVA software’s Variant Definition Syntax. [Since the location of the deletion was initially defined relative to exon 19 (positions 93-107), we must first calculate its position in the artificial Reference Sequence we are using: 328-342.] So, we type: ‘d(328-342)’ and press ‘Enter’. This highlights the Variant pattern in the Reference sequence according to the ‘Legend’ at the lower right corner (in this case highlighting a string of gaps in gray to represent the deletion; Figure 10–17).



Pattern d(328-342)

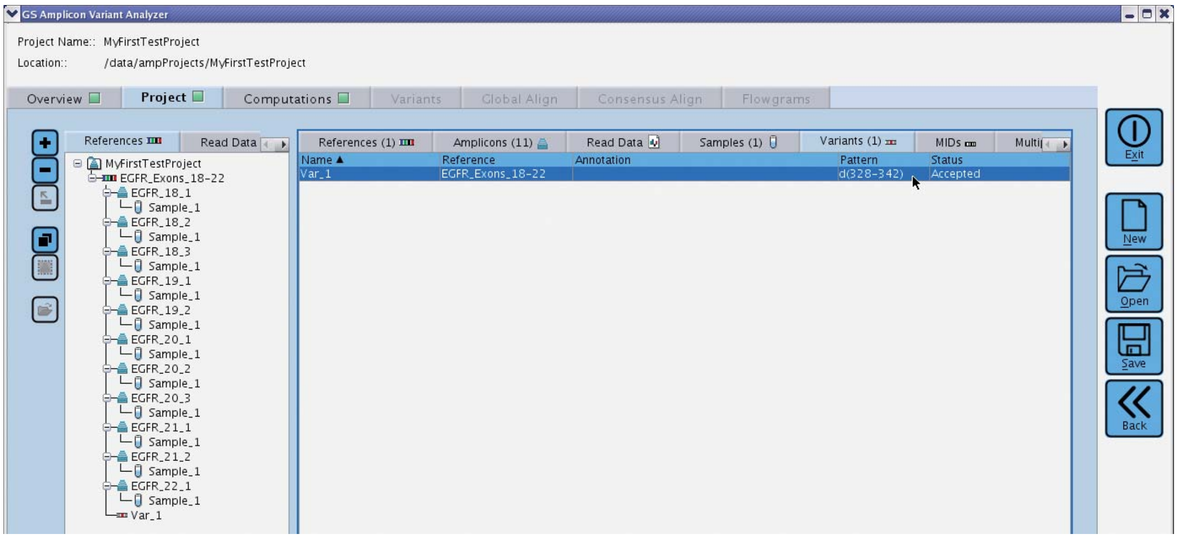
```
    1 ....:....|....:....|....:....|....:....|....:....|
    1GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACA
   51CCCAGTGGAGAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGAAAC
  101TGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGCGTTCGGCACGGTGT
  151ATAAGGTAAGGTCCCTGGCACAGGCCTCTGGGCTGGGCCGCAGGGCCTCT
  201CATGGTCTGGTGGGNNNNNNNNNNNNNNNNNNNNNTCACAATTGCCAGTT
  251AACGTCTTCCTTCTCTCTCTGTCATAGGGACTCTGGATCCCAGAAGGTGA
  301GAAAGTTAAAATTCCCGTCGCTATCAA---------------AACATCTC
  351CGAAAGCCAACAAGGAAATCCTCGATGTGAGTTTCTGCTTTGCTGTGTGG
  401GGGTCCATGGCTCTGAACCTCAGGCCCACCTTTTCTCNNNNNNNNNNNNN
  451NNNNNNNCCACACTGACGTGCCTCTCCCTCCCTCCAGGAAGCCTACGTGA
  501TGGCCAGCGTGGACAACCCCCACGTGTGCCGCCTGCTGGGCATCTGCCTC
  551ACCTCCACCGTGCAGCTCATCACGCAGCTCATGCCCTTCGGCTGCCTCCT
```

Legend

Must match
Substitute base
Insert bases
Delete bases
No constraint

Figure 10–17: The Edit Pattern window after entering the 15 bp deletion at positions 328-342 of the Reference Sequence. The nucleotides in question are replaced by dashes and are highlighted in gray, per the legend.

Clicking OK accepts the Pattern specification into the ‘Pattern’ field of the Variant in the Variant Definition Table (Figure 10–18).



Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

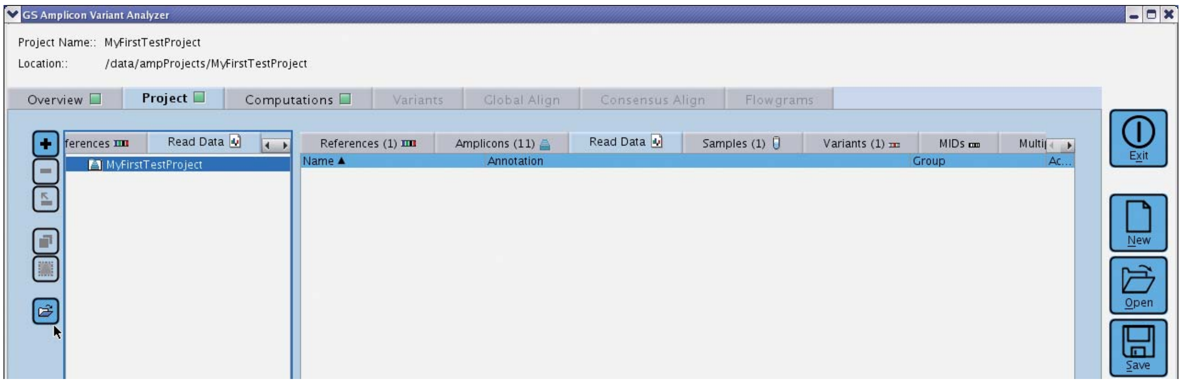
```
MyFirstTestProject
  EGFR_Exons_18-22
    EGFR_18_1
      Sample_1
    EGFR_18_2
      Sample_1
    EGFR_18_3
      Sample_1
    EGFR_19_1
      Sample_1
    EGFR_19_2
      Sample_1
    EGFR_20_1
      Sample_1
    EGFR_20_2
      Sample_1
    EGFR_20_3
      Sample_1
    EGFR_21_1
      Sample_1
    EGFR_21_2
      Sample_1
    EGFR_22_1
      Sample_1
    Var_1
```

Name ▲	Reference	Annotation	Pattern	Status
Var_1	EGFR_Exons_18-22		d(328-342)	Accepted

Figure 10–18: The AVA window after fully defining the Variant “Var_1”

10.2.7 Importing the Read Data Set

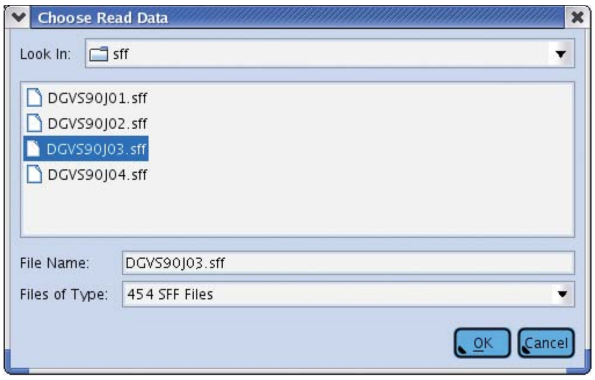
The next and final step in the set up of the Project is to add actual read data. This is done using the "Import" button at the left edge of the Project tab. This button is enabled by selecting either the 'Read Data' Tree sub-tab (left panel) or the 'Read Data' Definition Table sub-tab (right panel) (Figure 10–19).



Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject
MyFirstTestProject

Figure 10–19: The AVA window with the Read Data Tree and Definition Table selected. Note that the Import button to the left is now active. The panel clicked on last has the blue "active panel" border around it; in this case, it is the tree panel.

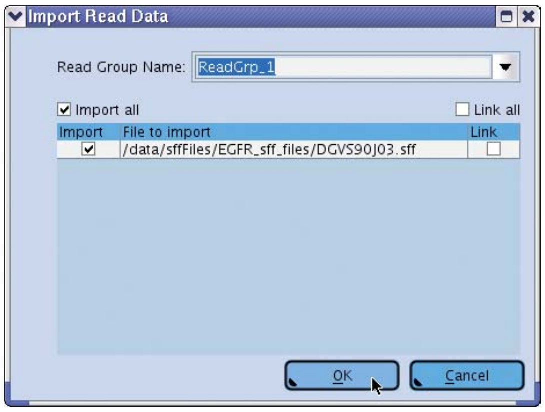
Clicking the "Import" button opens the "Choose Read Data" file browser window, which allows us to search for Read Data files to add to the Project. Since the data we want to import resides in a single region of a 4-region sequencing Run (and each region has its own SFF file), we select '454 SFF Files' from the 'Files of Type:' drop down menu. We then navigate to the folder that contains the SFF files of the EGFR Run, and select the file that contains the read data we want to import (DGVS90J03.sff). This selection populates the 'File Name:' field, in the Choose Read Data window (Figure 10–20).



Look In: sff
DGVS90J01.sff
DGVS90J02.sff
DGVS90J03.sff
DGVS90J04.sff
File Name: DGVS90J03.sff
Files of Type: 454 SFF Files

Figure 10–20: The Choose Read Data window, with the DGVS90J03.sff file selected

Clicking “OK” opens the Import Read Data window. We choose to use the default ‘Read Group Name’, and to import the data file itself (as opposed to simply creating a symbolic link) (Figure 10–21).



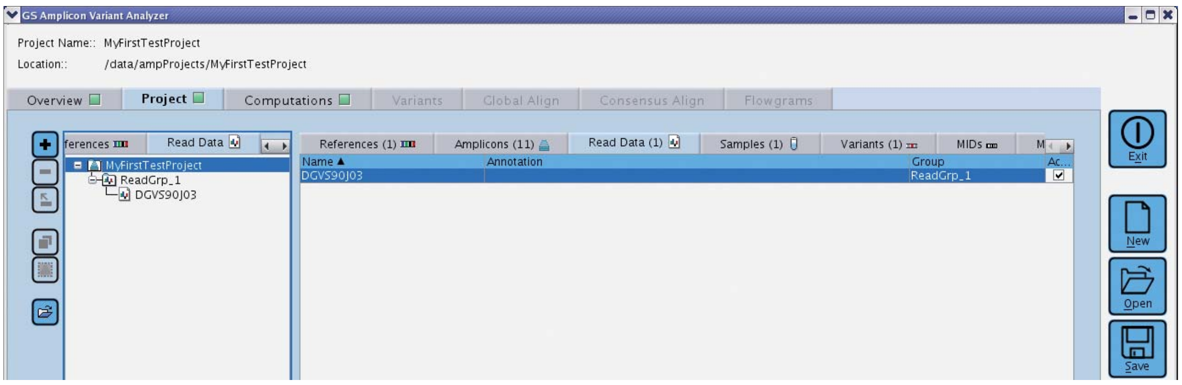
Read Group Name: ReadGrp_1

☑ Import all ☐ Link all

Import	File to import	Link
☑	/data/sffFiles/EGFR_sff_files/DGVS90J03.sff	☐

Figure 10–21: The Import Read Data window, ready to import the file selected

Clicking OK returns us to the AVA window and adds the new Read Data Set to both the Read Data Tree and the Read Data Table (Figure 10–22).



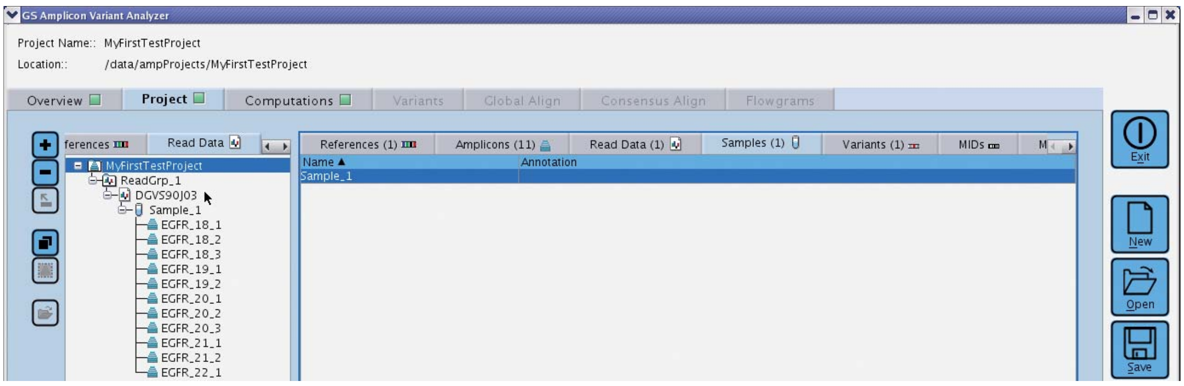
Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

MyFirstTestProject
ReadGrp_1
DGVS90J03

Name ▲	Annotation	Group	Ac...
DGVS90J03		ReadGrp_1	✓

Figure 10–22: The AVA window, after importing the DGVS90J03.sff Read Data file

Finally, we must associate the Sample-Amplicon groups with the Read Data, so the AVA software can properly demultiplex the reads in the Read Data and assign them to their respective Amplicons. To do this, we select the Read Data Tree and the Samples Definition Table, and drag the Sample_1 to the DGVS90J03 Read Data node. This creates the association between the Sample and the Read Data, with the prior Sample-Amplicon associations also maintained (Figure 10–23). We then click the ‘Save’ button to save all the information we entered, in the Project folder.



Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

MyFirstTestProject
ReadGrp_1
DGVS90J03
Sample_1
EGFR_18_1
EGFR_18_2
EGFR_18_3
EGFR_19_1
EGFR_19_2
EGFR_20_1
EGFR_20_2
EGFR_20_3
EGFR_21_1
EGFR_21_2
EGFR_22_1

Name ▲	Annotation
Sample_1	

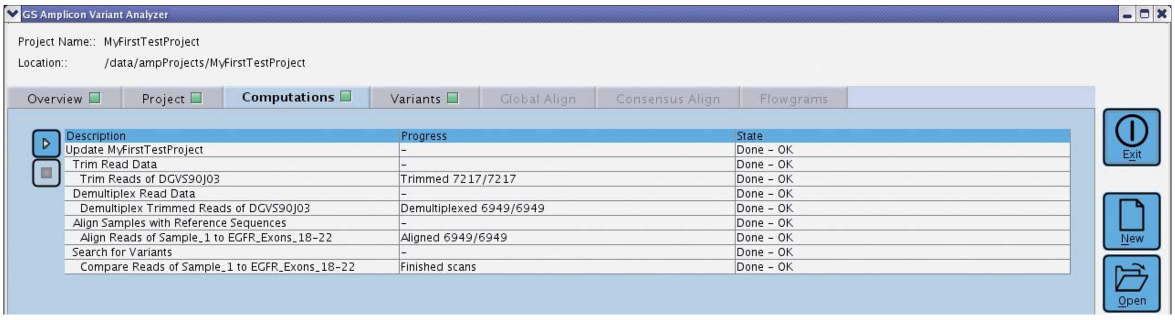
Figure 10–23: The AVA window after creating the association between the Sample_1 and the DGVS90J03 Read Data Set

10.3 Analysis of Known Variants

With the Project fully defined, we can now process ("compute") the Read Data and search for our known Variant, the 15 bp deletion in EGFR exon 19.

10.3.1 Compute the Project

To carry out the computation, we select the Computations tab and click on the "Start Computation" button. Status messages allow us to track progress. Computation is complete when all the 'State' messages say "Done - OK" and the "Start Computations" button is no longer grayed-out (Figure 10–24).



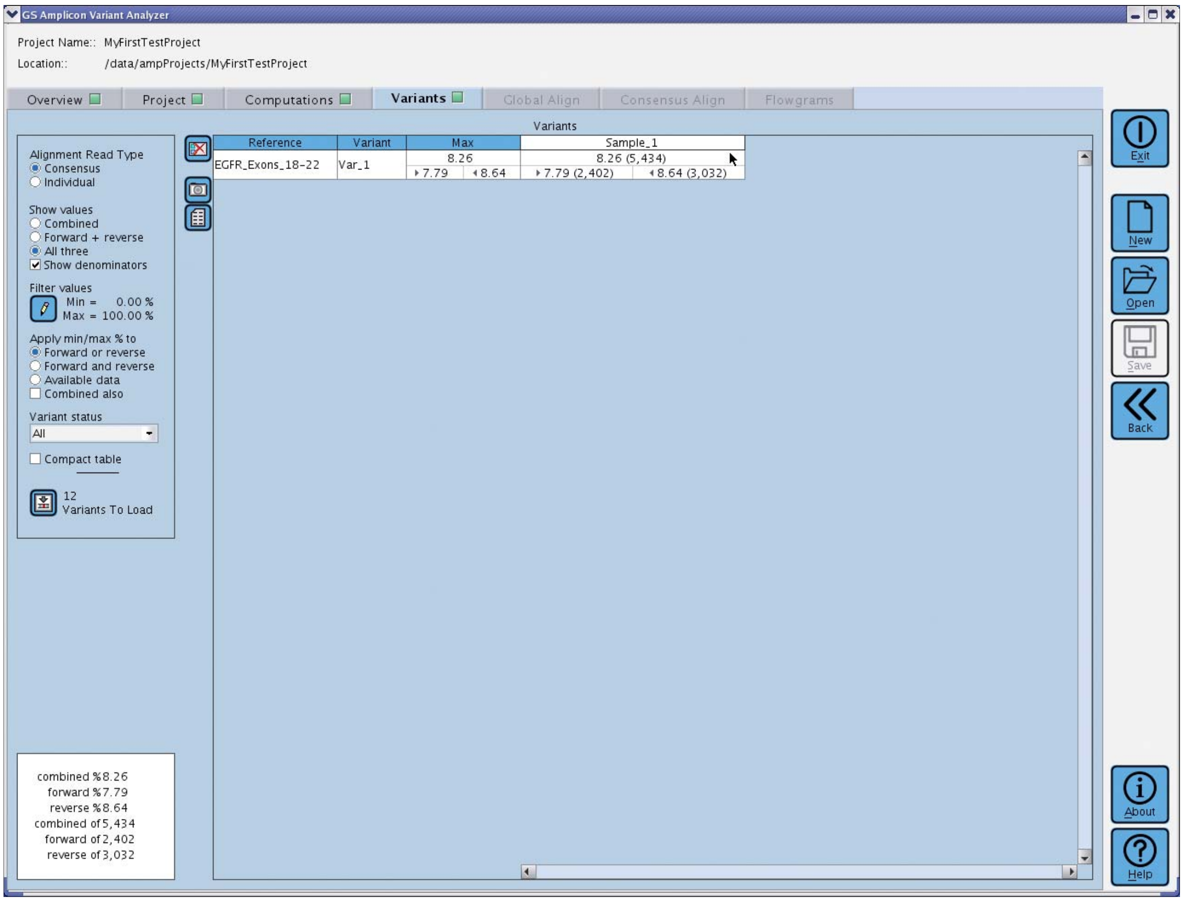
Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Description	Progress	State
Update MyFirstTestProject	–	Done – OK
Trim Read Data	–	Done – OK
Trim Reads of DGVS90J03	Trimmed 7217/7217	Done – OK
Demultiplex Read Data	–	Done – OK
Demultiplex Trimmed Reads of DGVS90J03	Demultiplexed 6949/6949	Done – OK
Align Samples with Reference Sequences	–	Done – OK
Align Reads of Sample_1 to EGFR_Exons_18-22	Aligned 6949/6949	Done – OK
Search for Variants	–	Done – OK
Compare Reads of Sample_1 to EGFR_Exons_18-22	Finished scans	Done – OK

Figure 10–24: The Computation tab, with the computation complete

10.3.2 Frequency of Known Variants

A green square appears on the Variants tab after completion of a computation that included at least one known (or auto-detected) Variant, which is our case. We click on the Variants tab to observe the results of the analysis (Figure 10–25). We choose to display the frequency and the number of reads of the Variant in the forward, reverse and combined orientations (‘All three’ and ‘Show denominators’ settings under “Show values”), to ascertain that the occurrence of the Variant isn’t orientation-dependent; the fact that it isn’t makes the observation more credible (verification of support in both orientations is helpful to eliminate false-positives that may occur due to artifacts in alignment or sequencing).



Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Alignment Read Type
(●) Consensus
( ) Individual

Show values
( ) Combined
( ) Forward + reverse
(●) All three
[✓] Show denominators

Filter values
Min = 0.00 %
Max = 100.00 %

Apply min/max % to
(●) Forward or reverse
( ) Forward and reverse
( ) Available data
[ ] Combined also

Variant status
All

[ ] Compact table

12
Variants To Load

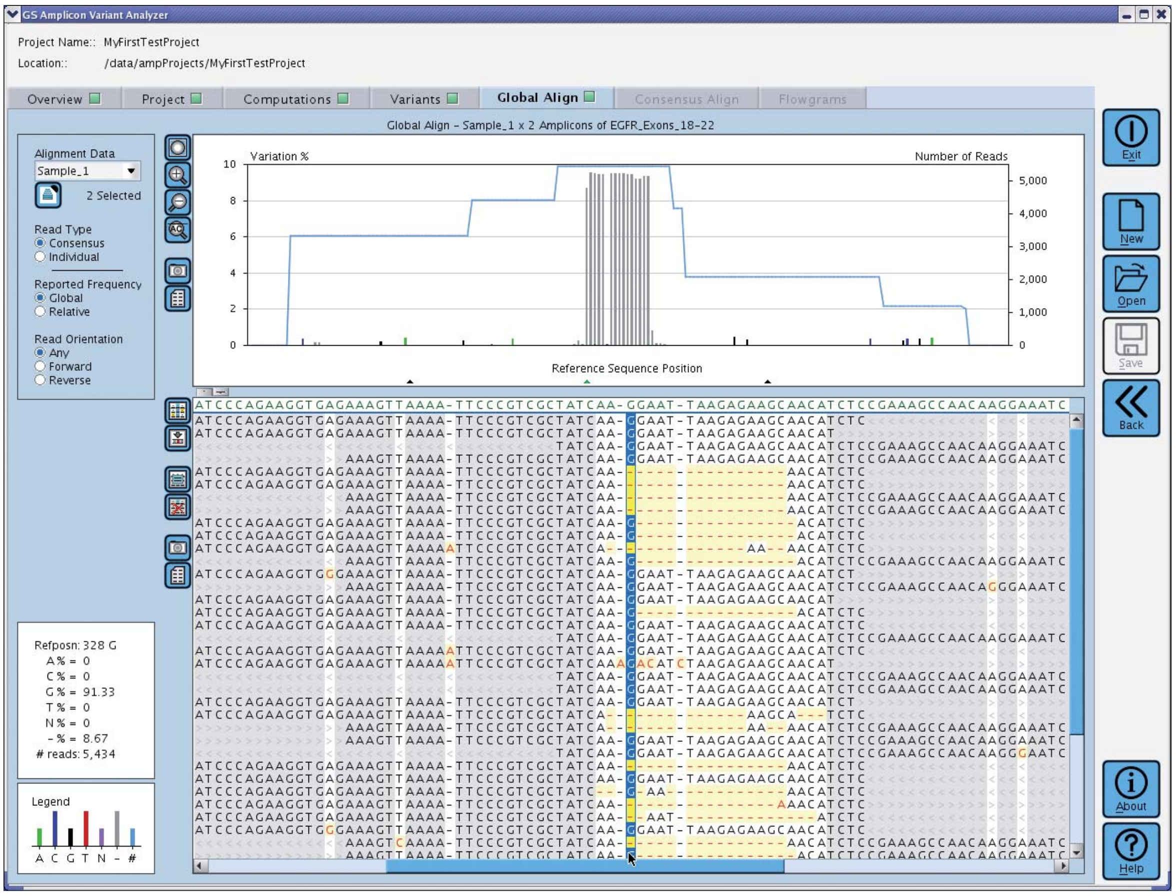
Variants

Reference	Variant	Max		Sample_1	
EGFR_Exons_18-22	Var_1	8.26		8.26 (5,434)	
		‣7.79	◂8.64	‣7.79 (2,402)	◂8.64 (3,032)

combined %8.26
forward %7.79
reverse %8.64
combined of 5,434
forward of 2,402
reverse of 3,032

Figure 10–25: The Variants tab after completion of the computation, showing the frequency of the Var_1 Variant in the reads included in the Sample_1 Sample

To further explore our known Variant, we right-click on the cell that intersects with Variant 'Var_1' and 'Sample_1', in the Variants tab, and choose 'Global Align' from the contextual menu. This loads all the reads corresponding to all the Amplicons that cover this Variant, in Sample_1 (Amplicons EGFR_19-1 and EGFR_19-2; see Figure 10–1, above), and displays them in the Global Align tab (Figure 10–26).



Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Alignment Data: Sample_1
2 Selected

Read Type
[✓ Consensus]
Individual

Reported Frequency
[✓ Global]
Relative

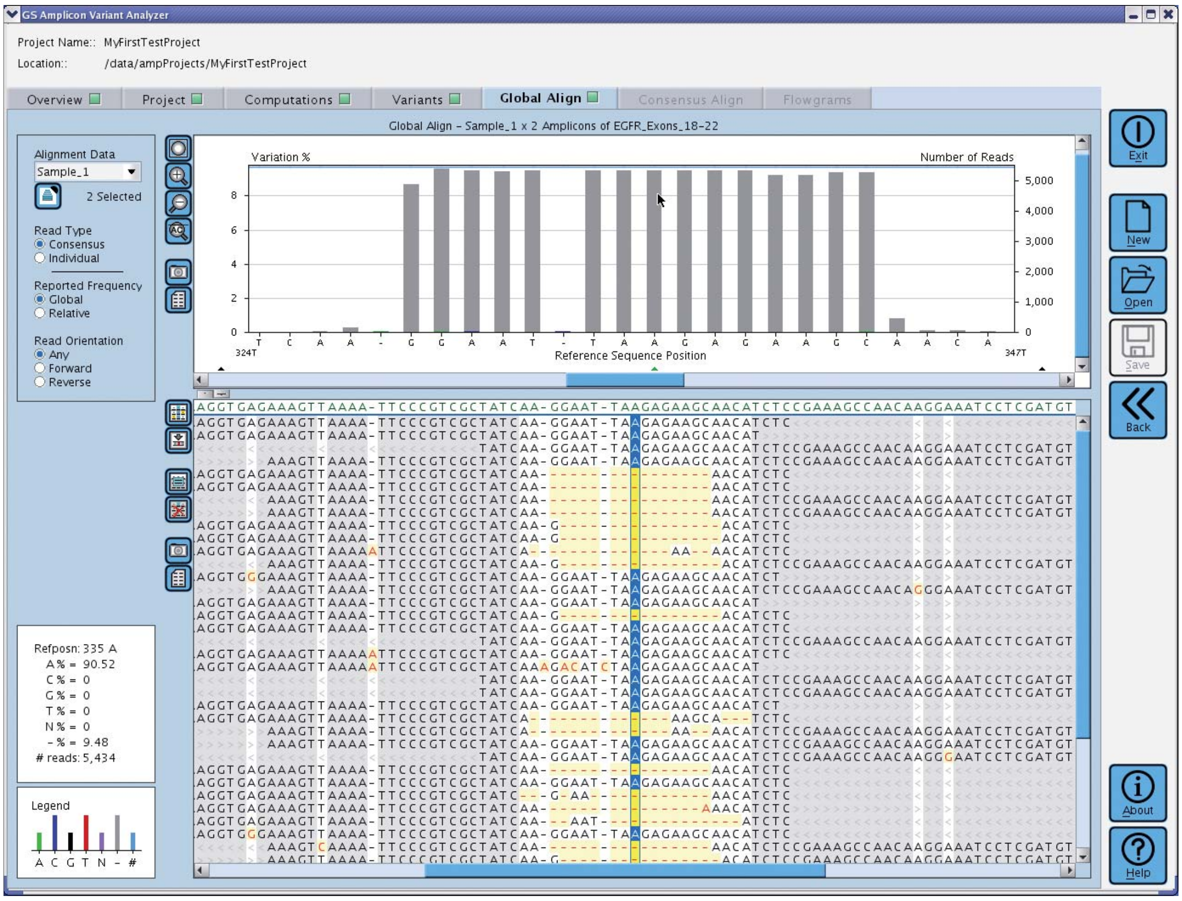
Read Orientation
[✓ Any]
Forward
Reverse

```
Refposn: 328 G
  A % = 0
  C % = 0
  G % = 91.33
  T % = 0
  N % = 0
  - % = 8.67
# reads: 5,434
```

```
ATCCCAGAAGGTGAGAAAGTTAAAA-TTCCCGTCGCTATCAA-GGAAT-TAAGAGAAGCAACATCTCCGAAAGCCAACAAGGAAATC
```

Figure 10–26: The Global Align tab, loaded with the reads relevant to Variant Var_1 in Sample_1

The Variant appears as a cluster of gray bars (indicating "gaps", *i.e.* a deletion) near the middle of the Variation Frequency Plot (the top panel). To get a better look, we draw a rectangle with the mouse, around the zone of interest in the Plot (this is the "Freehand Zoom In" tool). Then, by clicking on one of the gray bars in the plot, we can re-center the multi-alignment Table to the corresponding area (Figure 10–27).



Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Global Align - Sample_1 x 2 Amplicons of EGFR_Exons_18-22

Alignment Data: Sample_1
2 Selected

Read Type: (●) Consensus ( ) Individual

Reported Frequency: (●) Global ( ) Relative

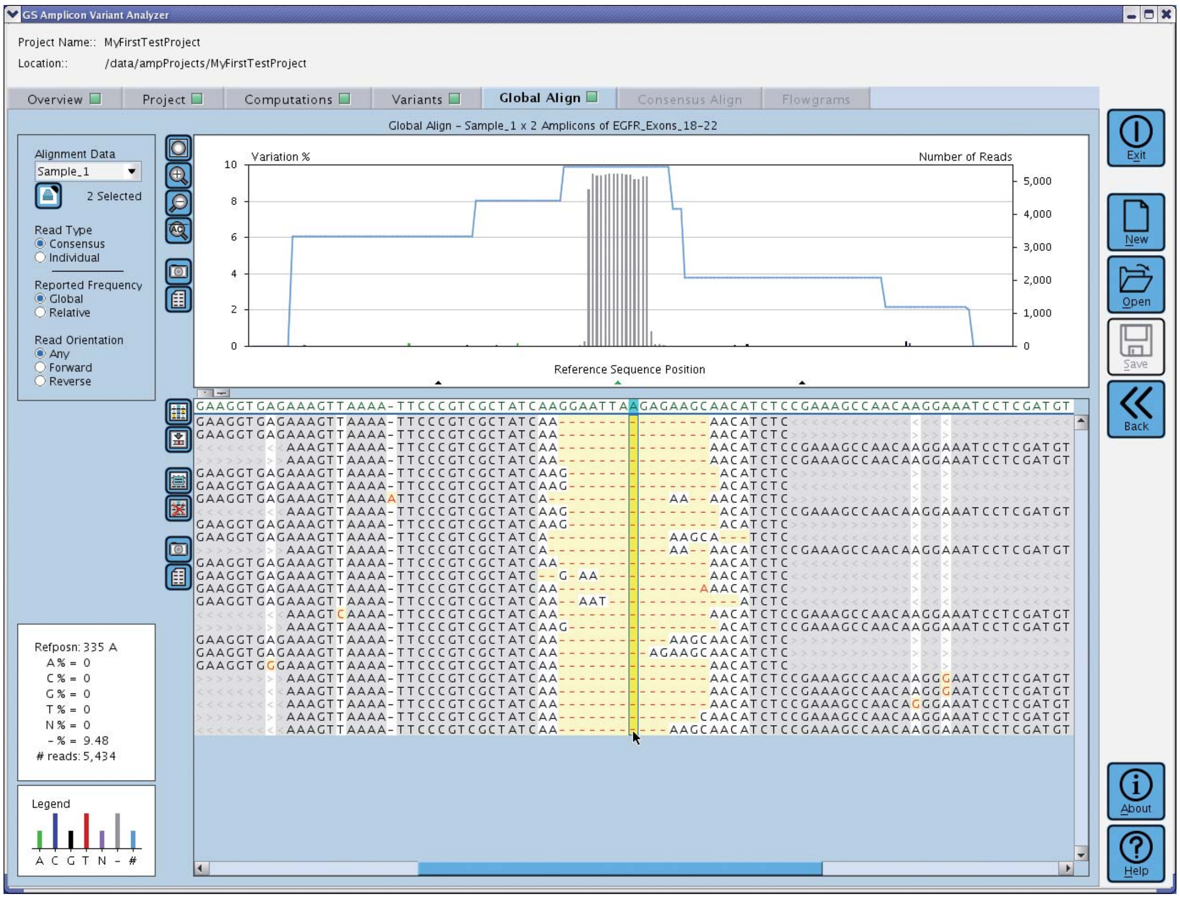
Read Orientation: (●) Any ( ) Forward ( ) Reverse

```
Refposn: 335 A
  A% = 90.52
  C% = 0
  G% = 0
  T% = 0
  N% = 0
  -% = 9.48
 # reads: 5,434
```

```
AGGTGAGAAAGTTAAAA-TTCCCGTCGCTATCAA-GGAAT-TAAGAGAAGCAACATCTCCGAAAGCCAACAAGGAAATCCTCGATGT
```

Figure 10–27: The Global Align tab for Var_1 in Sample_1, with the Variation Frequency Plot zoomed in around the deletion and the multi-alignment showing several consensi with a stretch of gaps

As can be seen, several of the consensi visible in the multi-alignment have many gaps in this region. To explore these in particular, we can select for viewing only the consensi with these gaps. This is done by right-clicking on a base (of any consensus) in a column within the stretch of gaps, and selecting for the gap character '-' in the contextual menu. The result, shown in Figure 10–28, is that only the consensi that have a gap at the position on which the selection was made (position 335 of the Reference Sequence in this case) are now displayed in the multi-alignment, and the Variation Frequency Plot is adjusted accordingly. Note in particular that the frequency axis (Variation %) is automatically re-scaled to best fit the data displayed, allowing us to clearly see that all the nucleotide positions in the stretch have the gap at a fairly consistent frequency, an observation consistent with a valid Variant. Note also that the frequency of 9.48% is close to, but a little on the high side of the value seen in the Variants Frequency Table for Var_1 (8.26%). The difference occurs because we made only one selection to focus the plot on the deletion area; not all the reads being displayed perfectly match our defined Variant. In part this is because some consensus reads have basecalling/alignment problems that keep them from being counted as part of the Variant for the Variants Table frequency calculation. But more significantly, in this case, there is another deletion variant present that, compared with Var_1, is shifted by a single base and is present at 0.81% (see the full list of automatically detected variants in Figure 10–44, below).



Project Name:: MyFirstTestProject

Location:: /data/ampProjects/MyFirstTestProject

Alignment Data: Sample_1 — 2 Selected

Read Type: [✓ Consensus] Individual

Reported Frequency: [✓ Global] Relative

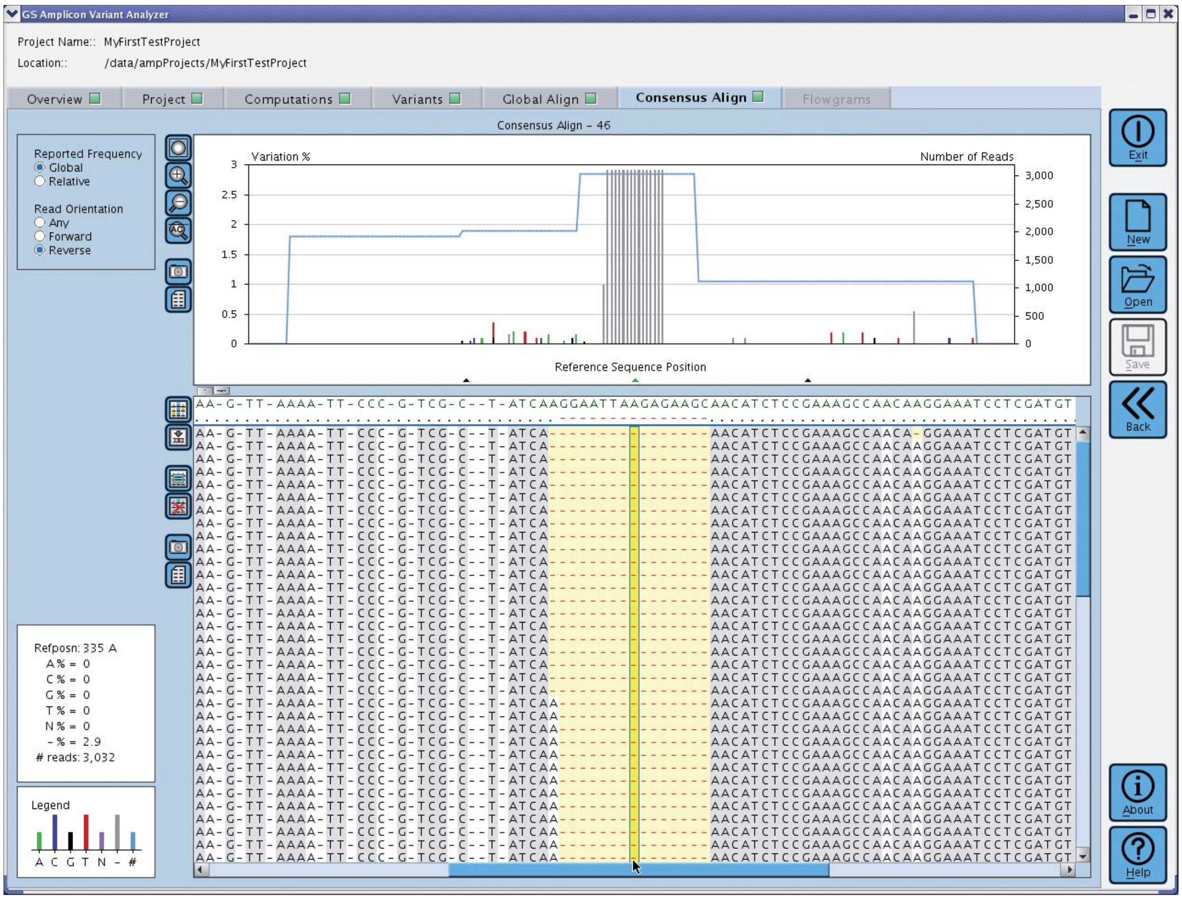
Read Orientation: [✓ Any] Forward Reverse

```
Refposn: 335 A
  A % = 0
  C % = 0
  G % = 0
  T % = 0
  N % = 0
  - % = 9.48
# reads: 5,434
```

```
GAAGGTGAGAAAGTTAAAA-TTCCCGTCGCTATCAAGGAATTAAGAGAAGCAACATCTCCGAAAGCCAACAAGGAAATCCTCGATGT
GAAGGTGAGAAAGTTAAAA-TTCCCGTCGCTATCAA---------------AACATCTC<<<<<<<<<<<<<<<<<<<<<<<<<<<<
GAAGGTGAGAAAGTTAAAA-TTCCCGTCGCTATCAA---------------AACATCTC>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<<<<<<<<AAAGTTAAAA-TTCCCGTCGCTATCAA---------------AACATCTCCGAAAGCCAACAAGGAAATCCTCGATGT
>>>>>>>>>AAAGTTAAAA-TTCCCGTCGCTATCAA---------------AACATCTCCGAAAGCCAACAAGGAAATCCTCGATGT
GAAGGTGAGAAAGTTAAAA-TTCCCGTCGCTATCAAG---------------ACATCTC>>>>>>>>>>>>>>>>>>>>>>>>>>>>
GAAGGTGAGAAAGTTAAAA-TTCCCGTCGCTATCAAG---------------ACATCTC<<<<<<<<<<<<<<<<<<<<<<<<<<<<
GAAGGTGAGAAAGTTAAAAATTCCCGTCGCTATCA------------AA--AACATCTC>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<<<<<<<<AAAGTTAAAA-TTCCCGTCGCTATCAAG---------------ACATCTCCGAAAGCCAACAAGGAAATCCTCGATGT
GAAGGTGAGAAAGTTAAAA-TTCCCGTCGCTATCAAG---------------ACATCTC>>>>>>>>>>>>>>>>>>>>>>>>>>>>
GAAGGTGAGAAAGTTAAAA-TTCCCGTCGCTATCA------------AAGCA---TCTC<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>AAAGTTAAAA-TTCCCGTCGCTATCA------------AA--AACATCTCCGAAAGCCAACAAGGAAATCCTCGATGT
GAAGGTGAGAAAGTTAAAA-TTCCCGTCGCTATCAA---------------AACATCTC<<<<<<<<<<<<<<<<<<<<<<<<<<<<
GAAGGTGAGAAAGTTAAAA-TTCCCGTCGCTATC--G-AA-----------AACATCTC<<<<<<<<<<<<<<<<<<<<<<<<<<<<
GAAGGTGAGAAAGTTAAAA-TTCCCGTCGCTATCAA--------------AAACATCTC>>>>>>>>>>>>>>>>>>>>>>>>>>>>
GAAGGTGAGAAAGTTAAAA-TTCCCGTCGCTATCAA--AAT-------------ATCTC<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<AAAGTCAAAA-TTCCCGTCGCTATCAA---------------AACATCTCCGAAAGCCAACAAGGAAATCCTCGATGT
>>>>>>>>>AAAGTTAAAA-TTCCCGTCGCTATCAAG---------------ACATCTCCGAAAGCCAACAAGGAAATCCTCGATGT
GAAGGTGAGAAAGTTAAAA-TTCCCGTCGCTATCAA-----------AAGCAACATCTC>>>>>>>>>>>>>>>>>>>>>>>>>>>>
GAAGGTGAGAAAGTTAAAA-TTCCCGTCGCTATCAA---------AGAAGCAACATCTC>>>>>>>>>>>>>>>>>>>>>>>>>>>>
GAAGGTGGGAAAGTTAAAA-TTCCCGTCGCTATCAA---------------AACATCTC>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>AAAGTTAAAA-TTCCCGTCGCTATCAA---------------AACATCTCCGAAAGCCAACAAGGGAATCCTCGATGT
<<<<<<<<<AAAGTTAAAA-TTCCCGTCGCTATCAA---------------AACATCTCCGAAAGCCAACAAGGGAATCCTCGATGT
<<<<<<<<<AAAGTTAAAA-TTCCCGTCGCTATCAA---------------AACATCTCCGAAAGCCAACAGGGAAATCCTCGATGT
>>>>>>>>>AAAGTTAAAA-TTCCCGTCGCTATCAA--------------CAACATCTCCGAAAGCCAACAAGGAAATCCTCGATGT
<<<<<<<<<AAAGTTAAAA-TTCCCGTCGCTATCAA-----------AAGCAACATCTCCGAAAGCCAACAAGGAAATCCTCGATGT
```

Figure 10–28: The Global Align tab for Var_1 in Sample_1, with a selection for gaps applied to position 335 of the Reference Sequence (in the stretch of gaps)

We will now dig further by examining the individual reads that comprise the third of these consensi. To do this, we right-click on a nucleotide of this consensus (we choose the same position 335 to keep the focus at the same location) and select the ‘open consensus alignment’ option from the contextual menu. This loads the Consensus Align tab with a multi-alignment of all the reads that contributed to the consensus we just selected (Figure 10–29). This view shows that certain reads lack an extra “A” nucleotide that is present in the rest of them. Looking at the sequence carefully we notice that the deletion has created a homopolymer of “A”, suggesting that the minority “gap extension” may actually be due to an undercall of this homopolymer in the reads that show it; in support of this, we note that this homopolymer is especially observed in reads in the reverse orientation (as shown in Figure 10–29), which places an environment very rich in “A” nucleotides just before the gap.



Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Reported Frequency
[✓ Global] Relative

Read Orientation
Any Forward [✓ Reverse]

Refposn: 335 A
A % = 0
C % = 0
G % = 0
T % = 0
N % = 0
- % = 2.9
# reads: 3,032

```
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCAAGGAATTAAGAGAAGCAACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACA-GGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCA----------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCAA---------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCAA---------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCAA---------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCAA---------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCAA---------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCAA---------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCAA---------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCAA---------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCAA---------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCAA---------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCAA---------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCAA---------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
AA-G-TT-AAAA-TT-CCC-G-TCG-C--T-ATCAA---------------AACATCTCCGAAAGCCAACAAGGAATCCTCGATGT
```

Figure 10–29: The Consensus Align tab for the reads included in the third consensus of the Var_1 in the Sample_1 global alignment shown in Figure 10–28

Finally, we go to the finest level of detail by examining the flowgram of the first read in this multi-alignment. Again, we right-click on the nucleotide position corresponding to position 335 of the Reference Sequenceto conveniently keep the focus at the same location, and we select the 'Open Flowgrams' option from the contextual menu. This loads the Flowgrams tab with the flowgram data for the read on which we clicked (Figure 10–30).

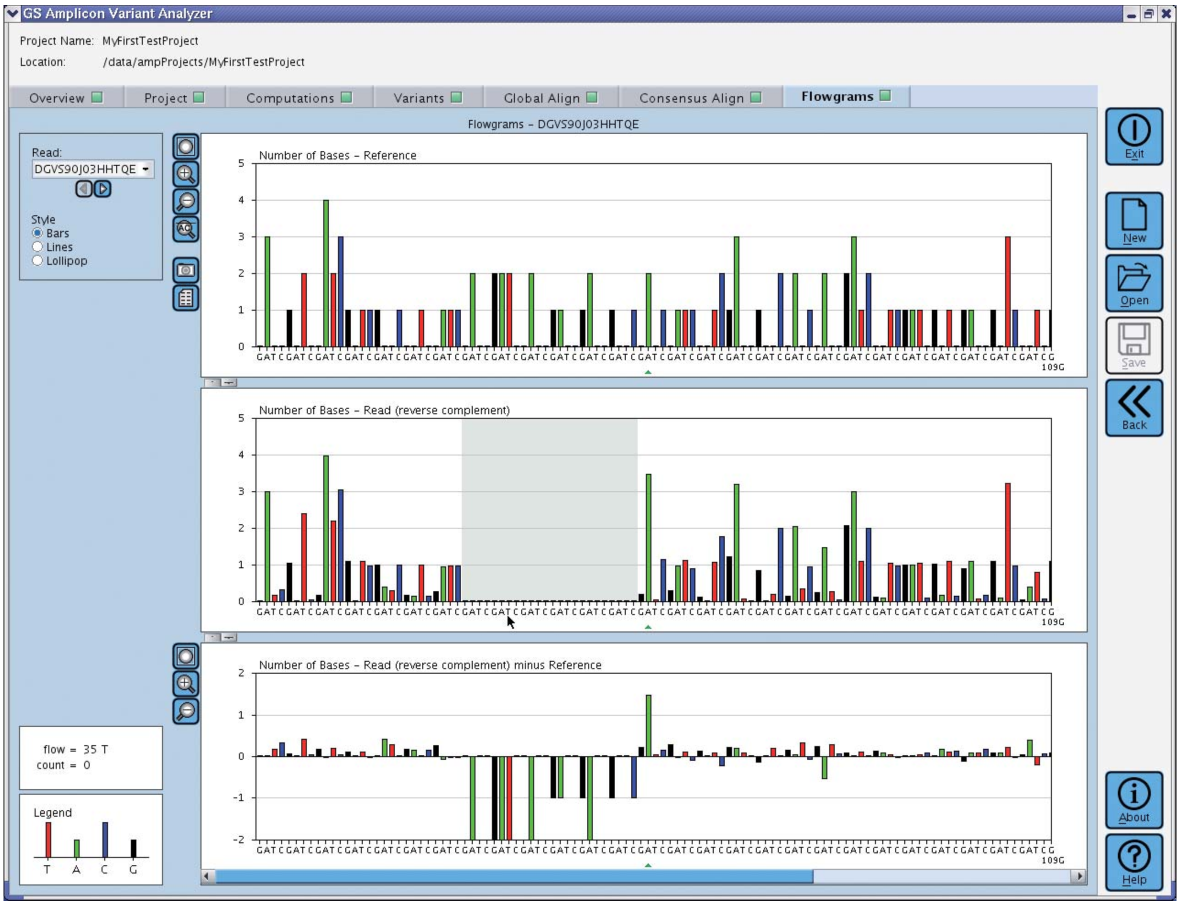

GS Amplicon Variant Analyzer
Project Name: MyFirstTestProject
Location: /data/ampProjects/MyFirstTestProject
Overview
Project
Computations
Variants
Global Align
Consensus Align
Flowgrams
Flowgrams – DGVS90J03HHTQE
Read:
DGVS90J03HHTQE
Style
Bars
Lines
Lollipop
Number of Bases – Reference
Number of Bases – Read (reverse complement)
Number of Bases – Read (reverse complement) minus Reference
109G
flow = 35 T
count = 0
Legend
T
A
C
G
Exit
New
Open
Save
Back
About
Help

Figure 10–30: The Flowgram tab for the first read on the third consensus of Var_1 in Sample 1 (in Figure 10–28), showing that a gap of several nucleotide flow cycles in the read allows it to maintain alignment with the Reference Sequence on both sides of the gap

We clearly see that in order to maintain the alignment with the Reference Sequence, the software introduced a gap of several nucleotide flow cycles at the position of the deletion (marked in gray in the read flowgram); this is very strong evidence for the presence of a true deletion. The elevated 'A' flow after the gap is caused by splicing together the two 'A' pairs on either side of the deletion into a single 'A' 4-mer.

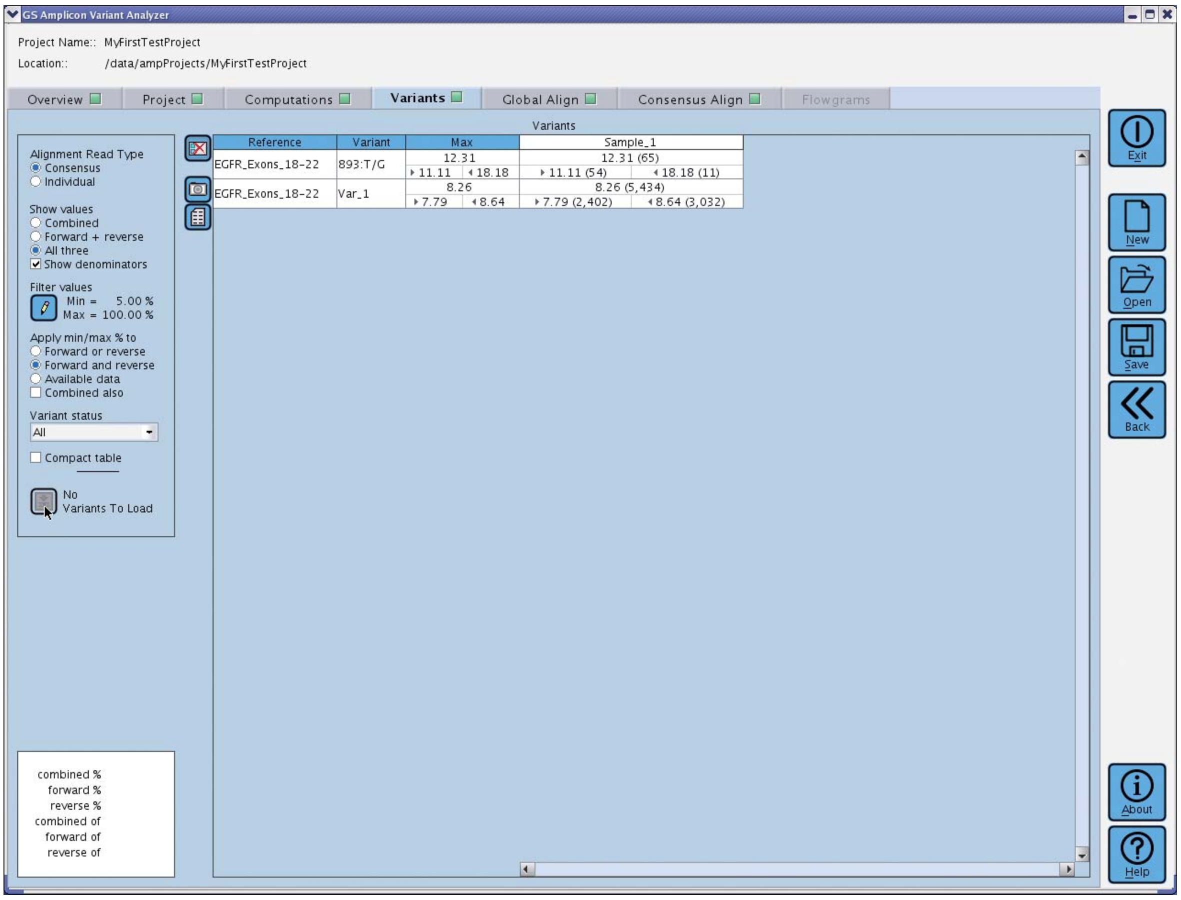
We can use the "arrow" buttons at the top left of the tab to "scroll" over the flowgrams of the reads present in the Consensus Align tab; this allows us to see how "stable" any particular flow or set of flows in the window seems to be. To make this easier, we can focus on one feature of the read on the difference flowgram as we scroll through the available flowgrams to see how the magnitude of the feature changes from read to read (for example, the green triangle below each flowgram can serve as a useful focus point during scrolling).

The initial deletion peaks in the graph on the Global Align tab plot were seen at a moderate percentage range (8.65–9.48%). The underlying alignments show that the deletions were linked together as a 15-bp deletion haplotype. The underlying flowgrams of reads exhibiting the haplotype further show that the deletions were not due to marginal calls, and demonstrate that the flows needed to be shifted to align properly. Taken together, the evidence is compelling that this 15 bp deletion is a true Variant in the sample. The 8.26% combined frequency for the Variant (on the Variants tab) is a conservative estimate that seeks to measure perfect instances of the defined Variant; this estimate relies on consensus reads that, by their combination of individual reads, can distort the frequency statistics. So the actual frequency of the variation in the Sample is likely higher than 8.26%. As seen in Figure 10–44, below, the combined Var_1 percentage, based on individual reads, is 8.70%, which is closer to the lower range of observed deletion peak values. Further inspection of the alignment suggests an overlapping deletion (see the 5th, 6th and other consensus lines of Figure 10–28 that end with a G just inside the deletion, rather than an A, and which end one base later inside the deletion, with a single A, rather than the double AA as occurs with Var_1). This additional deletion is reported as an automatically detected Variant (Figure 10–44) with a combined frequency of 0.81%, and helps explain the range of deletion peaks observed.

10.4 Mining a Project for New Variants

We started the project with only one predefined Variant. As part of the computation done to measure our defined Variant, the AVA software also examined the alignments in the Project to propose potential Variants. We can access them via the main Variants tab. If we look again at the view of the Variants tab in Figure 10–25, we can see that the "Load" button at the bottom left of the Variants Frequency Table filter control box states that there are 12 Variants to load. The automated Variant detector is sensitive and likely to include false positives, so it is wise to use some of the filters to narrow down the potential set of variants rather than just importing them all.

By setting the "Min" value to 5.00% and choosing the "Forward and reverse" filter on the Variants tab, the status of the "Load" button changes to show that there is only one Variant to load that meets the criteria. Pressing the "Load" button adds the new Variant to the Project, and the "Load" button becomes grayed-out with the "No Variants To Load" status message (Figure 10–31).



Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Alignment Read Type: [✓ Consensus] Individual

Show values: Combined / Forward + reverse / [✓ All three]
[✓ Show denominators]

Filter values
Min = 5.00 %
Max = 100.00 %

Apply min/max % to: Forward or reverse / [✓ Forward and reverse] / Available data
Combined also

Variant status: All

Compact table

No Variants To Load

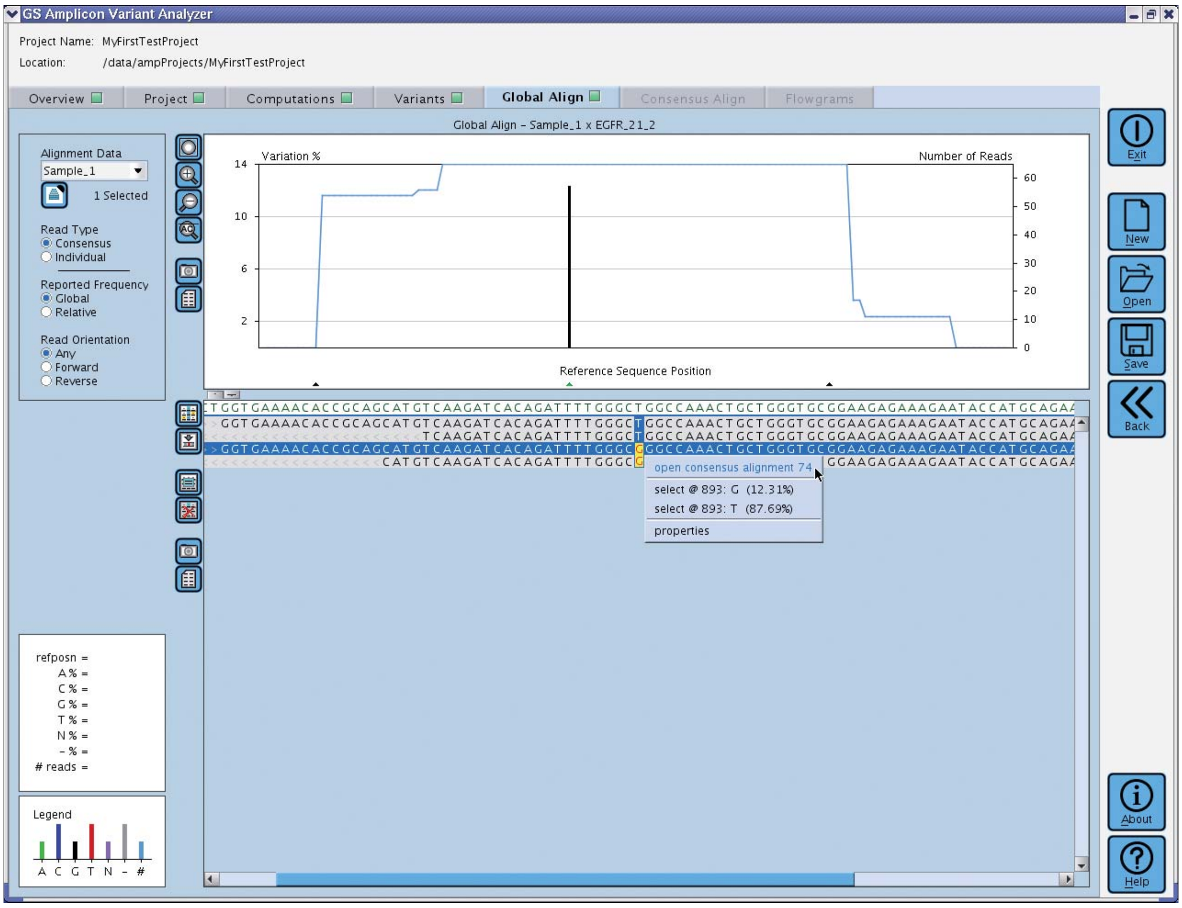
Variants

Reference	Variant	Max		Sample_1	
EGFR_Exons_18-22	893:T/G	12.31		12.31 (65)	
		▸ 11.11	◂ 18.18	▸ 11.11 (54)	◂ 18.18 (11)
EGFR_Exons_18-22	Var_1	8.26		8.26 (5,434)	
		▸ 7.79	◂ 8.64	▸ 7.79 (2,402)	◂ 8.64 (3,032)

combined %
forward %
reverse %
combined of
forward of
reverse of

Figure 10–31: The Variants tab after setting filters and loading the lone surviving Variant

After right-clicking on one of the frequency cells for the new Variant (893:T/G) in the Sample_1 column, we can use the “Global Align” in the menu to load the Global Align tab with the reads covering the Variant position for Sample_1. The global alignment (Figure 10–32) reveals that the Variant is covered by the EGFR_21_2 Amplicon and that there is an imbalance between forward and reverse read representation for this Amplicon. However, the Variant is present in both forward and reverse reads and has a combined frequency of over 12%, so it could well be a legitimate Variant. By right-clicking on the forward consensus containing the Variant, we can navigate to the consensus alignment.

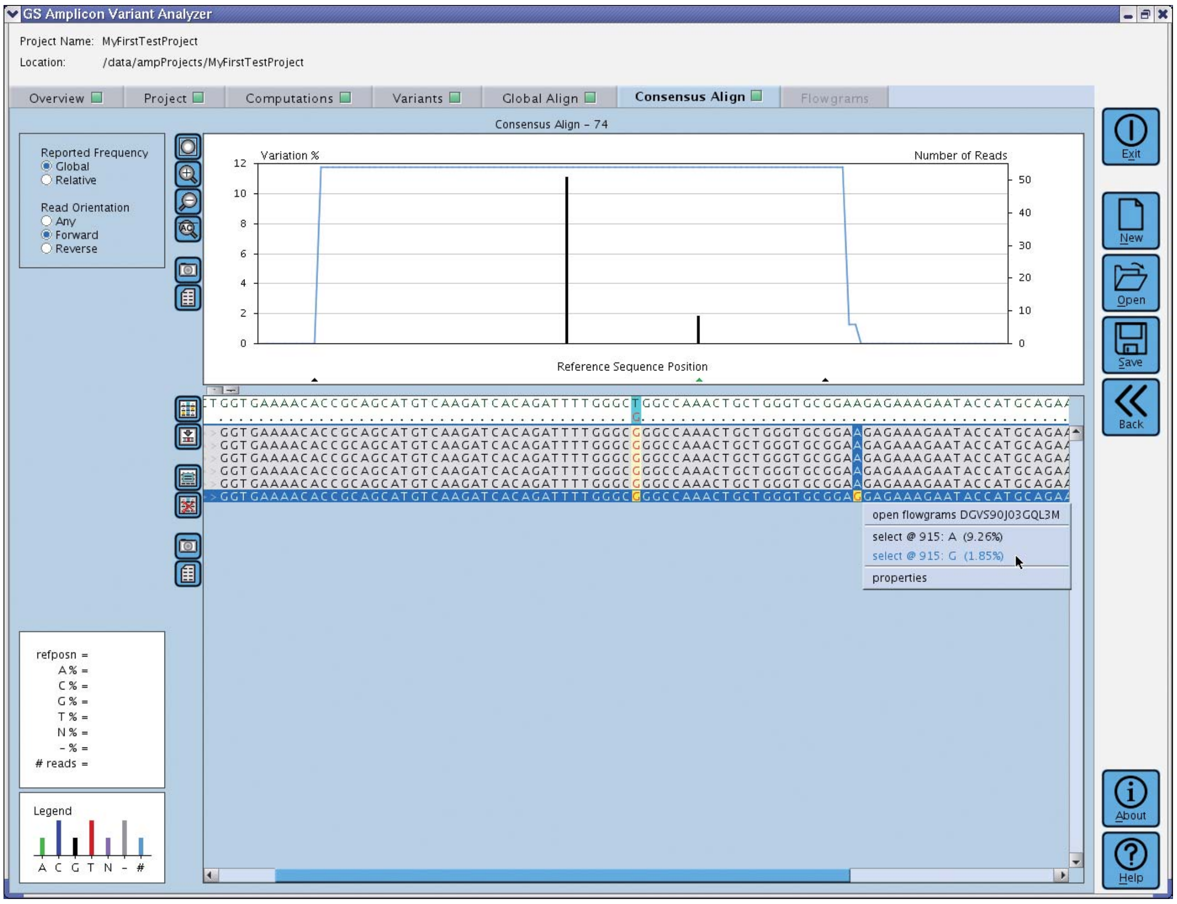


```
TGGTGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCTGGCCAAACTGCTGGGTGCGGAAGAGAAAGAATACCATGCAGAA
>GGTGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCTGGCCAAACTGCTGGGTGCGGAAGAGAAAGAATACCATGCAGAA
<<<<<<<<<<<<<<<<<<<<<TCAAGATCACAGATTTTGGGCTGGCCAAACTGCTGGGTGCGGAAGAGAAAGAATACCATGCAGAA
>GGTGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCGGGCCAAACTGCTGGGTGCGGAAGAGAAAGAATACCATGCAGAA
<<<<<<<<<<<<<<<<<<CATGTCAAGATCACAGATTTTGGGCG                    GGAAGAGAAAGAATACCATGCAGAA
```

Figure 10–32: The Global Align tab displaying the Consensi for Sample_1 covering the region of the “893:T/G” Variant. The right-click context-sensitive menu of the forward consensus is shown in preparation for navigating to the Consensus Align tab.

The Consensus Align tab (see Figure 10–33) shows the 6 (forward) reads that comprise this Consensus, all of which contain the Variant of interest. However, one of those reads has an additional variation (an “A” to “G” substitution at position 915). The automated Variant detection does not scan for haplotypic variations (except for contiguous deletions), so even if this haplotype is real, we would never see it in the Variants Frequency Table unless we introduce the haplotypic variation to the Project manually (although we might encounter the parts of a haplotype in the table, individually).

To define this haplotype, we use the alignment filter selections to narrow down the view to meet the new haplotype: we right-click over the columns of interest in the alignment (893 and 915) and select the Variant base for those columns (“G” for both) (Figure 10–33).



Project Name: MyFirstTestProject
Location: /data/ampProjects/MyFirstTestProject

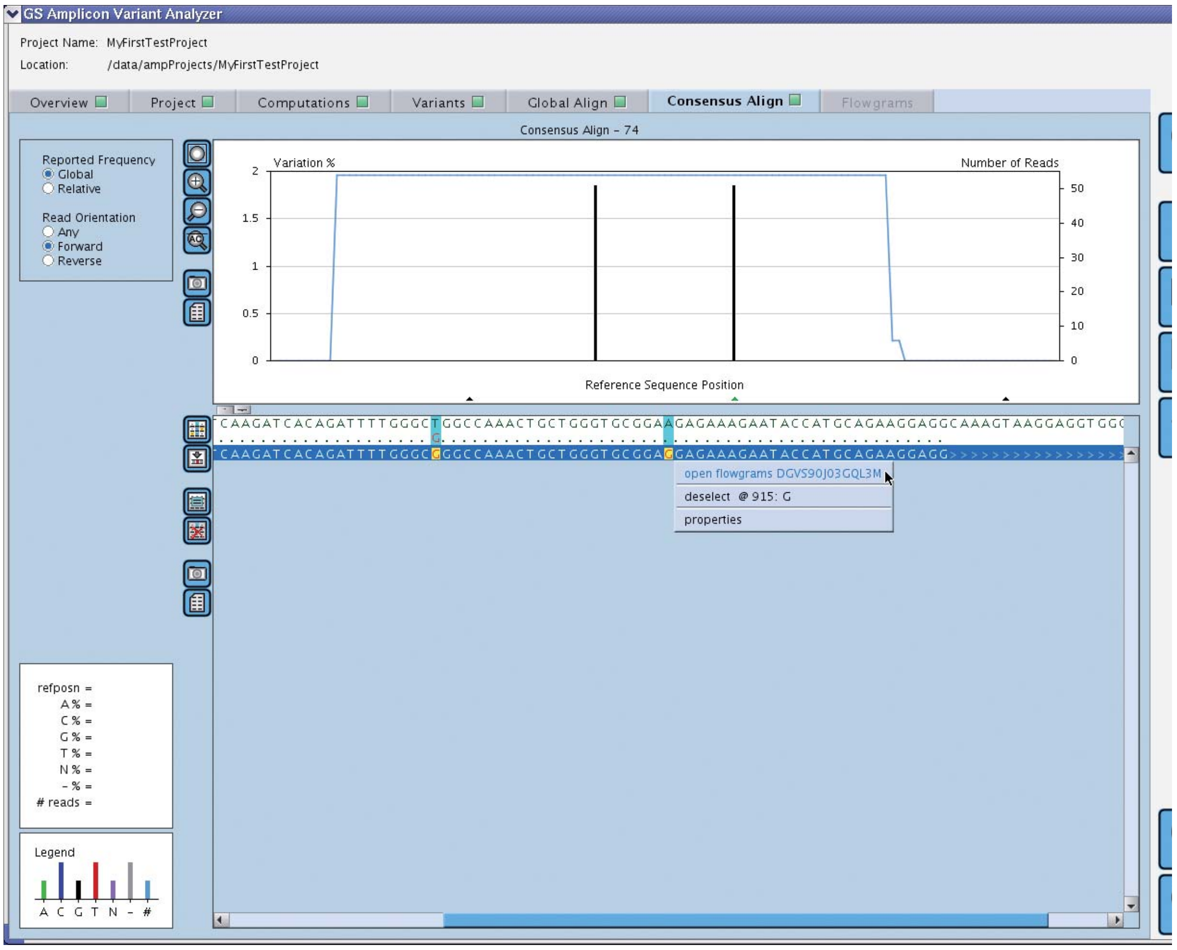
Reported Frequency
[✓ Global]
Relative

Read Orientation
Any
[✓ Forward]
Reverse

```
TGGTGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCTGGCCAAACTGCTGGGTGCGGAAGAGAAAGAATACCATGCAGAA
                                          G
GGTGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCGGGCCAAACTGCTGGGTGCGGAAGAGAAAGAATACCATGCAGAA
GGTGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCGGGCCAAACTGCTGGGTGCGGAAGAGAAAGAATACCATGCAGAA
GGTGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCGGGCCAAACTGCTGGGTGCGGAAGAGAAAGAATACCATGCAGAA
GGTGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCGGGCCAAACTGCTGGGTGCGGAAGAGAAAGAATACCATGCAGAA
GGTGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCGGGCCAAACTGCTGGGTGCGGAAGAGAAAGAATACCATGCAGAA
GGTGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCGGGCCAAACTGCTGGGTGCGGAGGAGAAAGAATACCATGCAGAA
```

Figure 10–33: The Consensus Align tab displaying the forward reads for Sample_1 with the “893:T/G” Variant. The right-click context-sensitive menu is shown in preparation for making a second filter selection on the alignment (a “G” at position 915).

After both selections are made, we have narrowed down the view to a single read. Although a single read isn't very good evidence of a true Variant, we are going through this exercise just to show how a haplotype might be defined. We can right click over this read (Figure 10–34) to load the Flowgrams tab so we can judge whether the variations look real or not.



Project Name: MyFirstTestProject
Location: /data/ampProjects/MyFirstTestProject

Reported Frequency
[✓ Global]
Relative

Read Orientation
Any
[✓ Forward]
Reverse

```
CAAGATCACAGATTTTGGGCTGGCCAAACTGCTGGGTGCGGAAGAGAAAGAATACCATGCAGAAGGAGGCAAAGTAAGGAGGTGGC
....................G.....................................................
CAAGATCACAGATTTTGGGCGGGCCAAACTGCTGGGTGCGGAGGAGAAAGAATACCATGCAGAAGGAGG>>>>>>>>>>>>>>>>>
```

refposn =
A % =
C % =
G % =
T % =
N % =
- % =
reads =

Figure 10–34: The Consensus Align tab displaying the only read with a haplotype including the "893:T/G" Variant and the "915:A/G" Variant. The right-click context-sensitive menu is shown in preparation for navigating to the Flowgrams tab.

The Flowgrams tab view (Figure 10–35) shows that based on the actual flows of the read, the haplotype appears convincing (or would be if it were supported by multiple reads): the original "893:T/G" substitution Variant exhibits a flow cycle shift (the gray column in the middle flowgram for the read). Furthermore, the flow values for the original Variant and the new Variant ("915:A/G") are not marginal: the difference flowgram at the bottom shows the reference bases for both Variants decreasing by a solid value of '1' each, and likewise, the replacement bases are each also increased by '1'. Although we have seen only one instance of this haplotype we will go ahead and set up the haplotype as a Project Variant so we can see if any other instances are to be found (it is conceivable that the haplotype could be hidden in other consensi).

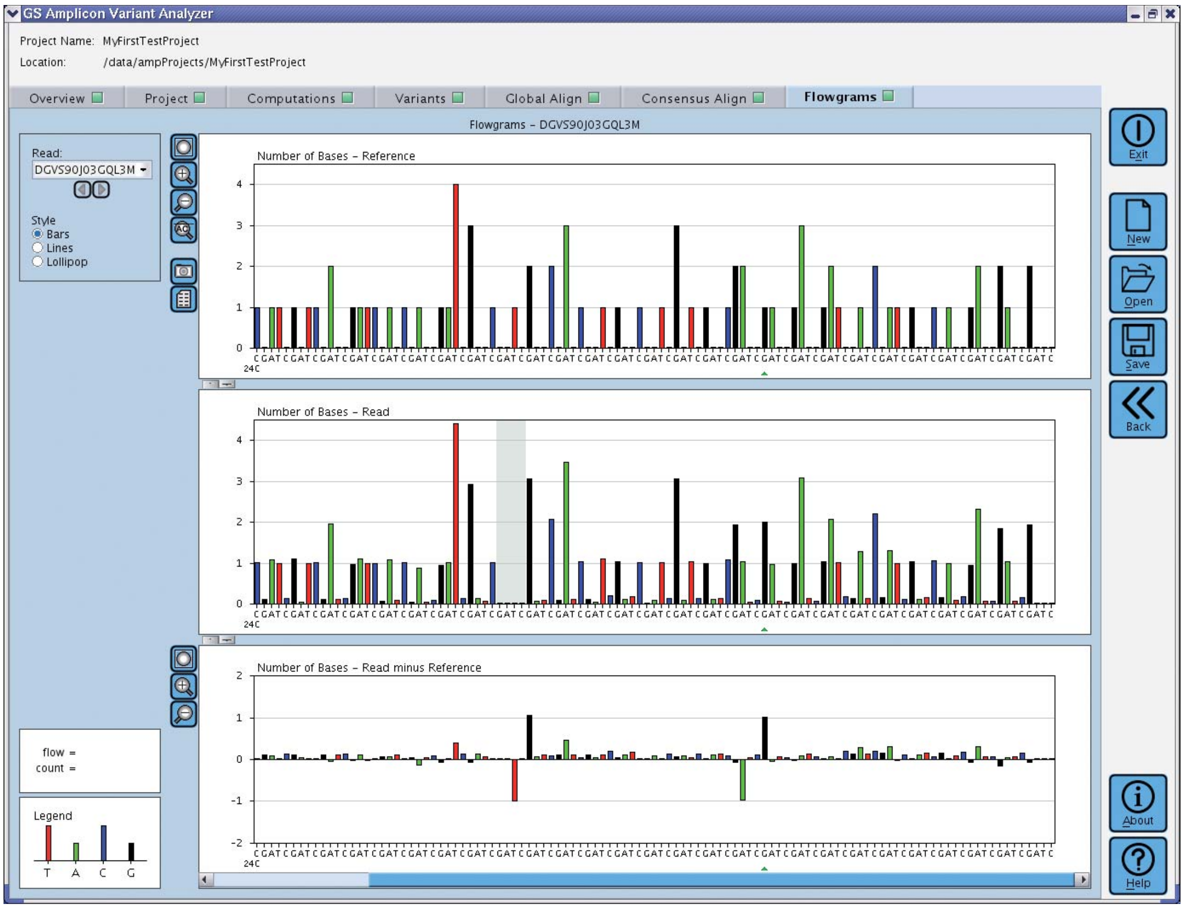
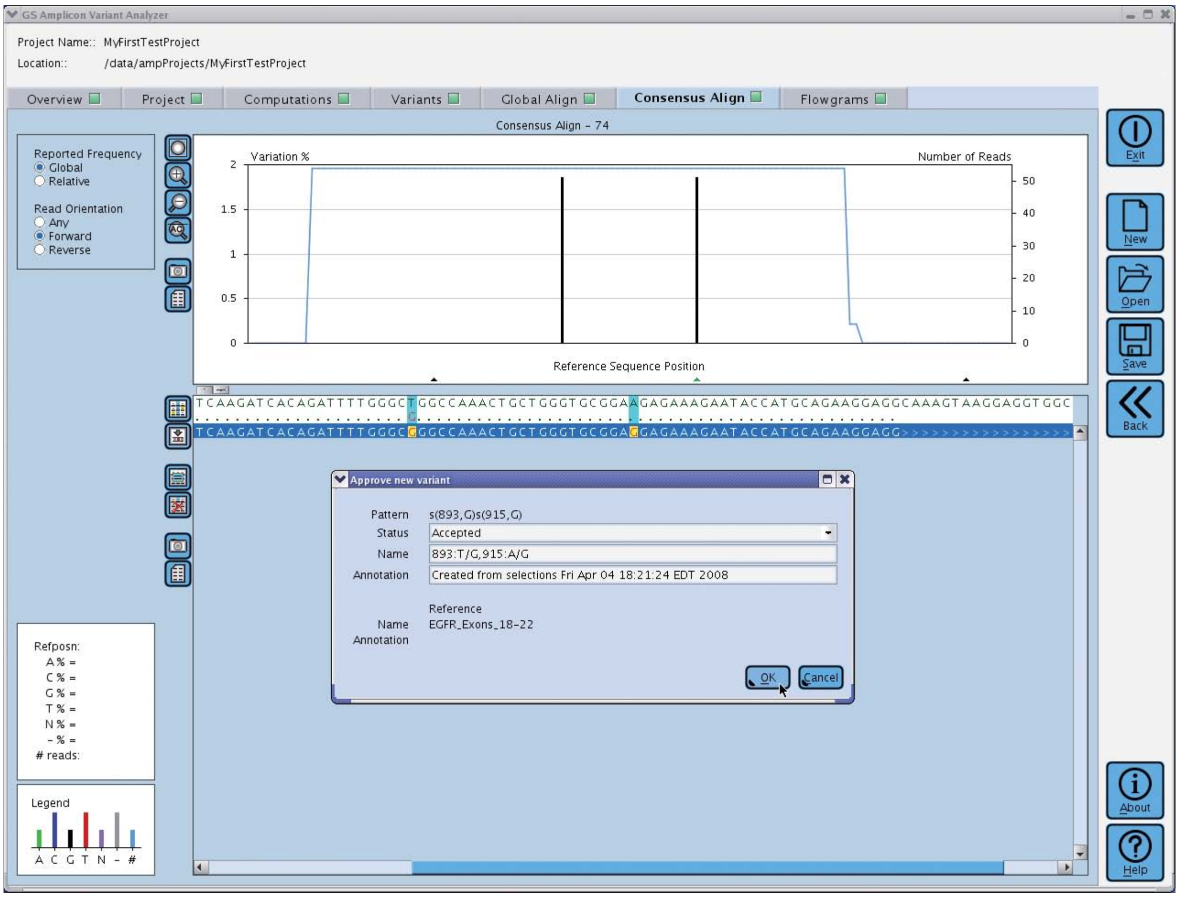

GS Amplicon Variant Analyzer
Project Name: MyFirstTestProject
Location: /data/ampProjects/MyFirstTestProject
Overview
Project
Computations
Variants
Global Align
Consensus Align
Flowgrams
Flowgrams - DGVS90J03GQL3M
Read:
DGVS90J03GQL3M
Style
Bars
Lines
Lollipop
Number of Bases - Reference
Number of Bases - Read
Number of Bases - Read minus Reference
flow =
count =
Legend
T A C G
Exit
New
Open
Save
Back
About
Help

Figure 10–35: The Flowgrams tab for the read displaying the haplotype including the "893:T/G" Variant and the "915:A/G" Variant. The gray shaded column in the middle flowgram denotes a flow cycle shift caused by the "T" to "G" substitution of the first Variant, while the other Variant is indicated by the loss of an "A" and the gain of a "G" in the bottom difference flowgram.

To add our haplotype to the Project as a Variant, we can return to the Consensus Align tab where we have already made the appropriate filter selections (Figure 10–34). We click on the "Declare project variant" button to the left of the alignment, and the "Approve new variant" window opens (Figure 10–36). The automatically created default name for the variant is a sensible concatenation of the two individual Variant names, sorted by position ("893:T/G,915:A/G"), and the rest of the defaults are reasonable. Therefore, we can click "OK" to define the haplotype as a Variant, so subsequent rounds of computation will search for it.



Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Reported Frequency
Global [✓]
Relative

Read Orientation
Any
Forward [✓]
Reverse

```
TCAAGATCACAGATTTTGGGCTGGCCAAACTGCTGGGTGCGGAAGAGAAAGAATACCATGCAGAAGGAGGCAAAGTAAGGAGGTGGC
TCAAGATCACAGATTTTGGGCGGGCCAAACTGCTGGGTGCGGAGGAGAAAGAATACCATGCAGAAGGAGG>>>>>>>>>>>>>>>
```

Refposn:
A % =
C % =
G % =
T % =
N % =
– % =
# reads:

Approve new variant

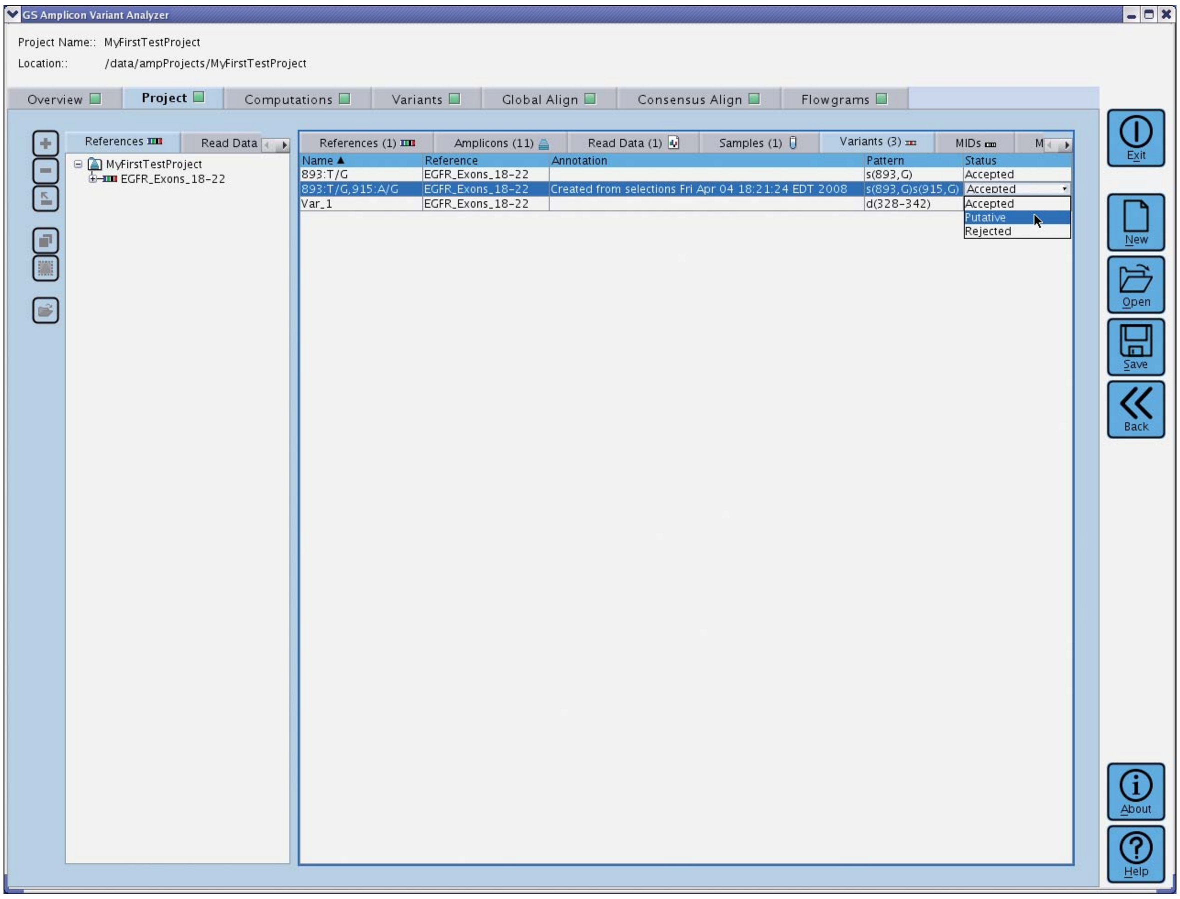
Pattern s(893,G)s(915,G)
Status Accepted
Name 893:T/G,915:A/G
Annotation Created from selections Fri Apr 04 18:21:24 EDT 2008

Reference
Name EGFR_Exons_18-22
Annotation

OK Cancel

Figure 10–36: Creating a Variant from selection filters in the Consensus Align tab

Clicking on the “OK” button to define the haplotype Variant for the Project is a little anti-climactic because the creation takes place behind the scenes and you remain on the same tab we were on when we submitted the Variant. To view the Variant we just created, we can select the Variants sub-tab of the Project tab to see the Variant definition table. That view also allows us to edit the Status of individual Variants in the Project. The Var_1 Variant that we entered manually (section 10.2.6) was automatically marked as “Accepted”. The Auto-Detected Variant that we loaded into the Project (“893:T/G”) defaulted to “Putative”. The haplotype Variant that we manually created from filter selections defaulted to “Accepted”. Now that we have had a chance to look at the data, we can make some reasonable Status changes by double-clicking on the Status field of Variants in the table (Figure 10–37). We should set the Auto-Detected Variant to “Accepted” rather than “Putative” since we saw ample evidence that it is real. The haplotype, however, is very questionable because it is supported by a single read, so we will demote it to “Putative”. Alternatively, we could have initially assigned a “Putative” Status to the haplotype, by changing the default “Accepted” status in the “Status” drop down menu (as seen in Figure 10–36) to “Putative”, prior to clicking the “OK” button in the “Approve new variant” popup.



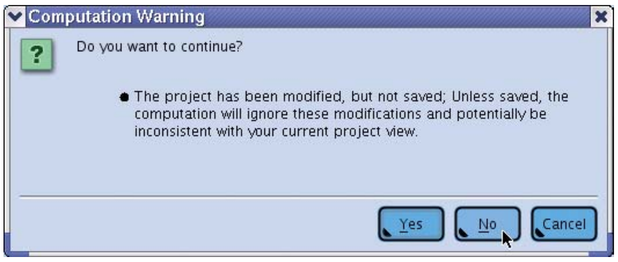
Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Name ▲	Reference	Annotation	Pattern	Status
893:T/G	EGFR_Exons_18-22		s(893,G)	Accepted
893:T/G,915:A/G	EGFR_Exons_18-22	Created from selections Fri Apr 04 18:21:24 EDT 2008	s(893,G)s(915,G)	Accepted
Var_1	EGFR_Exons_18-22		d(328-342)	Accepted

Accepted
Putative
Rejected

Figure 10–37: Changing the Status of Variants in the Variants sub-tab of the Project tab

With our new Variants defined, we are ready to compute the Project again to get frequencies for the new Variants. If we rush off to the Computation tab and press the start button, we will get a warning (Figure 10–38): we forgot to save the Project first. After hitting “No” or “Cancel” and pressing the “Save” button for the Project, we should be able to start the computation successfully.

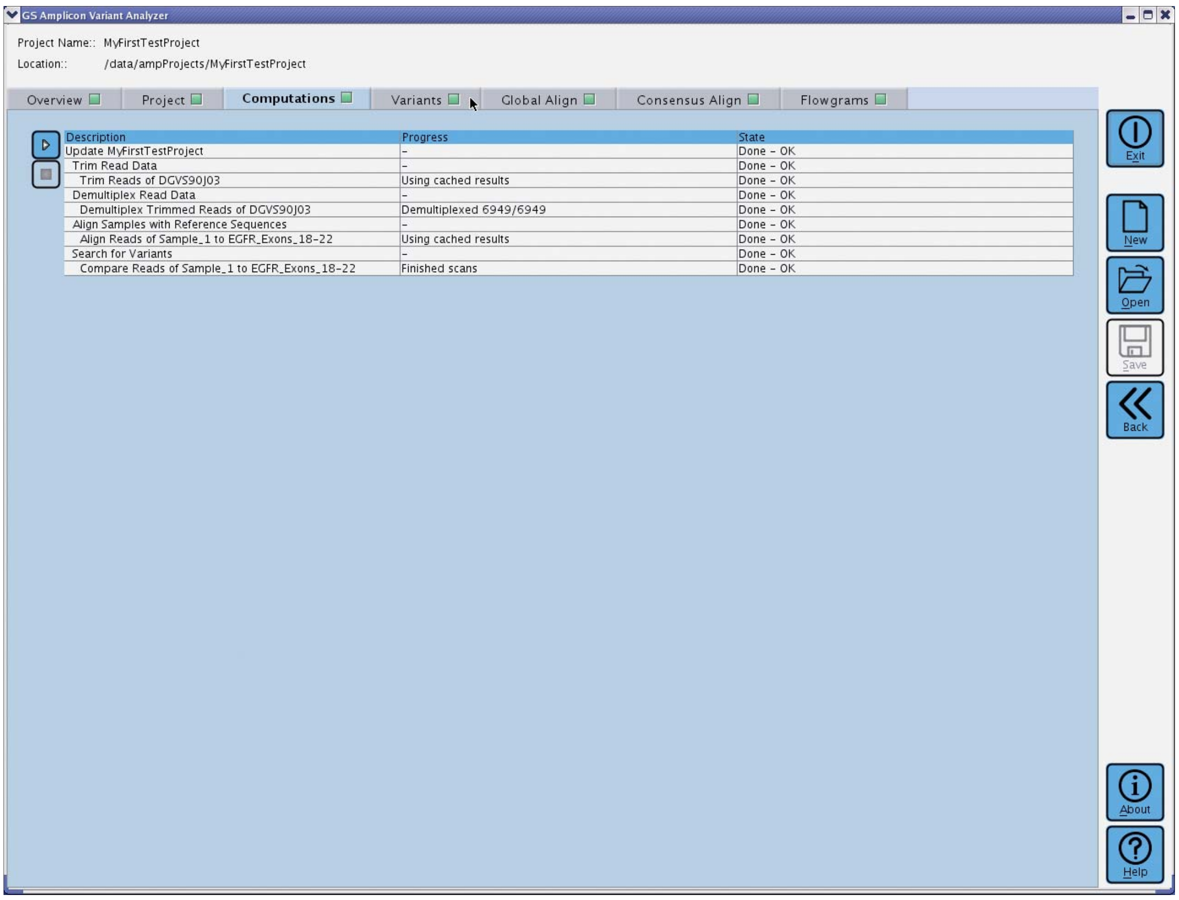


Do you want to continue?

- The project has been modified, but not saved; Unless saved, the computation will ignore these modifications and potentially be inconsistent with your current project view.

Figure 10–38: Warning message alerting the user that a re-computation is being set up but that some details of the Project have not been saved to disk

The computation should finish very quickly (Figure 10–39). Note that the computation made use of cached results from the previous computation (section 10.3.1). Except for the demultiplexing step, which is rerun with every computation, the only novel work the computation had to do was the “Search for Variants” step.

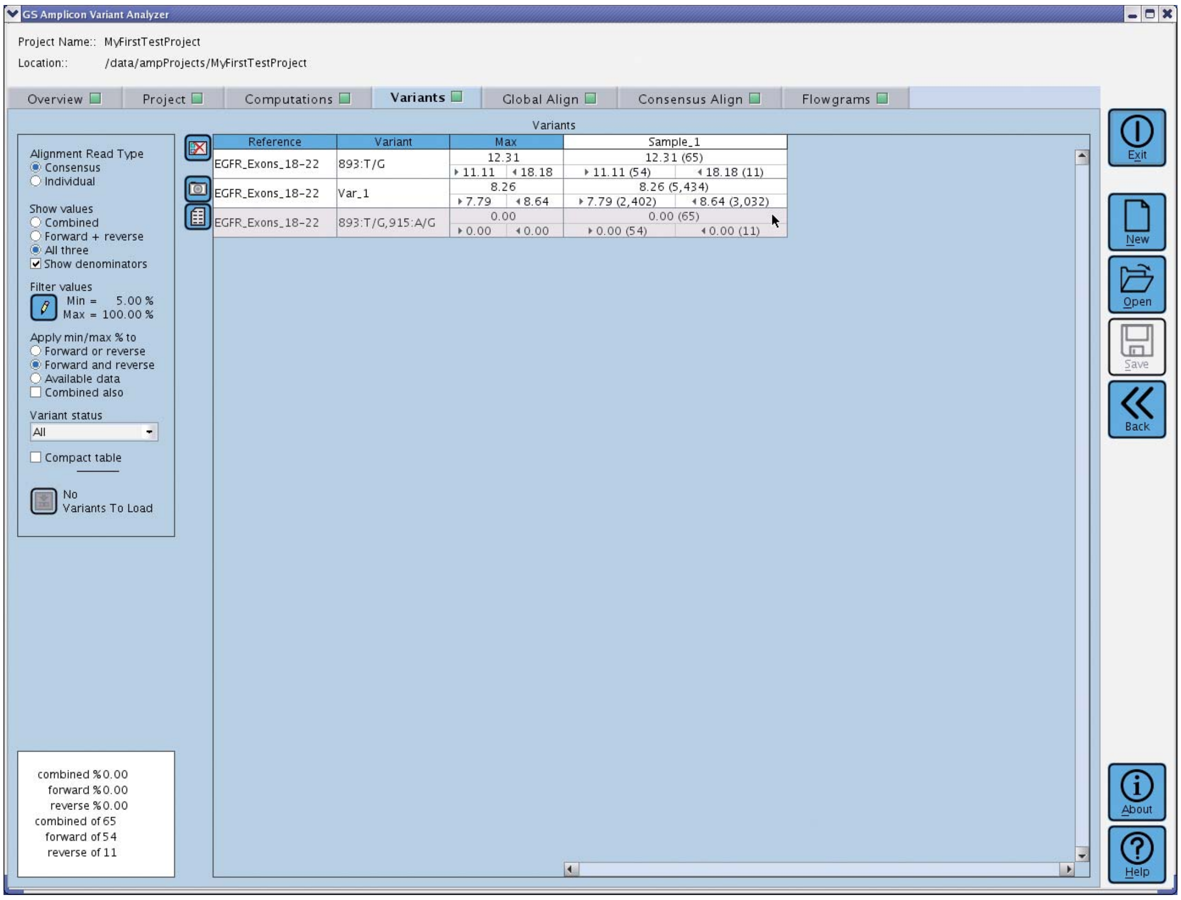


Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Description	Progress	State
Update MyFirstTestProject	–	Done – OK
Trim Read Data	–	Done – OK
Trim Reads of DGVS90J03	Using cached results	Done – OK
Demultiplex Read Data	–	Done – OK
Demultiplex Trimmed Reads of DGVS90J03	Demultiplexed 6949/6949	Done – OK
Align Samples with Reference Sequences	–	Done – OK
Align Reads of Sample_1 to EGFR_Exons_18-22	Using cached results	Done – OK
Search for Variants	–	Done – OK
Compare Reads of Sample_1 to EGFR_Exons_18-22	Finished scans	Done – OK

Figure 10–39: The Computations tab showing the results of a second round of computation on the Project, including the use of cached results but a new Variant search

After the computation is complete, we can click on the main Variants tab to see the frequencies of our Variants in our Sample. The haplotype Variant we defined appears to not have been detected at all in the initial view of the Variants Frequency Table (Figure 10–40; frequency of 0.00% with a total of 65 reads, and grayed out row); this is because the haplotype Variant was defined from an individual read that was buried inside a consensus sequence, but the "Alignment Read Type" filter happens to be set to "Consensus" in the current view.



GS Amplicon Variant Analyzer

Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Variants

Alignment Read Type
[✓ Consensus]
Individual

Show values
Combined
Forward + reverse
[✓ All three]
[✓ Show denominators]

Filter values
Min = 5.00 %
Max = 100.00 %

Apply min/max % to
Forward or reverse
[✓ Forward and reverse]
Available data
Combined also

Variant status
All

Compact table

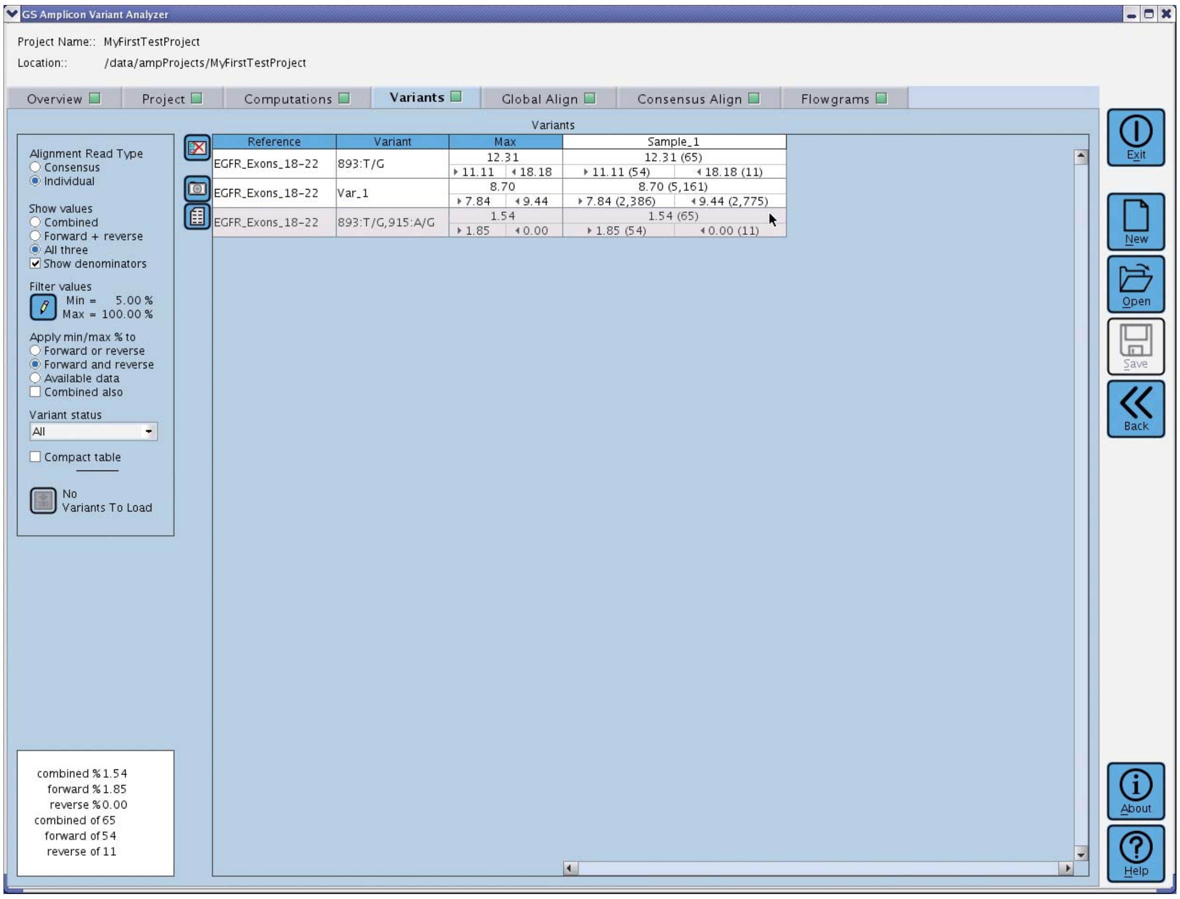
No
Variants To Load

Reference	Variant	Max		Sample_1	
EGFR_Exons_18-22	893:T/G	12.31		12.31 (65)	
		▸11.11	◂18.18	▸11.11 (54)	◂18.18 (11)
EGFR_Exons_18-22	Var_1	8.26		8.26 (5,434)	
		▸7.79	◂8.64	▸7.79 (2,402)	◂8.64 (3,032)
EGFR_Exons_18-22	893:T/G,915:A/G	0.00		0.00 (65)	
		▸0.00	◂0.00	▸0.00 (54)	◂0.00 (11)

```
combined %0.00
 forward %0.00
 reverse %0.00
combined of 65
 forward of 54
 reverse of 11
```

Figure 10–40: The Variants tab showing the results of a second round of computation on the Project. The haplotype Variant was not detected in this view because the "Alignment Read Type" is set to "Consensus".

If we toggle the “Alignment Read Type” to “Individual”, we can see that the haplotype Variant was not missing entirely (Figure 10–41). The frequency of 1.54% out of 65 reads for this Variant reveals that only one read was found with the haplotype (the very one we used to define it). Without further supporting evidence, this haplotype Variant should probably not be considered legitimate despite the fact that the flowgram evidence was good: it is most likely a read that had a PCR error at position 915.



Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Alignment Read Type
( ) Consensus
(●) Individual

Show values
( ) Combined
( ) Forward + reverse
(●) All three
[✓] Show denominators

Filter values
Min = 5.00 %
Max = 100.00 %

Apply min/max % to
( ) Forward or reverse
(●) Forward and reverse
( ) Available data
[ ] Combined also

Variant status
All

[ ] Compact table

No
Variants To Load

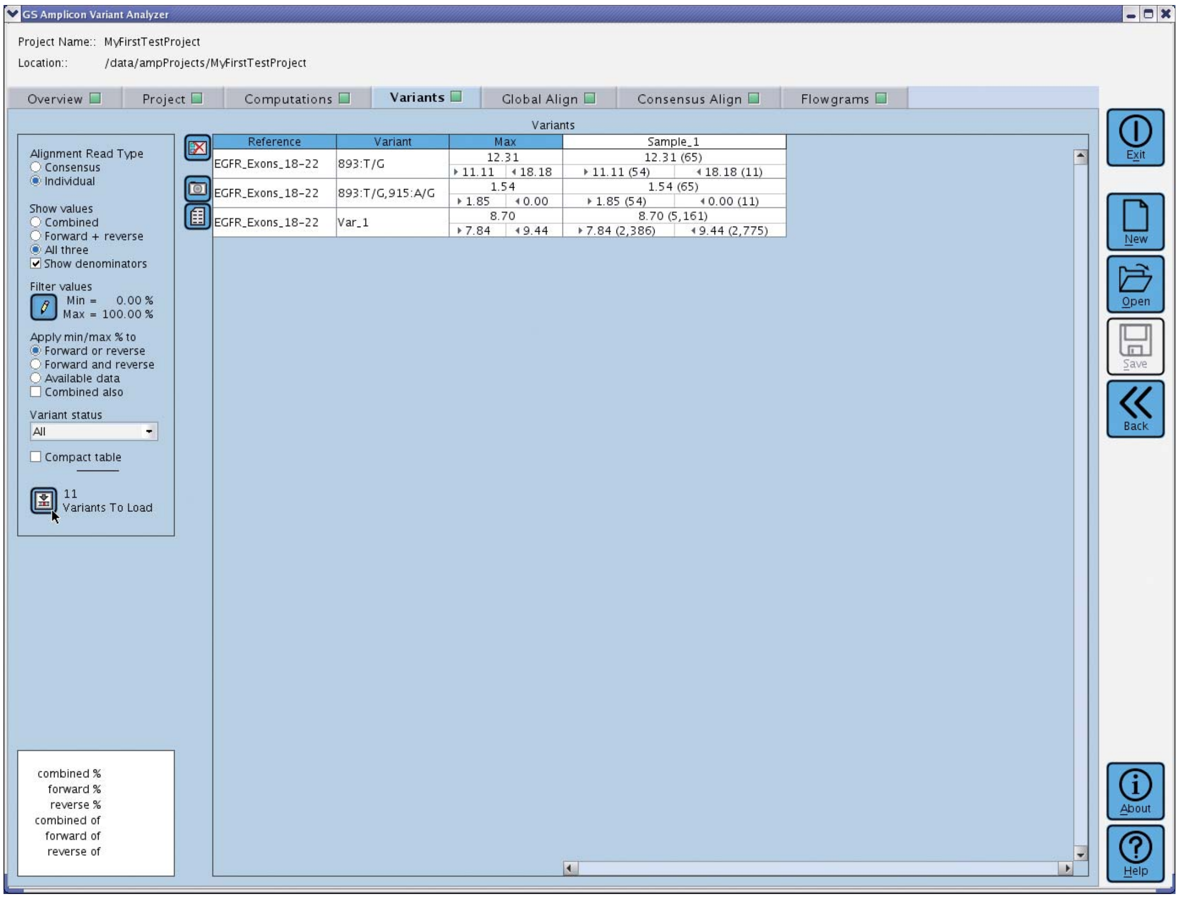
Variants

Reference	Variant	Max		Sample_1	
EGFR_Exons_18-22	893:T/G	12.31		12.31 (65)	
		▸11.11	◂18.18	▸11.11 (54)	◂18.18 (11)
EGFR_Exons_18-22	Var_1	8.70		8.70 (5,161)	
		▸7.84	◂9.44	▸7.84 (2,386)	◂9.44 (2,775)
EGFR_Exons_18-22	893:T/G,915:A/G	1.54		1.54 (65)	
		▸1.85	◂0.00	▸1.85 (54)	◂0.00 (11)

combined % 1.54
forward % 1.85
reverse % 0.00
combined of 65
forward of 54
reverse of 11

Figure 10–41: The Variants tab with “Alignment Read Type” toggled to “Individual”, showing that the haplotype Variant was detected after all, but only at 1.54% of 65 reads (a single read)

The first time we loaded Auto-Detected Variants, we used fairly stringent filters, to start with the most likely candidate Variants (there turned out to be a single Variant that met these criteria: the "893:T/G" Variant). However, there might be some Variants in the data that are real but in suboptimal contexts. Perhaps the Variant is in a position of the Amplicon that is only covered by reads from one direction or maybe the Variant is present at a frequency lower than the 5% cut-off we used for our first filter. We will now reset the filters to their most permissive values to allow all the remaining Auto-Detected Variants to be loaded into the Project. We do this by resetting the Min to 0.00% and selecting the "Forward or reverse" option. Under those selections, the "Load" button reveals that there are 11 variants to load (Figure 10–42). This also causes all the rows in the table to be displayed with a white background (the haplotype Variant row was previously grayed-out because of its low frequency).


GS Amplicon Variant Analyzer

Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Alignment Read Type
- ○ Consensus
- ● Individual

Show values
- ○ Combined
- ○ Forward + reverse
- ● All three
- ☑ Show denominators

Filter values
Min = 0.00 %
Max = 100.00 %

Apply min/max % to
- ● Forward or reverse
- ○ Forward and reverse
- ○ Available data
- ☐ Combined also

Variant status: All

☐ Compact table

11
Variants To Load

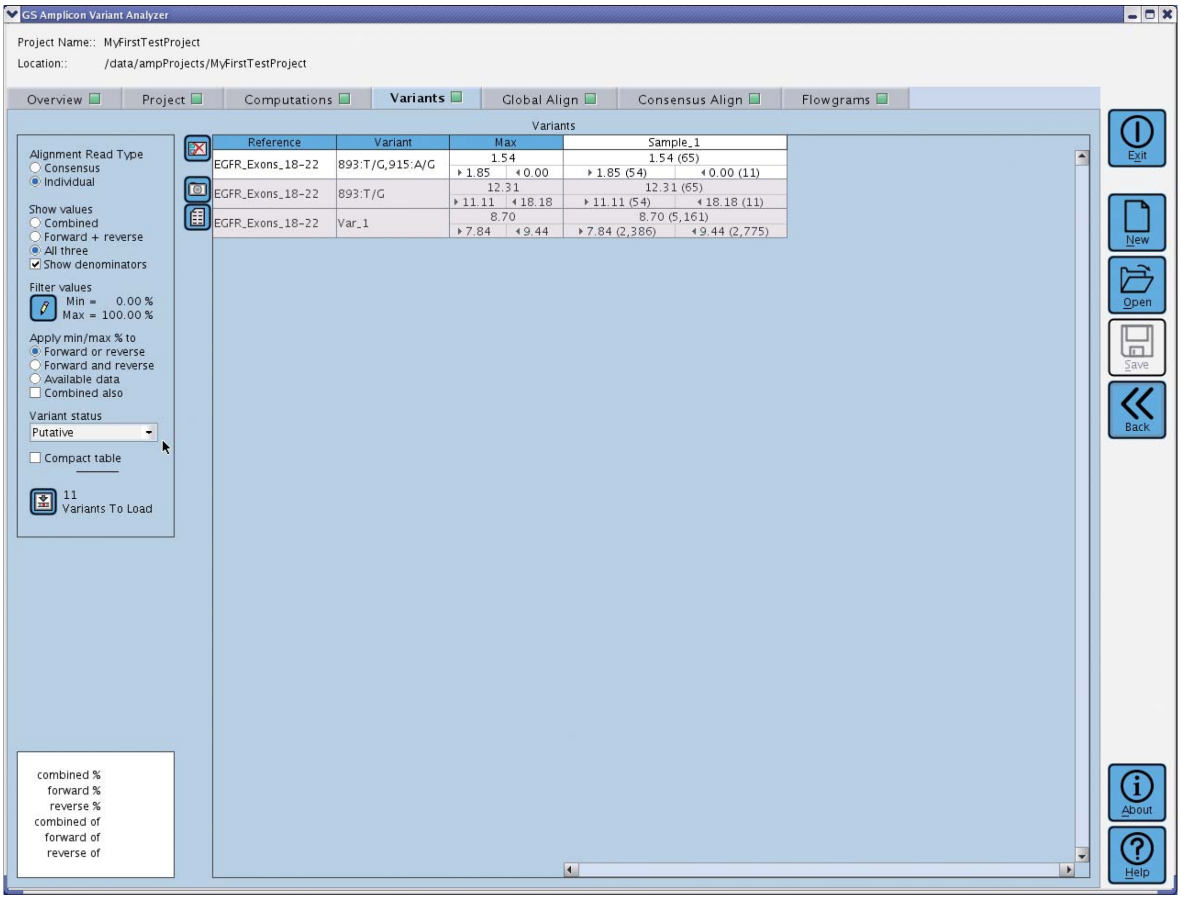
Variants

Reference	Variant	Max		Sample_1	
EGFR_Exons_18-22	893:T/G	12.31		12.31 (65)	
		▸11.11	◂18.18	▸11.11 (54)	◂18.18 (11)
EGFR_Exons_18-22	893:T/G,915:A/G	1.54		1.54 (65)	
		▸1.85	◂0.00	▸1.85 (54)	◂0.00 (11)
EGFR_Exons_18-22	Var_1	8.70		8.70 (5,161)	
		▸7.84	◂9.44	▸7.84 (2,386)	◂9.44 (2,775)

combined %
forward %
reverse %
combined of
forward of
reverse of

Figure 10–42: The Variants tab with filters relaxed to allow loading of the rest of the Auto-Detected Variants

Prior to loading the Auto-Detected Variants, we can use the "Variant status" filter to prepare the Variants Frequency Table to assist us with workflow for examining the Variants. This filter influences the display by graying-out the rows of Variants in the table that do not meet the filter selection. If we set the filter to "Putative", the rows for the two "Accepted" Variants that are already in the Project are grayed-out (Figure 10–43), but the haplotype Variant remains white because its Status was previously marked as "Putative". Note that this filter has no influence on the "Load" button. Even though the "Load" button imports the Auto-Detected Variants as "Putative", setting this filter to "Accepted" would not change the number of Variants (11, in this case) that can be loaded; this number is based solely on the other filter settings.



GS Amplicon Variant Analyzer

Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Alignment Read Type: Consensus / [✓ Individual]

Show values: Combined / Forward + reverse / [✓ All three]

[✓] Show denominators

Filter values: Min = 0.00 % Max = 100.00 %

Apply min/max % to: [✓ Forward or reverse] / Forward and reverse / Available data

[ ] Combined also

Variant status: Putative

[ ] Compact table

11 Variants To Load

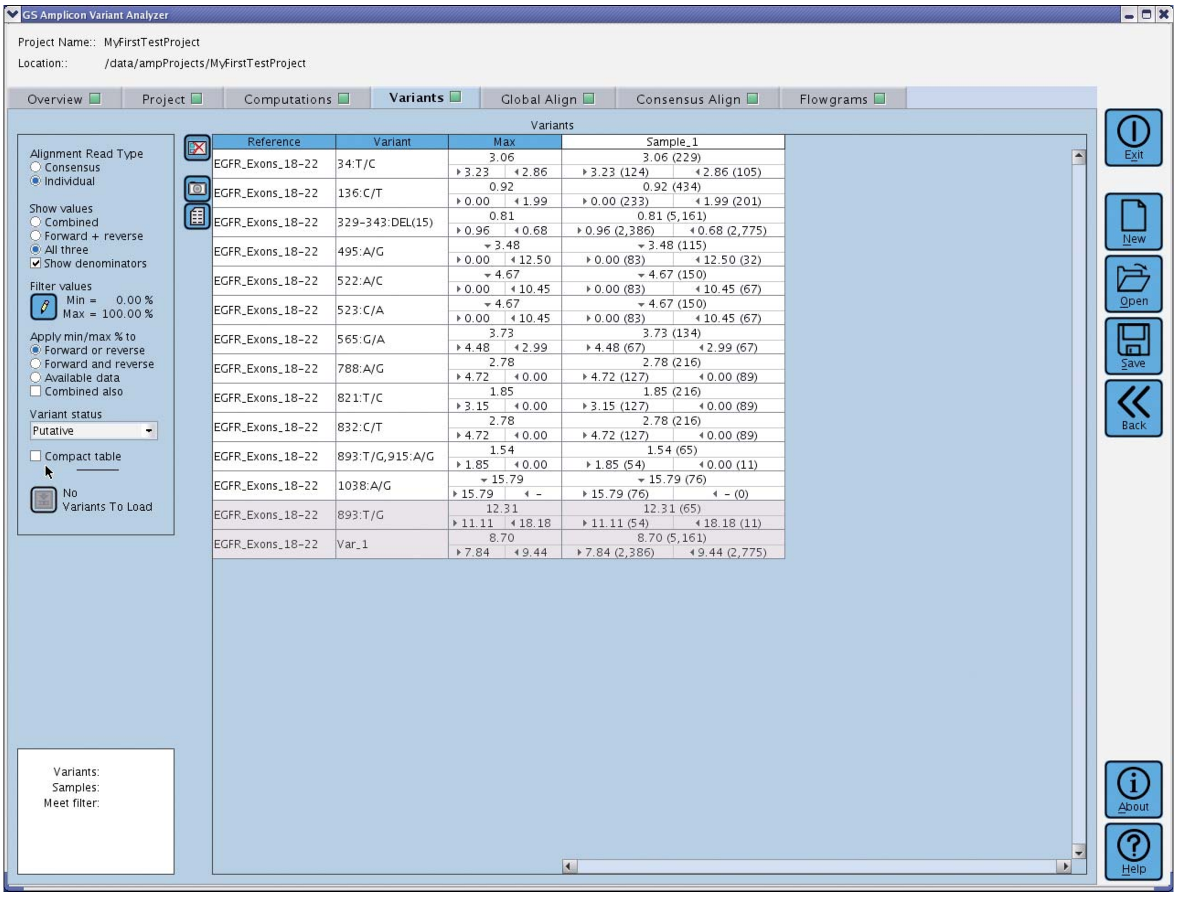
Variants

Reference	Variant	Max		Sample_1	
EGFR_Exons_18-22	893:T/G,915:A/G	1.54		1.54 (65)	
		▸1.85	◂0.00	▸1.85 (54)	◂0.00 (11)
EGFR_Exons_18-22	893:T/G	12.31		12.31 (65)	
		▸11.11	◂18.18	▸11.11 (54)	◂18.18 (11)
EGFR_Exons_18-22	Var_1	8.70		8.70 (5,161)	
		▸7.84	◂9.44	▸7.84 (2,386)	◂9.44 (2,775)

combined %
forward %
reverse %
combined of
forward of
reverse of

Figure 10–43: The Variants tab with the "Variant status" filter set to "Putative". This causes the two "Accepted" variant rows to be grayed-out

Eleven Variants is a manageable number that we can reasonably load and examine at one time, so we click the "Load" button to import them. Once we do, the new Variants are all visible as white rows in the Variant Frequencies Table because the "Variant Status" filter is set to "Putative", the default for Auto-Detected Variants (Figure 10–44). The frequencies for these Variants are automatically filled out and are valid as of the completion of the last computation: we don't need to run another round of computation to update the frequencies until we make changes to the Project that would affect the calculation of the frequencies (such as new Samples or Read Data, or any change in the Reference Sequence). The situation is different for manually defined Variants, which require a round of computation after their definition in order to appear in the Table.



Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Alignment Read Type
○ Consensus
● Individual

Show values
○ Combined
○ Forward + reverse
● All three
✓ Show denominators

Filter values
Min = 0.00 %
Max = 100.00 %

Apply min/max % to
● Forward or reverse
○ Forward and reverse
○ Available data
☐ Combined also

Variant status
Putative

☐ Compact table

No
Variants To Load

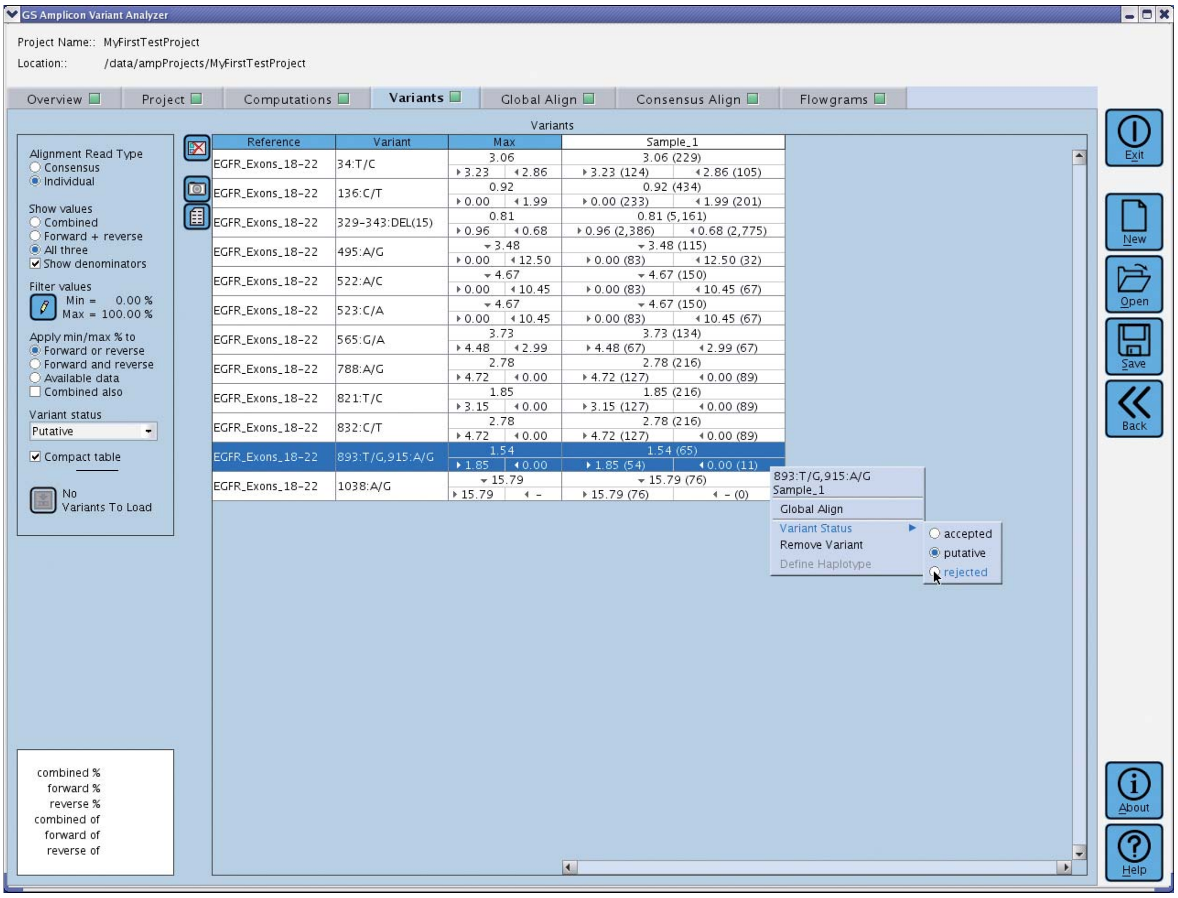
Variants:
Samples:
Meet filter:

Variants

Reference	Variant	Max	Sample_1
EGFR_Exons_18-22	34:T/C	3.06 ▸3.23 ◂2.86	3.06 (229) ▸3.23 (124) ◂2.86 (105)
EGFR_Exons_18-22	136:C/T	0.92 ▸0.00 ◂1.99	0.92 (434) ▸0.00 (233) ◂1.99 (201)
EGFR_Exons_18-22	329-343:DEL(15)	0.81 ▸0.96 ◂0.68	0.81 (5,161) ▸0.96 (2,386) ◂0.68 (2,775)
EGFR_Exons_18-22	495:A/G	▾3.48 ▸0.00 ◂12.50	▾3.48 (115) ▸0.00 (83) ◂12.50 (32)
EGFR_Exons_18-22	522:A/C	▾4.67 ▸0.00 ◂10.45	▾4.67 (150) ▸0.00 (83) ◂10.45 (67)
EGFR_Exons_18-22	523:C/A	▾4.67 ▸0.00 ◂10.45	▾4.67 (150) ▸0.00 (83) ◂10.45 (67)
EGFR_Exons_18-22	565:G/A	3.73 ▸4.48 ◂2.99	3.73 (134) ▸4.48 (67) ◂2.99 (67)
EGFR_Exons_18-22	788:A/G	2.78 ▸4.72 ◂0.00	2.78 (216) ▸4.72 (127) ◂0.00 (89)
EGFR_Exons_18-22	821:T/C	1.85 ▸3.15 ◂0.00	1.85 (216) ▸3.15 (127) ◂0.00 (89)
EGFR_Exons_18-22	832:C/T	2.78 ▸4.72 ◂0.00	2.78 (216) ▸4.72 (127) ◂0.00 (89)
EGFR_Exons_18-22	893:T/G,915:A/G	1.54 ▸1.85 ◂0.00	1.54 (65) ▸1.85 (54) ◂0.00 (11)
EGFR_Exons_18-22	1038:A/G	▾15.79 ▸15.79 ◂ -	▾15.79 (76) ▸15.79 (76) ◂ - (0)
EGFR_Exons_18-22	893:T/G	12.31 ▸11.11 ◂18.18	12.31 (65) ▸11.11 (54) ◂18.18 (11)
EGFR_Exons_18-22	Var_1	8.70 ▸7.84 ◂9.44	8.70 (5,161) ▸7.84 (2,386) ◂9.44 (2,775)

Figure 10–44: The Variants tab after the Auto-Detected Variants are loaded

The next phase of workflow control for the newly added Variants is to select the "Compact table" option while we leave the "Variant status" filter set to "Putative". This hides any Variant rows where the Status is either "Accepted" or "Rejected". In this case the immediate effect is to hide the rows of the two Variants that we have already validated and set to "Accepted" (Figure 10–45). Under this configuration of the Variants Frequency Table, we can right-click any Sample-Variant frequency cell to expose the "Global Align" navigation link, as we did before for the first Auto-Detected Variant we loaded. After investigating the "Putative" Variants visible in the table and editing their status to either "Accepted" or "Rejected", they will drop out of view. In this case, we have already decided that the haplotype Variant probably isn't real, so we can go ahead and mark it as "Rejected". The Status of a Variant can be changed via a sub-menu available when you right-click on a Variant cell in the Variants Frequency Table (this is shown for the haplotype Variant in Figure 10–45), or by editing the Status field of the Variant in the definition table, in the Variants sub-tab of the Project tab.



Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Alignment Read Type
Consensus [✓ Individual]

Show values
Combined / Forward + reverse [✓ All three]
[✓ Show denominators]

Filter values
Min = 0.00 %
Max = 100.00 %

Apply min/max % to
[✓ Forward or reverse] / Forward and reverse / Available data
Combined also

Variant status: Putative

[✓ Compact table]

No Variants To Load

Variants

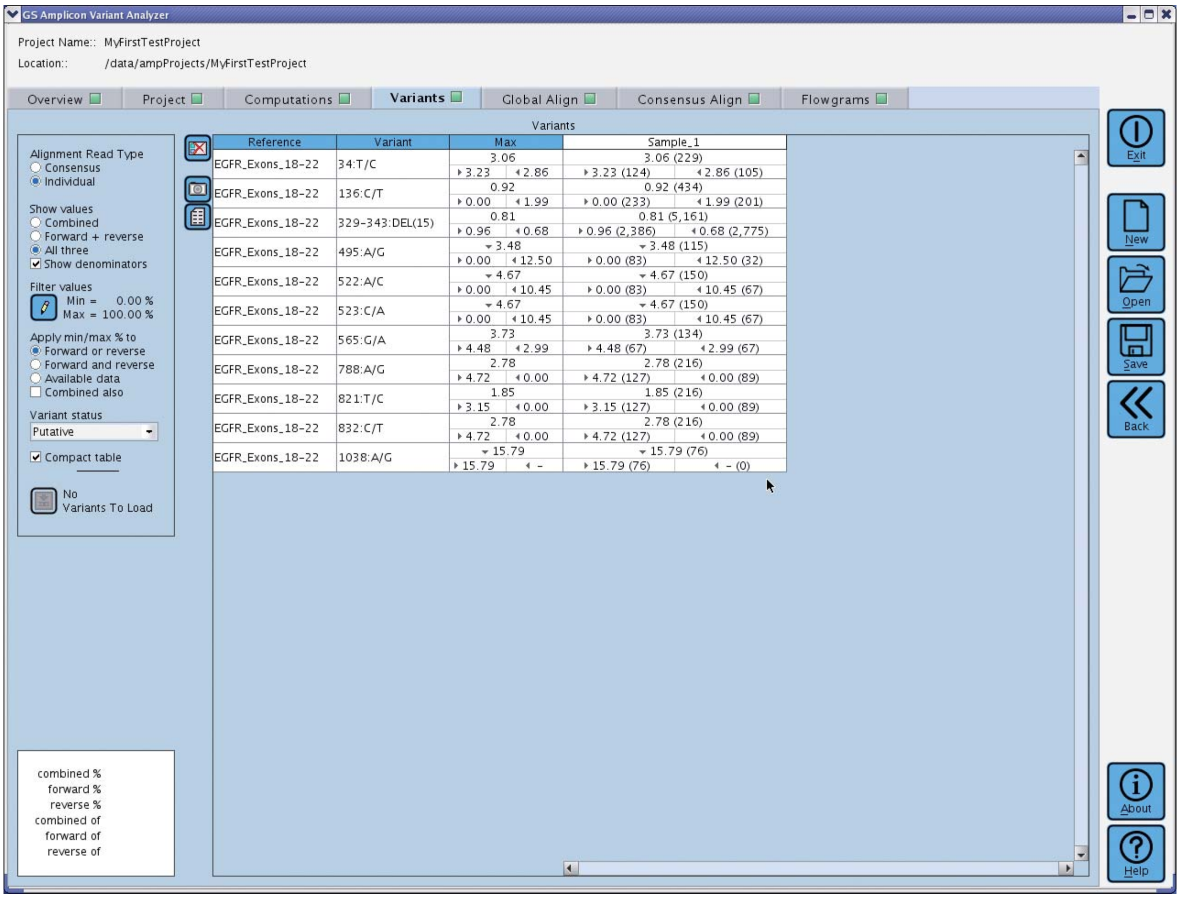
Reference	Variant	Max			Sample_1		
EGFR_Exons_18-22	34:T/C	3.06	▸3.23	◂2.86	3.06 (229)	▸3.23 (124)	◂2.86 (105)
EGFR_Exons_18-22	136:C/T	0.92	▸0.00	◂1.99	0.92 (434)	▸0.00 (233)	◂1.99 (201)
EGFR_Exons_18-22	329-343:DEL(15)	0.81	▸0.96	◂0.68	0.81 (5,161)	▸0.96 (2,386)	◂0.68 (2,775)
EGFR_Exons_18-22	495:A/G	▾3.48	▸0.00	◂12.50	▾3.48 (115)	▸0.00 (83)	◂12.50 (32)
EGFR_Exons_18-22	522:A/C	▾4.67	▸0.00	◂10.45	▾4.67 (150)	▸0.00 (83)	◂10.45 (67)
EGFR_Exons_18-22	523:C/A	▾4.67	▸0.00	◂10.45	▾4.67 (150)	▸0.00 (83)	◂10.45 (67)
EGFR_Exons_18-22	565:G/A	3.73	▸4.48	◂2.99	3.73 (134)	▸4.48 (67)	◂2.99 (67)
EGFR_Exons_18-22	788:A/G	2.78	▸4.72	◂0.00	2.78 (216)	▸4.72 (127)	◂0.00 (89)
EGFR_Exons_18-22	821:T/C	1.85	▸3.15	◂0.00	1.85 (216)	▸3.15 (127)	◂0.00 (89)
EGFR_Exons_18-22	832:C/T	2.78	▸4.72	◂0.00	2.78 (216)	▸4.72 (127)	◂0.00 (89)
EGFR_Exons_18-22	893:T/G,915:A/G	1.54	▸1.85	◂0.00	1.54 (65)	▸1.85 (54)	◂0.00 (11)
EGFR_Exons_18-22	1038:A/G	▾15.79	▸15.79	◂ -	▾15.79 (76)	▸15.79 (76)	◂ - (0)

Variant Status: accepted [✓ putative] rejected

combined %
forward %
reverse %
combined of
forward of
reverse of

Figure 10–45: The Variants tab after "Compact table" has been selected. This has hidden the two "Accepted" Variants that were previously grayed-out because of the "Putative" setting of the "Variant status" filter. The expanded right-click menus are poised to mark the haplotype Variant as "rejected".

After marking the haplotype Variant as “Rejected” it immediately disappears from view (Figure 10–46). Note that marking a Variant as “Rejected” rather than deleting it outright from the Project can be useful because this keeps the system from subsequently re-proposing it and forcing you to validate it more than once. Similarly, if we investigate one of the Auto-Detected Variants and determine that it is valid, we can change its Status to “Accepted” and it will also drop from view. In this way we can continue to work through Variants until all have been evaluated and the table is empty. At that point we could set the “Variant Status” filter to “Accepted” to display only the Variants in which we are confident, generating a convenient report table that we can export.



Project Name:: MyFirstTestProject
Location:: /data/ampProjects/MyFirstTestProject

Alignment Read Type
- Consensus
- [✓ Individual]

Show values
- Combined
- Forward + reverse
- [✓ All three]
- [✓ Show denominators]

Filter values
Min = 0.00 %
Max = 100.00 %

Apply min/max % to
- [✓ Forward or reverse]
- Forward and reverse
- Available data
- Combined also

Variant status
Putative

[✓ Compact table]

No
Variants To Load

Variants

Reference	Variant	Max		Sample_1	
EGFR_Exons_18–22	34:T/C	3.06		3.06 (229)	
		▸3.23	◂2.86	▸3.23 (124)	◂2.86 (105)
EGFR_Exons_18–22	136:C/T	0.92		0.92 (434)	
		▸0.00	◂1.99	▸0.00 (233)	◂1.99 (201)
EGFR_Exons_18–22	329–343:DEL(15)	0.81		0.81 (5,161)	
		▸0.96	◂0.68	▸0.96 (2,386)	◂0.68 (2,775)
EGFR_Exons_18–22	495:A/G	▾3.48		▾3.48 (115)	
		▸0.00	◂12.50	▸0.00 (83)	◂12.50 (32)
EGFR_Exons_18–22	522:A/C	▾4.67		▾4.67 (150)	
		▸0.00	◂10.45	▸0.00 (83)	◂10.45 (67)
EGFR_Exons_18–22	523:C/A	▾4.67		▾4.67 (150)	
		▸0.00	◂10.45	▸0.00 (83)	◂10.45 (67)
EGFR_Exons_18–22	565:G/A	3.73		3.73 (134)	
		▸4.48	◂2.99	▸4.48 (67)	◂2.99 (67)
EGFR_Exons_18–22	788:A/G	2.78		2.78 (216)	
		▸4.72	◂0.00	▸4.72 (127)	◂0.00 (89)
EGFR_Exons_18–22	821:T/C	1.85		1.85 (216)	
		▸3.15	◂0.00	▸3.15 (127)	◂0.00 (89)
EGFR_Exons_18–22	832:C/T	2.78		2.78 (216)	
		▸4.72	◂0.00	▸4.72 (127)	◂0.00 (89)
EGFR_Exons_18–22	1038:A/G	▾15.79		▾15.79 (76)	
		▸15.79	◂ –	▸15.79 (76)	◂ – (0)

combined %
forward %
reverse %
combined of
forward of
reverse of

Figure 10–46: The Variant tab after the haplotype Variant has been rejected. The haplotype is immediately hidden because the “Variant status” filter is set to “Putative”, while the “Compact table” option is active.

10.5 Important Factors in the Assessment of New Variants

The examples above clearly show that variations observed in the reads of a sequencing experiment should be given careful scrutiny before they can be considered to be true Variants, existing physically in the DNA sample that was sequenced. This section enumerates and briefly describes some of the main data features to examine when making this kind of assessment.

10.5.1 Above the Noise

One major factor is the noise level in the Variation Frequency Plot. If you observe a lot of low-level frequency variation in the plot, a potential Variant would have to be convincingly above that noise level to be believed.

10.5.2 Coverage

If the depth of coverage at the potential Variant position is very low, a low frequency variant becomes less believable. At higher coverage, low frequency events become more believable provided they are convincingly above the noise. In general, one would want sufficient coverage to see several concrete instances of the variation. However, at extremely high coverage, you should be aware that identical noise type events can occur more than once, so seeing a small number of variant instances in such a case would not necessarily provide convincing evidence.

10.5.3 Bidirectional Support

If your experiment was designed so that the Target has been sequenced from both directions, you can use that information to probe the validity of a potential Variant. This is only useful if the position of your Variant is in a region of the alignment that is covered by both forward and reverse reads; if the alignment position is only covered by reads of one direction you shouldn't penalize the validity of the Variant for lack of bidirectional evidence.

If the specific Variant in question can be found in both forward and reverse reads, it is more believable as a true Variant. If the frequency of the Variant is similar in both directions, it is even more believable. If the frequency of the Variant is drastically different between the two directions, or if the Variant can only be found in one direction, you should be less likely to believe the Variant.

10.5.4 Homopolymers

If your potential Variant results from the overcall or undercall of a homopolymer, the length of the homopolymer and the sequence context will affect your assessment of the Variant. As homopolymer length increases, it becomes more likely that an erroneous overcall or undercall will occur. If the homopolymer affected by the Variant occurs close to another homopolymer of the same nucleotide, known sequencing artifacts called carry-forward and incomplete extension could have caused the undercall or overcall.

10.5.5 Flowgram Evidence

If you filter the alignment to show only those reads containing your variant of interest, you can dig down into the flowgrams of the individual reads to see how convincing the Variant is.

The flowgram lowest on the page represents the subtraction of the reference flowgram from the read flowgram and shows which bases have been overcalled or undercalled in the read according to the reference. Does your potential Variant cause an appropriate shift in the heights of flowgram bars across the series of flowgrams? Is the intensity value of the shift always on the high end or low end of the expected value, or does it more appropriately form a narrow distribution around the middle of the expected value?

If your variant is an insertion or deletion that affects a single flowgram bar, you only have the intensity of the bar to guide you. If your variant is a substitution, it will simultaneously affect the heights of at least two bars, making the Variant more believable. The most convincing flowgram evidence will be if your Variant happens to cause a nucleotide flow cycle shift in the flowgram. This will be detectable as some inserted flows in the reference flowgram at the top plot or in the read flowgram in the middle plot, which are highlighted in grey.

10.5.6 Read Length

Sequencing quality can begin to drop off at the trailing edge of a read. You should closely examine potential Variants that are at the edges of reads. On the alignment tabs, the orientation of the reads with respect to the reference is indicated, so the end of a forward read is to the right, while the end of a reverse read is to the left. If your Variant is at the end of a forward or reverse read, you should examine other types of corroborating factors like bidirectional support and flowgram evidence. If your candidate Variant only has support in a single direction, you should look at multiple reads in the same direction that share the Variant of interest. If the reads have multiple additional errors in close proximity to the Variant, it is likely an indication that the Variant isn't real but is the result of read quality drop off.

10.6 Other Issues of Special Interest

When you are familiar with the basics of the 'Amplicon Variant Analysis' software and you are ready to setup projects with your own data, you should take some time to consider the optimum setup for your project based on the specifics of your experimental design and the way in which you intend to analyze the results. Primarily, you need to decide what the term 'Sample' means to you; you need to decide what type of project organization you need; and you need to decide on the relationships between your Amplicons and Reference Sequences.

10.6.1 What Does 'Sample' Mean?

One of the major decisions to make is what the term 'Sample' means to you within the context of your Project. The software recognizes a 'Sample' as a generic grouping unit of data. It is at its essence, merely a label with some optional annotation. It is up to you to decide how to best group your experimental data into Samples.

A typical Project might use a 'Sample' to represent DNA from a distinct source, such as a tube of genomic DNA from a particular subject. Another Project might have a more specific definition of Sample, and split out different classes of DNA from a single individual and call them separate Samples, such as control and experimental Samples or pre-treatment and post-treatment research Samples. Yet another project might define Samples as distinct replicates of a DNA source to allow for statistical comparison between them.

You are free to get more granular with Sample definitions (such as assigning each amplicon for an individual to its own Sample), but you should keep in mind that you can only view a single Sample-Reference Sequence alignment/difference plot at a time. You can examine cross-Sample Variant frequency statistics in the main 'Variant' tab, but reads from different Samples cannot be viewed in the same alignment. You can, however, navigate from Sample to Sample for a particular Reference Sequence.

There is a limit on the granularity of your Sample definitions. The fundamental unit of computation is the individual Read Data Set. If you intend to divide an individual Read Data Set into multiple Samples, it must be feasible for the software to assign the individual reads from that Set to Samples. If MIDs are not used, this Sample assignment will be performed based solely on the primer content of the Amplicons being measured for the Samples. If MIDs are used, then a read's primer content will first be used to determine the Amplicon it represents, and then that Amplicon's association with a Multiplexer, within the read's Read Data Set of origin, will be used to assign the read to the appropriate Sample. It is important to remember that for a given Read Data Set, each Amplicon may be associated with at most one Sample, unless MIDs are used. With MIDs, a Multiplexer can associate an Amplicon with multiple Samples within a Read Data Set, but each Amplicon may still be associated with at most one Multiplexer.

10.6.2 How Should Your Project Be Organized?

Once you have decided on your preferred definition of 'Sample', you need to decide how your Samples and Amplicons should be organized within one or more Projects. Projects should be used to group together analyses for either ease of comparison or ease of navigation.

Much of the effort in setting up a Project has to do with defining Reference Sequences and Amplicons. Therefore, it is advisable to set up your Projects so that they contain Amplicons that will be measured across a variety of Samples. That way you can continue to add new Samples to the existing Project rather than starting a new Project for each batch of Samples.

If you are scanning for a large number of different Amplicons from a single data source, collecting them within the same Project would also make sense because it would make navigation easier. For example, while reviewing your results, you could jump from Amplicon to Amplicon without having to open different Projects.

You are free to organize your Sample-Amplicon analyses into one or more Projects as you find convenient. If you are a low volume user, you may be tempted to keep all of your analyses in the same Project even if they are unrelated to each other. However, you should keep in mind that pooling too many unrelated Samples together in a Project may unnecessarily clutter navigation menus and Variant summary pages.

10.6.3 Should Amplicons Share a Reference Sequence or Have Individual Ones?

When you are setting up your Amplicons for a Project, you will need to consider two opposing issues. The first issue is that smaller Reference Sequences are more efficient for computation. Excessively large Reference Sequences can lead to long computation times and slow scrolling and navigation, so shorter ones are preferable on that count. On the other hand, alignment views are restricted by Sample and Reference Sequence combinations. This means that if you want to look at alignments or difference plots for two or more different Amplicons at the same time, those Amplicons must be defined from within the same Reference Sequence.

It makes sense to use a common Reference Sequence when your Amplicons actually overlap with one another and to use separate ones for Amplicons that don't overlap. However, you can also construct artificial Reference Sequences that allow you to view multiple unrelated Amplicons in a tab at the same time. These artificial Reference Sequences can be constructed by concatenating Amplicon sequences together with a string of N's as separators. Such a Reference Sequence would be convenient if you have a small to moderate set of Amplicons that you are measuring in Samples that contain an unknown number of Variants. You would then be able to look at the difference plot and get an overview of all the Amplicons at the same time, so you can identify obvious variations.

However, if you use an artificial Reference Sequence with too many Amplicons in it, you will get diminishing returns; the longer Reference Sequence will slow down computation, and the alignments will get more inconvenient to navigate. In general it is best to keep your Reference Sequences as compact as possible, thus if you wanted to measure a large number of exons from a particular gene, it would be better to use a Reference Sequence constructed by concatenating the exons (joining them with N-separators), rather than using the full genomic sequence of the gene. As long as the exons don't overlap with each other, it would be even better to use separate Reference Sequences for each exon (provided viewing the exons within the same alignment or difference plot is not a priority).

10.6.4 When should MIDs be used?

The GS Amplicon Variant Analyzer (AVA) software provides a number of mechanisms for demultiplexing reads, allowing multiple Amplicons from the same or different Samples to be sequenced simultaneously within a PTP region. The simplest demultiplexing method, which has been available since the first release of the AVA software, exploits the template-specific primer regions of the Adaptors, used to prepare the library, to identify the Amplicons. When an Amplicon library is prepared following the procedures given in the *GS FLX Amplicon DNA Library Preparation Method Manual*, these sequences appear at the beginning of the reads, just after the sequencing key (which is part of Primers A and B). If an experiment calls for measuring multiple distinct Amplicons from the same Sample, those Amplicons may be mixed together in a PTP region, and the Project can be set up such that reads of the various Amplicons are associated with the appropriate Sample by virtue of their known template-specific primer sequences.

But with the large number of sequencing reads that can be obtained in a single PicoTiterPlate device region in the Genome Sequencer System, the situation may be common whereby a single region would produce a vast excess of reads compared to what is necessary for any given Amplicon library (Sample). If the experiment includes multiple Samples, the obvious economical solution would be to load multiple Samples in each region such that each Sample will be covered at the appropriate depth, in a single sequencing Run. If different Amplicons were to be sequenced in each of the Samples, the standard demultiplexing method using the template-specific Primer 1 and Primer 2 sequences would be sufficient to assign each read to the proper Sample.

However, experiments where the same Amplicon, or set of Amplicons, are to be sequenced in several Samples are probably much more common. In such cases, one would face the restriction that an Amplicon can be associated with no more than one Sample, within a Read Data Set (equivalent to a PicoTiterPlate device region, unless the data has been manipulated using the SFF Tools).

MIDs are short, recognizable sequence tags that can be added to the design of the Adaptors used for library preparation. Multiple Amplicon libraries (the Project's Samples) can be prepared that include the same Amplicon target sequences (with the same template-specific primers), each labeled with different MID tags. The MID sequences provide extra context that is specific to each library and that, in concert with the template-specific primers, allows flexible demultiplexing options and, specifically enables the sequencing of the same Amplicon across multiple Samples within the same PTP region.

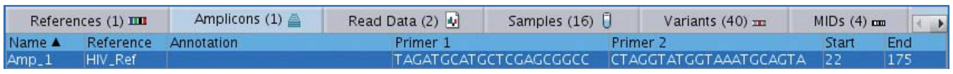
10.6.5 What is the purpose of Multiplexers?

Multiplexers are used as a means to help the user avoid unnecessary duplication of effort when entering Project setup information and to help make the computation of Project results more efficient by allowing a consolidation of processing steps. To illustrate this, this section describes an example case whereby the Project is specified with and without the use of Multiplexers.

As the starting point, assume an experiment involving 16 different Samples where the same gene (single Amplicon) is being measured in each Sample, but each Sample has a specific pairing of MIDs at each end so that they can be multiplexed together for sequencing. The MID sequences being used are Mid1, Mid2, Mid3, and Mid4 and a 'both' encoding is being specified so those 4 sequences can be used combinatorially in pairs to indicate all 16 Samples. All the Samples are multiplexed into a pool and sequenced in two regions of the PicoTiterPlate device (resulting in two Read Data Sets).

10.6.5.1 Non-Multiplexer Example

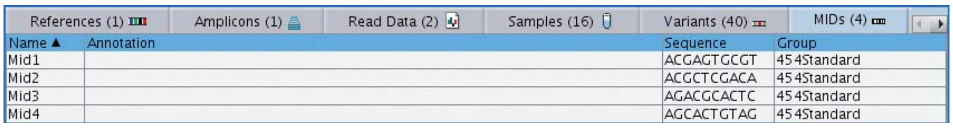
First, let's define the Amplicon to be measured in the 16 different Samples (Figure 10–47). Since the Samples will be pooled together on the same Read Data Set, MIDs are necessary.



Name ▲	Reference	Annotation	Primer 1	Primer 2	Start	End
Amp_1	HIV_Ref		TAGATGCATGCTCGAGCGGCC	CTAGGTATGGTAAATGCAGTA	22	175

Figure 10–47: An Amplicon to be measured in 16 different Samples

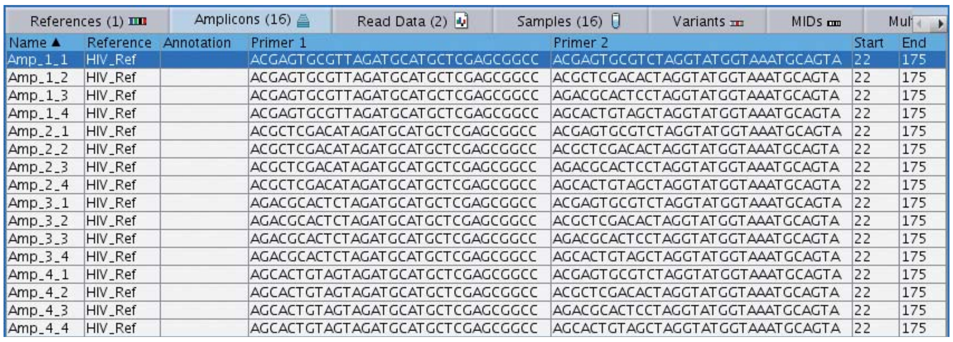
Using MIDs on both sides of the Amplicon, and assuming the final product is short enough to be sequenced in full within a read, only 4 MID sequences (Figure 10–48) are needed (combinatorially, using the "Both" encoding) to distinguish the Amplicons from all 16 Samples.



Name ▲	Annotation	Sequence	Group
Mid1		ACGAGTGCGT	454Standard
Mid2		ACGCTCGACA	454Standard
Mid3		AGACGCACTC	454Standard
Mid4		AGCACTGTAG	454Standard

Figure 10–48: The sequences of 4 MIDs being used to identify 16 Samples by employing a "Both" encoding

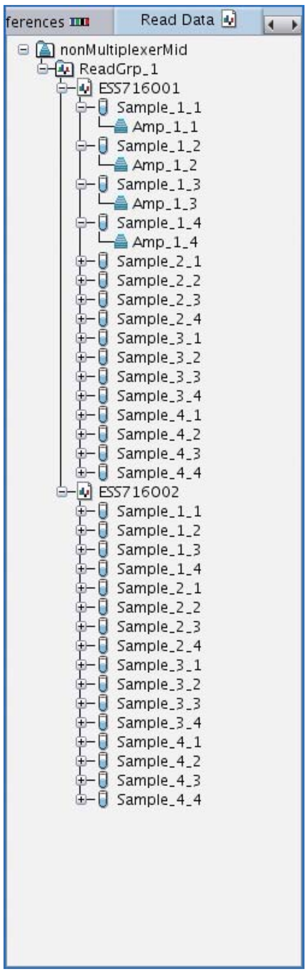
If Multiplexers were unavailable it would be necessary to define 16 different Amplicons. This is because in actuality there are 16 different Amplicon library products involved where the MID sequences would need to be considered as part of the template-specific portion of the Amplicon primers. Figure 10–49 shows the 16 different Amplicons that would need to be created for the experiment. The Amplicons are named according to their MID layout; "Amp_4_3" has Mid4 upstream of Primer1 and Mid3 upstream of Primer 2.



Name ▲	Reference	Annotation	Primer 1	Primer 2	Start	End
Amp_1_1	HIV_Ref		ACGAGTGCGTTAGATGCATGCTCGAGCGGCC	ACGAGTGCGTCTAGGTATGGTAAATGCAGTA	22	175
Amp_1_2	HIV_Ref		ACGAGTGCGTTAGATGCATGCTCGAGCGGCC	ACGCTCGACACTAGGTATGGTAAATGCAGTA	22	175
Amp_1_3	HIV_Ref		ACGAGTGCGTTAGATGCATGCTCGAGCGGCC	AGACGCACTCCTAGGTATGGTAAATGCAGTA	22	175
Amp_1_4	HIV_Ref		ACGAGTGCGTTAGATGCATGCTCGAGCGGCC	AGCACTGTAGCTAGGTATGGTAAATGCAGTA	22	175
Amp_2_1	HIV_Ref		ACGCTCGACATAGATGCATGCTCGAGCGGCC	ACGAGTGCGTCTAGGTATGGTAAATGCAGTA	22	175
Amp_2_2	HIV_Ref		ACGCTCGACATAGATGCATGCTCGAGCGGCC	ACGCTCGACACTAGGTATGGTAAATGCAGTA	22	175
Amp_2_3	HIV_Ref		ACGCTCGACATAGATGCATGCTCGAGCGGCC	AGACGCACTCCTAGGTATGGTAAATGCAGTA	22	175
Amp_2_4	HIV_Ref		ACGCTCGACATAGATGCATGCTCGAGCGGCC	AGCACTGTAGCTAGGTATGGTAAATGCAGTA	22	175
Amp_3_1	HIV_Ref		AGACGCACTCTAGATGCATGCTCGAGCGGCC	ACGAGTGCGTCTAGGTATGGTAAATGCAGTA	22	175
Amp_3_2	HIV_Ref		AGACGCACTCTAGATGCATGCTCGAGCGGCC	ACGCTCGACACTAGGTATGGTAAATGCAGTA	22	175
Amp_3_3	HIV_Ref		AGACGCACTCTAGATGCATGCTCGAGCGGCC	AGACGCACTCCTAGGTATGGTAAATGCAGTA	22	175
Amp_3_4	HIV_Ref		AGACGCACTCTAGATGCATGCTCGAGCGGCC	AGCACTGTAGCTAGGTATGGTAAATGCAGTA	22	175
Amp_4_1	HIV_Ref		AGCACTGTAGTAGATGCATGCTCGAGCGGCC	ACGAGTGCGTCTAGGTATGGTAAATGCAGTA	22	175
Amp_4_2	HIV_Ref		AGCACTGTAGTAGATGCATGCTCGAGCGGCC	ACGCTCGACACTAGGTATGGTAAATGCAGTA	22	175
Amp_4_3	HIV_Ref		AGCACTGTAGTAGATGCATGCTCGAGCGGCC	AGACGCACTCCTAGGTATGGTAAATGCAGTA	22	175
Amp_4_4	HIV_Ref		AGCACTGTAGTAGATGCATGCTCGAGCGGCC	AGCACTGTAGCTAGGTATGGTAAATGCAGTA	22	175

Figure 10–49: A table of all 16 Amplicons where the MID sequences have been incorporated into the template-specific Primer 1 and Primer 2 sequences. In the highlighted "Amp_1_1" sequence, both primers begin with "ACGAGTGCGT" which is the MID1 sequence from Figure 10–48. Since the MID sequences are not actually present in the Reference, the Reference is constrained to the Amplicon being measured so that a single Reference could be used for all of the Amplicons. Otherwise 16 different MID-containing References would have to have been defined.

To finish the setup for a non-Multiplexer experiment, each of the 16 different Amplicons would have to be individually associated with its proper Sample, and those 16 Sample-Amplicon pairs would have to be associated with the Read Data Sets (Figure 10–50).

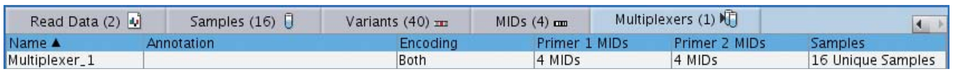


```
ferences    Read Data
⊟ nonMultiplexerMid
  ⊟ ReadGrp_1
    ⊟ ESS716001
      ⊟ Sample_1_1
          Amp_1_1
      ⊟ Sample_1_2
          Amp_1_2
      ⊟ Sample_1_3
          Amp_1_3
      ⊟ Sample_1_4
          Amp_1_4
      ⊞ Sample_2_1
      ⊞ Sample_2_2
      ⊞ Sample_2_3
      ⊞ Sample_2_4
      ⊞ Sample_3_1
      ⊞ Sample_3_2
      ⊞ Sample_3_3
      ⊞ Sample_3_4
      ⊞ Sample_4_1
      ⊞ Sample_4_2
      ⊞ Sample_4_3
      ⊞ Sample_4_4
    ⊟ ESS716002
      ⊞ Sample_1_1
      ⊞ Sample_1_2
      ⊞ Sample_1_3
      ⊞ Sample_1_4
      ⊞ Sample_2_1
      ⊞ Sample_2_2
      ⊞ Sample_2_3
      ⊞ Sample_2_4
      ⊞ Sample_3_1
      ⊞ Sample_3_2
      ⊞ Sample_3_3
      ⊞ Sample_3_4
      ⊞ Sample_4_1
      ⊞ Sample_4_2
      ⊞ Sample_4_3
      ⊞ Sample_4_4
```

Figure 10–50: Read Data Tree without Multiplexers. Each of the 16 Amplicons had to be individually assigned to a separate Sample and the Sample-Amplicons had to be assigned to the Read Data Sets.

10.6.5.2 Multiplexer Example

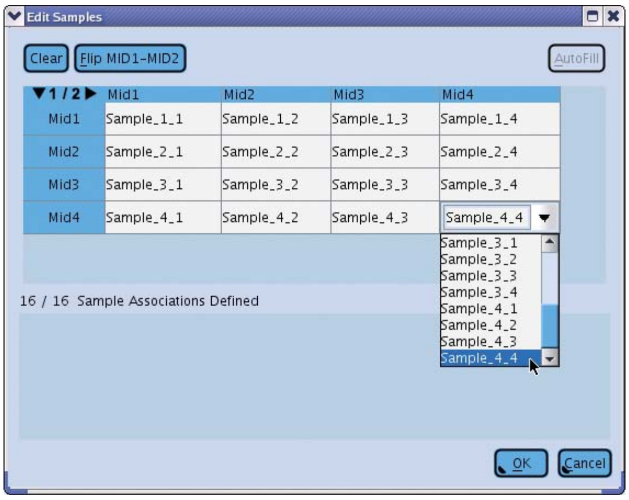
With the introduction of Multiplexers, there is no need to define 16 different Amplicons. Only the basic Amplicon in Figure 10–47 needs to be defined and the Multiplexer contains the information necessary to assign the reads to their proper Sample based on their MID content. This experiment only requires a single Multiplexer that can be used on both Read Data sets. The Multiplexer needs to have the "Both" encoding with 4 MID choices (Mid1, Mid2, Mid3, and Mid4) for Primer 1 MIDs and the same four choices for Primer 2 MIDs. The Multiplexer definition table is shown in Figure 10–51.



Name ▲	Annotation	Encoding	Primer 1 MIDs	Primer 2 MIDs	Samples
Multiplexer_1		Both	4 MIDs	4 MIDs	16 Unique Samples

Figure 10–51: Multiplexer definition table entry

This Multiplexer setup provides a 16 cell grid of Primer 1 – Primer 2 MID pairs that can be assigned to the appropriate Samples (Figure 10–52).



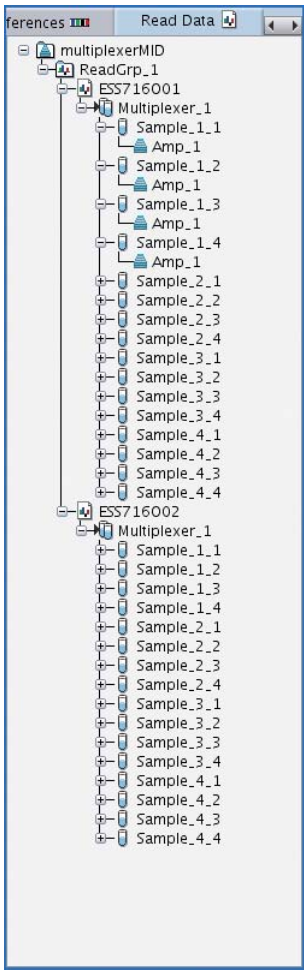
▼1 / 2▶	Mid1	Mid2	Mid3	Mid4
Mid1	Sample_1_1	Sample_1_2	Sample_1_3	Sample_1_4
Mid2	Sample_2_1	Sample_2_2	Sample_2_3	Sample_2_4
Mid3	Sample_3_1	Sample_3_2	Sample_3_3	Sample_3_4
Mid4	Sample_4_1	Sample_4_2	Sample_4_3	Sample_4_4

```
Sample_3_1
Sample_3_2
Sample_3_3
Sample_3_4
Sample_4_1
Sample_4_2
Sample_4_3
Sample_4_4
```

16 / 16 Sample Associations Defined

Figure 10–52: The Edit Samples window for the Multiplexer. The “Both” encoding being used allows all 16 cells in the MID grid to be assigned to distinct Samples.

To finish of the setup using Multiplexers, the Multiplexer has to be associated with the Read Data Sets. The single Amplicon being measured here can be associated with the Read Data Set – Multiplexer pair and the Amplicon automatically gets associated with each of the Samples encoded by the Multiplexer (see Figure 10–53).



```
multiplexerMID
  ReadGrp_1
    ESS716001
      Multiplexer_1
        Sample_1_1
          Amp_1
        Sample_1_2
          Amp_1
        Sample_1_3
          Amp_1
        Sample_1_4
          Amp_1
        Sample_2_1
        Sample_2_2
        Sample_2_3
        Sample_2_4
        Sample_3_1
        Sample_3_2
        Sample_3_3
        Sample_3_4
        Sample_4_1
        Sample_4_2
        Sample_4_3
        Sample_4_4
    ESS716002
      Multiplexer_1
        Sample_1_1
        Sample_1_2
        Sample_1_3
        Sample_1_4
        Sample_2_1
        Sample_2_2
        Sample_2_3
        Sample_2_4
        Sample_3_1
        Sample_3_2
        Sample_3_3
        Sample_3_4
        Sample_4_1
        Sample_4_2
        Sample_4_3
        Sample_4_4
```

Figure 10–53: Read Data Tree with Multiplexers. The Multiplexer is associated with each Read Data Set and the single Amplicon gets associated with each Read Data Set – Multiplexer. The Read Data Set – Multiplexer automatically associates the Amplicon with each of the underlying Samples encoded by the Multiplexer.

10.6.5.3 Multiplexer Benefits Summary

Multiplexers help prevent redundant data entry during project setup when using MIDs. Without Multiplexers, every sample would have to have its own Sample-specific Amplicon defined, and all of those Amplicons would contain some level of duplication of MID sequences contained within them. With Multiplexers, the MID sequences only need to entered once, and one can define the common portion of the Amplicon library product as a single Amplicon rather than needing to define as many Amplicons as there are Samples. Since the Multiplexers also contain the rules for associating an Amplicon with its proper Sample, there is no need to manually make individual Sample – Amplicon associations prior to associating the Sample with the Read Data Set. Instead, the Multiplexer gets associated with the Read Data set and associating the Amplicon with the Read Data Set – Multiplexer pair automatically generates the Sample – Amplicon relationships.

Beyond streamlined data entry, Multiplexers are also important for computational efficiency behind the scenes. The non-Multiplexer example provided in section 10.6.5.1 was included as an illustrative point, but it would run into trouble from a computational point of view. The 16 Amplicons only differ by at most 10 bases in each of their primers. When analyzing an individual read without any foreknowledge of MID specifics, the read needs to be compared against 16 very similar Amplicons. Allowing for distributed error in the read matches to the primer regions, it might be difficult to reliably assign a read to its proper Amplicon. With shorter MID sequences, this would be even more of a problem; the common portions of the primers from all of the Amplicons ends up making the differences in the MID regions seem less significant.

Multiplexers allow a read to be compared with expected template-specific primer sequences to first identify the Amplicon of the read. The knowledge of MID content and layout encoded by the Multiplexer allows the MID regions to be considered in a focused manner after the Amplicon assignment has already been established. This is more efficient and more likely to yield unambiguous results.

11. GS Amplicon Variant Analyzer Command Line Interface

The GS Amplicon Variant Analyzer (AVA) software includes a Command Line Interface (CLI) that allows the user to carry out various functions batch-wise, *e.g.* on various objects simultaneously and/or with multiple tasks queued through a script. This can afford the user substantial time savings compared with entering all data and carrying out all actions one at a time via the Graphical User Interface (GUI), as described in section 9.

The AVA-CLI is accessed via a command interpreter called "`doAmplicon`". The CLI has a flexible interface and depending on how it is invoked, you can either execute individual commands directly on the command line, read in a list of commands via a script file or a pipe, or type in commands manually in an interactive shell (see section 11.3.2.1 for the full usage statement for the `doAmplicon` command interpreter). The command language for the interpreter allows you to set up, manage, and compute Projects and trigger result reports (see section 11.4 for the full command language documentation).

11.1 Purpose of the CLI

The CLI in general allows many aspects of Project setup and management to be accomplished in a higher throughput manner than manipulating Projects via the GUI, where you must usually deal with elements on an individual basis. This can be especially useful in environments where large Amplicon Projects are carried out; or where Projects are carried out in large numbers, yet need to be kept strictly controlled and separate (*e.g.* if you are a sequencing service provider). More specifically, the CLI was primarily developed to meet four major needs: Data Import, Data Export, Automating the Triggering of Computations, and Result Reporting.

11.1.1 Data Import

In the GUI, you add Project objects one at a time; fully specifying objects like Reference Sequences and Amplicons can involve a lot of cutting and pasting. This is manageable for Projects where the number of Amplicons to be measured is small, but for more complex Projects, Project setup via the GUI can be taxing. The CLI remedies this situation by providing the ability to create Project objects via properly formatted tabular input information (see section 11.3.2.3 for more details on tabular input options). This means that files of Reference Sequences and Amplicons, *e.g.* provided by a client or perhaps generated from an in-house database, could be imported in bulk into the Project using "create" commands with the files as input (see section 11.4.4 for the create usage statement).

11.1.2 Data Export

It is often convenient to use an existing Project as the starting point for the creation of a new Project. For example, you may be measuring the same set of Amplicons for different Projects that have different owners and therefore need to be kept separate. Or, you may find that some set of Reference Sequences or Amplicons get reused across several different Projects. In such situations, the ability to export the information from an existing Project, and import it into a new Project, eliminates the duplication of labor and accumulated error risk of data re-entry. The CLI addresses this problem through a Project cloning command (the "utility clone" command usage statement is given in section 11.4.17.4) and a set of more focused "list" commands (see section 11.4.7 for the usage statement).

The "utility clone" command allows you to safely make a copy of any existing Project (with or without its accompanying Read Data) to a new location. The cloned Project can be renamed and edited to be appropriately used as a new Project.

The "list" command can be used in a more focused manner to export data for particular Project objects rather than exporting the setup of an entire Project. For example, you could use the command "list reference -outputFile referenceExport.txt" to export a table of Reference Sequences from a Project that could then be imported into another Project with the command "create reference -file referenceExport.txt".

11.1.3 Automating the Triggering of Computations

In the GUI you trigger the computation of a Project via a manual click on the Start button, on the Computation tab. In order to be able to truly automate the bulk of Amplicon Variant Analysis, you need a way to trigger computation independently of manual GUI interaction. The CLI has a set of "computation" commands (see section 11.4.3 for the usage statement) that allow you to control Project computation from the `doAmplicon` command interpreter, freeing you from GUI interaction through the computation stage.

11.1.4 Result Reporting

In the GUI, after you have run a computation, the Sample-Variant Table on the main Variants tab gets updated with Variant frequencies, and you can export the data from that table to a file manually. The CLI allows you to trigger the generation of the report using the "report variantHits" command (see section 11.4.12.1 for the usage statement). You can then choose to process this report on your own to prioritize which Variants are the most promising for user verification in the GUI.

11.2 AVA-CLI Command Language Overview

The AVA-CLI command language consists mainly of a set of commands to create, modify, and associate Project entities and to perform, and report the results of, Project computations. Additional commands exist for Project validation, data export, and setting the behavior of the `doAmplicon` command interpreter itself, to assist in debugging problems in CLI scripts. The commands and their command option specifier are all case-insensitive. The Project entities that can be created or manipulated, and their associated commands, are listed below. The usage statement for the `doAmplicon` command interpreter itself is in section 11.3.2.1. Help information that applies generally to the command language is presented in section 11.3. Detailed usage statements for individual commands are listed in section 11.4, providing a Reference Guide to the command language. Section 11.5 provides a more high-level overview of the language, including a full example script to create a new Project in section 11.5.15. Finally, another (more limited) example script is provided in section 11.6, to display some of the particular features of MID-based Projects.

11.2.1 Entities

There are 10 Project object or "entity" types supported by the AVA-CLI. Many of the CLI language commands can act on more than one of these.

- amplicon – This entity type may be abbreviated to "amp". See section 9.1.1.3 for more information on Amplicons.
- project – This entity type may be abbreviated to "proj". See section 9.1.1.1 for more information on Projects.
- readData – This entity type may not be abbreviated, but it is case insensitive so you don't have to capitalize the "D" of "Data"; this is done here only to improve readability. See section 9.1.1.4 for more information on Read Data Sets.
- readGroup – This entity type may not be abbreviated, but it is case insensitive so you don't have to capitalize the "G" of "Group"; this is done here only to improve readability. See section 9.1.1.4 for more details on Read Groups.
- reference – This entity type may be abbreviated to "ref". See section 9.1.1.2 for more information on Reference Sequences.
- sample – This entity type may be abbreviated to "samp". See section 9.1.1.6 for more information on Samples.
- variant – This entity type may be abbreviated to "var". See section 9.1.1.5 for more information on Variants.
- mid – This entity type does not need to be abbreviated and is used as "mid". See section 9.1.1.7 for more information on Mids.
- midGroup – This entity type may not be abbreviated, but it is case insensitive so you don't have to capitalize the "G" of "Group"; this is done here only to improve readability. See section 9.1.1.7 for more details on Mid Groups.
- multiplexer – This entity type may be abbreviated to "mul". See section 9.1.1.8 for more information on Multiplexers.

11.2.2 Available Commands

The top level commands recognized by the AVA-CLI are introduced briefly below. Some of the commands like "set" don't act directly on any entity type, while many others can be used to act on more than one entity type. The full set of online help showing the usage statements for the many ways these commands can be used may be found in section 11.4.

In general, you can create the Project and the objects within it using the "create" command (except the Read Data Sets, which must be added to the Project via a specialized "load" command). Many of the commands accept input in a tabular format that enables bulk import and allows you to process many objects at a time rather than forcing you to fully specify a command for each individual object. Once you have objects in the Project, they can be edited using the "update" and "rename" commands or they can be deleted from the Project entirely with the "remove" command. Associations between Read Data Sets, Samples, and Amplicons can be managed via the "associate" and "dissociate" commands. Computation can be automatically triggered and managed via the "computation" family of commands. Certain "utility" commands can be used to validate the Project and to export data. There are also several other commands (such as "set", "save", "close", and "exit") used to control interaction with the CLI and for other miscellaneous functions.

- associate – This command makes associations between appropriate records. It may be abbreviated to "assoc". It accepts tabular input. A full usage statement is available in section 11.4.1.
- close – This command closes the Project that is currently open. A full usage statement is available in section 11.4.2.
- computation – The command controls computations on the Project. It may be abbreviated to "comp". A full usage statement is available in section 11.4.3.
- create – This command creates entities including Projects, Reference Sequences, Amplicons, Samples, Variants, MIDs / MID Groups, Multiplexers, and Read Groups. It accepts tabular input. A full usage statement is available in section 11.4.4.
- dissociate – This command removes associations between records. It may be abbreviated to "dissoc". It accepts tabular input. A full usage statement is available in section 11.4.5.
- exit – This command exits the interpreter. A full usage statement is available in section 11.4.6.
- list – This command lists information about entities. It can optionally send output to a file. A full usage statement is available in section 11.4.7.
- load – This command loads Read Data Sets into the Project that is currently open. It accepts tabular input. A full usage statement is available in section 11.4.8.
- open – This command loads a pre-existing Project, making it the current open Project in the CLI. A full usage statement is available in section 11.4.9.
- remove – This command removes records from a Project. It accepts tabular input. A full usage statement is available in section 11.4.10.
- rename – This command renames entities. It accepts tabular input. A full usage statement is available in section 11.4.11.
- report – This command produces reports about computations. It can optionally send output to a file in tab-delimited or comma-separated value format. A full usage statement is available in section 11.4.12.
- save – This command saves any modifications to the Project that is currently open. A full usage statement is available in section 11.4.13.
- set – This command sets environment variables. A full usage statement is available in section 11.4.14.
- show – This command is used to show various information about the interpreter. A full usage statement is available in section 11.4.15.
- update – This command updates entities' properties. It accepts tabular input. A full usage statement is available in section 11.4.16.
- utility – This command performs utility functions such as Project cloning. A full usage statement is available in section 11.4.17.

11.3 AVA-CLI General Online Help

This section provides the verbatim content of the general online help files for the AVA-CLI. To enter the upper level of the help files, run the CLI “help” command. Information to access help on more detailed topics is as indicated below. The online help file content for each individual command, providing the full command usage statement, is provided in section 11.4. A more gentle introduction to the commands, with a full example script for setting up and performing computations on a Project, is given in section 11.5, and a smaller Project script displaying MID features is given in section 11.6.

11.3.1 Help

```
This provides an overview of available commands. For more specific
information, use ‘help <command>’ where ‘<command>’ can be one of the
following.  For example, for help on the ‘update’ command, run ‘help
update’.  For general help with the command interpreter, run
‘help general’.

create          Creates entities such as projects and project records.
open            Opens projects.
save            Saves the currently open project.
close           Closes the currently open project.
update          Updates entities’ properties.
rename          Renames entities.
remove          Removes records from a project.
load            Loads read data into the currently open project.
associate       Makes associations between appropriate records.
dissociate      Removes associations between records.
computation     Controls computations on the project definition.
report          Produces reports about computations.
list            Lists information about entities.
utility         Performs utility functions such as project cloning.
set             Sets environment variables.
show            Shows information about the interpreter settings.
exit            Exits the interpreter.
```

11.3.2 General Help

```
Find help on general use of the command interpreter in the sections below.
Run ‘help general <subsection>’.

commandLine         Information about the command line arguments to start
                        the command interpreter itself.
parsing             Information about how commands are parsed.
filePaths           Information about how file paths are interpreted.
tabularCommands     Information about using tables to succinctly construct
                        commands.
recordNames         Information about record naming for the command line
                        interpreter.
abbreviations       Information about abbreviations for options and
                        commands that can be used throughout the command
                        line interpreter.
multiplexing        Information about how the GS Amplicon Variant Analyzer
                        software supports multiplexing of amplicons and/or
                        samples within a single read data set.
```

11.3.2.1 CommandLine Help

```
doAmplicon [<files>] [-onErrors <"stop" or "continue">]
                     [-i[nteractive]]
                     [-v[erbose]]
                     [-c[ommand] <command>]
                     [-p[roject] <project path>]
                     [-h[elp]]
                     [-a[bout]]

Runs the command interpreter.  If no '<files>' are given, the interpreter
reads from standard input for its commands.  If one or more files is
given, each file is executed in order.  If "-" is encountered as one of
the file arguments, standard input is read for commands at that position.
For example:

     doAmplicon preamble.ava -

will execute the preamble and start reading commands from standard input.

The '-onErrors' option sets the value of the 'onErrors' parameter.  If
'onErrors' is set to "stop", the command interpreter will exit if an error
is encountered.  If 'onErrors' is set to "continue", the command
interpreter will abort the command that caused the error but will continue
running and executing subsequent commands.

The '-interactive' option indicates that the interpreter is being used
interactively.  A prompt is written to standard output and some commands
attempt to interact with the user for further input when necessary.

If neither '<files>' nor the '-command' option are given, the interpreter
implicitly enters interactive mode, even if '-interactive' is not
specified.  If "-" is supplied as one of the '<files>' option, then the
interpreter will read from standard input, but will not implicitly enter
interactive mode.  Thus, one syntax allows the interpreter to be used
as an interactive command line interface while the other facilitates the
creation of automated pipelined scripts, as in:

    generateScript | doAmplicon - > resultFile

Unless explicitly given an '-onErrors' option value, the interpreter, in
interactive mode, behaves as if 'onErrors' were set to "continue" and, in
non-interactive mode, behaves as if 'onErrors' were set to "stop."

The '-verbose' option will cause the interpreter to output information
about the commands that it is executing as it executes them.

The '-command' option can be used to execute a single command in the
interpreter.  For example, if you want to create an empty project, you
would execute:

    doAmplicon -command "create project /data/new/project/path"

The '-project' option can be used to open a project before executing the
rest of the specified commands.  For example, you may have a script that
removes all of the variants in a project.  You could choose on what project
to run this script by using the '-project' option.  For example:

    doAmplicon -project "/some/project" myRemoveVariants.ava

Combined with the "-command" option, this can be used to execute single
commands on a project.  For example:

    doAmplicon -project "/some/project" -command "list amplicon"

This command will list all of the amplicons of the project at
"/some/project".

The '-project' option attempts to open the project with exclusive control
and will fail if another instance of the program has control of the
project.  To attempt to preempt control of the project or open it in
a read-only fashion, requires the use of the 'open' command from within
the interpreter itself.

The '-help' option displays this help.  Online help for the interpreter
commands is available by entering the "help" command to interpreter
itself.

The '-about' option displays version information about the interpreter.
```

11.3.2.2 Parsing Help

The interpreter is case insensitive with respect to its commands and options. For example, consider the two commands below:

```
create amplicon Amp1
CREATE AMPLICON Amp1
```

These commands are equivalent. Note, however, that all strings that are part of the project itself are case sensitive. For example, consider the two commands below:

```
create amplicon Amp1
create amplicon AMP1
```

These commands are not equivalent, since record names are case sensitive.

The character ‘#’ may be used to document your scripts. The ‘#’ may appear anywhere on the line, and everything from the ‘#’ until the end of the line is ignored. For example:

```
# The next command line lists the Variants of the project
list variant
list amplicon # and this command lists the Amplicons
```

To use an argument that contain spaces or the comment character, surround the argument with double quotes. For example, you can set an annotation of an amplicon to an unusual string by running the following:

```
update amplicon Amp1 -annotation “My unusual string with a
  comment # character and even line breaks
”
```

If you need a double quote in your argument, “escape” it with a preceding backslash. For example:

```
update amplicon Amp1 -annotation “The \”Best\” Amplicon”
```

Inside of double quotes, the backslash (except when preceding a double quote) and new line characters are treated literally. Thus, in the example:

```
update amplicon Amp1 -annotation “Testing 1 \
2 3”
```

both the backslash and new line will become part of the annotation.

Outside of double quotes, the backslash character can be used to make any single character ordinary, avoiding the need to use double quotes. Thus, the following two commands are equivalent:

```
create amplicon Amp\#1
create amplicon “Amp#1”
```

Note that without the ‘\’ or surrounding quotes, the #1 in the both of the commands above would have been treated as a comment, and an amplicon simply named “Amp” would have been created.

Outside of double quotes, the backslash character can also be used for line continuation, allowing you to split a command over multiple lines. A backslash immediately followed by a new line will join the following line to the current line. This allows you to format commands nicely. For example:

```
update amplicon Amp1 \
    -annotation “The best amplicon” \
    -reference “ref1”
```

is equivalent to the single line command:

```
update amplicon Amp1 -annotation “The best amplicon” -reference “ref1”
```

11.3.2.3 Tabular Commands Help

To facilitate high-throughput project setup and modification, it is possible to run commands with tables of data as input. The table column headers are simply the options of the command that is to be run, but with the leading "-" removed. As with the command options themselves, the column headers are case insensitive. The tabular data may be supplied from an external file, or from a table embedded in the command script itself, using tab or comma separated value formats.

For example, suppose you need to add 100 amplicons to a project. Instead of adding them one by one with 'create amplicon' commands, you can issue a single 'create amplicon' with a table as input. For example:

```
create amplicon -file - << end_marker
Name      Reference
Amp1      Ref1
Amp2      Ref2
Amp3      Ref3
Amp4      Ref4
Amp5      Ref5
Amp6      Ref6
Amp7      Ref7
Amp8      Ref8
end_marker
```

This command will create 8 amplicons when run. Let us examine each element of this invocation. First, the 'create amplicon' indicates that we are creating amplicons. The '-file - <<' option indicates that we are going to be supplying a table in the form of a "Here" document. A "Here" document is essentially a document supplied to the command that can be specified in place. The 'end_marker' indicates that we are creating a here document that terminates when 'end_marker' is seen by itself on a line.

The document itself must be a tab-separated table whose first row indicates what option each column represents. Thus, when the second line of our table is executed, it is precisely the same as if we were to have written:

```
create amplicon -name "Amp1" -reference "Ref1"
```

In fact, our table command is the same as executing the following:

```
create amplicon -name "Amp1" -reference "Ref1"
create amplicon -name "Amp2" -reference "Ref2"
create amplicon -name "Amp3" -reference "Ref3"
create amplicon -name "Amp4" -reference "Ref4"
create amplicon -name "Amp5" -reference "Ref5"
create amplicon -name "Amp6" -reference "Ref6"
create amplicon -name "Amp7" -reference "Ref7"
create amplicon -name "Amp8" -reference "Ref8"
```

However, it is much more succinct in table form.

This works for any command that takes a '-file' option. For example:

```
update reference -annotation "Updated 2/12/07" -file - << end
Name     Sequence
Ref1     ATAGCAGATAGATAATATATAAAAAAGACGAT
Ref2     ATAGCAGATATAGATAGTGATGCAGTATAGACAGTAAGATAGACAG
Ref3     ATGAATAAAAAATCCCCCCCTAGTAGTACTTTTTTAAAAATA
Ref4     TGACGAAACATAGTGTAAACGTGTGCAGACAGCCCAC
Ref5     GCAGACGATAAAAAAATGATGACGACGTAATACAATAT
Ref6     GACGCATTTTTTTTAGATATACTATATATT
Ref7     TATAATAAAAATTATATCGGGATAGTAGTGCAGAGAGAGAGTAGTAGCAC
Ref8     TACGACATATAGATGATAGACAAATAACAGATAGTAGTAGTAGAAGT
end
```

This time we are updating references rather than creating amplicons. You will also note that we specified an annotation in the main command and not in the here document. Options specified in this manner are applied to each row of the command. Our table command is the same as executing the following:

```
update reference -annotation "Updated 2/12/07" -reference Ref1 -sequence ..
update reference -annotation "Updated 2/12/07" -reference Ref2 -sequence ..
update reference -annotation "Updated 2/12/07" -reference Ref3 -sequence ..
update reference -annotation "Updated 2/12/07" -reference Ref4 -sequence ..
update reference -annotation "Updated 2/12/07" -reference Ref5 -sequence ..
update reference -annotation "Updated 2/12/07" -reference Ref6 -sequence ..
update reference -annotation "Updated 2/12/07" -reference Ref7 -sequence ..
update reference -annotation "Updated 2/12/07" -reference Ref8 -sequence ..
```

Instead of using here documents, external files can be supplied using the '-file' option. For example:

```
create variant -file /data/variants.txt
```

In the previous examples, we specified the table in place using a here document. Here, we refer to the external file, "/data/variants.txt". The format of the external file is expected to be exactly the same as that of the here document (without the need for an end marker, however).

So "/data/variants.txt" could look like this:

```
<begin /data/variants.txt>
Name      Reference   Status
Var1      Ref1        Accepted
Var2      Ref1        Accepted
Var3      Ref1        Accepted
Var4      Ref1        Rejected
Var5      Ref1        Rejected
Var6      Ref1        Accepted
<end /data/variants.txt>
```

You will also note in this example that the rows do not line up exactly. This is because we always expect one tab character to separate each column, regardless of the size of the data in the column.

If you prefer comma-separated columns, use the '-format' option. For example:

```
update readData -file - -format csv << end
Name,Active
Data1,true
Data2,true
Data3,false
Data4,true
Data5,false
end
```

Note the '-format csv' option. Valid values are "csv" and "tsv" to indicate comma-separated and tab-separated table formats, respectively. The default is "tsv", except when a file is provided with a ".csv" extension (such as those exported from Excel).

It is also important to note that empty cells are not omitted from the arguments. For example:

```
update variant -file - << end
Name      Reference
Var1      Ref1
Var2
Var3
Var4      Ref4
Var5
Var6      Ref6
Var7      Ref7
Var8      Ref8
end
```

Executing this command will make variants "Var2", "Var3", and "Var5" refer to no reference sequence.

Finally, note that the parsed table values are what are used to supply values to the command arguments, as opposed to the literal table text itself. This means that the table contents must follow the syntactic conventions of tab and comma separated values tables, not that of the command interpreter. In particular, this means that neither the interpreter's comment character '#' nor the special '\' constructs have any special meaning inside of tables. Similarly the conventions for quoting double quotes in tables should be followed.

Rather than, as one would embed a '"' in a command line argument:

```
"This is how \"double quotes\" are embedded for the interface"
```

in a table, one must use the double-double quote convention:

```
"Tables use ""double-double quotes"" to embed a quote character"
```

For more information on the interpreter's parsing of commands and special characters, run 'help general parsing'.

11.3.2.4 Record Names Help

The command line interpreter primarily uses record names to identify and distinguish records. Duplicate record names lead to ambiguity that the interpreter cannot resolve in most cases. For example, it is technically allowed for two reference sequences to have the same name, “Ref1”. If we want to update one of these reference, we issue the command:

```
update reference “Ref1” -annotation “New annotation”
```

This will report an error, since the interpreter cannot discern which record to update. It is therefore recommended that unique names be used for records.

There are exceptions to this rule. Amplicons and variants can be disambiguated by their reference sequences if duplicate names are found. For example, if you have two amplicons named “Amp1”, but one of them refers to reference “Ref1” and the other to “Ref2”, the ‘-ofRef’ option of commands dealing with amplicons can be used to disambiguate them. For example:

```
update amplicon “Amp1” -annotation “New annotation”
```

This will result in an error, since there are two amplicons named “Amp1”. However, consider this command:

```
update amplicon “Amp1” -ofRef “Ref1” -annotation “New annotation”
```

This is allowed because the ‘-ofRef’ option has been used to determine which amplicon to update. This can be used in other commands as well:

```
associate -sample “Sam1” -amplicon “Amp1” -ofRef “Ref1”
```

Again, we are distinguishing between the duplicately named amplicons by using the ‘-ofRef’ option.

The ‘utility validateNames’ command is provided to help determine if your project has any such ambiguity and if so, help correct. Type ‘help utility validateNames’ for more information.

11.3.2.5 Abbreviations Help

Many commands and options can be abbreviated. For example:

```
create amp Amp1
```

This command is the same as:

```
create amplicon Amp1
```

Such abbreviations are noted in the help documentation. For example, the documentation for ‘create amplicon’ specifies:

```
create amp[licon]
```

to indicate that it can be abbreviated as such. This also goes for some options. For example:

```
assoc -sam “Sam1” -amp “Amp1”
```

This is the same as:

```
associate -sample “Sam1” -amplicon “Amp1”
```

The option abbreviations are also similarly noted in the help documentation.

11.3.2.6 File Paths Help

```
File paths are used in commands to specify projects, script files, tabular
data, and, more generally, the location of input and output files.  For
example, records can be listed to a file:

    list amplicon -outputFile someFile.txt

or other scripts can be executed:

    utility execute someOtherScript.ava
```

In these examples, relative paths (*i.e.*, paths that don’t start with a ‘/’) specify the files to use. These paths are considered relative to the interpreter’s current directory (currDir), which may be set with the ‘set currDir’ command.

When the interpreter starts, the currDir is initially set to the directory in which the interpreter was invoked. For example, if the current working directory is /home/me/projects when doAmplicon is invoked, the initial currDir will be /home/me/projects. In this situation, for the example above, the relative path someFile.txt would be resolved to the absolute path /home/me/projects/someFile.txt.

If ‘set currDir’ is used, the file resolution will change. For example:

```
    set currDir /some/other/directory
    list amplicon -outputFile someFile.txt
```

Now the relative path someFile.txt will be resolved to the absolute path /some/other/directory/someFile.txt.

A few special path prefix shortcuts, denoted with a leading ‘%’, are also available to make specifying files easier. The first of these, currDir, has already been described. This may be used to explicitly specify the currDir in a path, but is entirely equivalent to the default interpretation of relative paths. For example, “%currDir/someFile.txt” and “someFile.txt” will refer to the same file.

There is also a special path prefix shortcut to access the user’s home directory. For example, if the user’s home directory is /home/me, the path “%homeDir/someFile.txt” will be resolved to the absolute path /home/me/someFile.txt.

Finally, there is a special path prefix shortcut, libDir, to access a system library path that is set up as part of installation of the software. This provides access to a standard library that may be modified by the site administrator.

Path prefixes are only recognized when they prefix the path and match a known shortcut. For example, suppose the values of the shortcuts are as follows:

```
    currDir=/some/dir
    homeDir=/home/me
    libDir=/opt/454/apps/amplicons/config/lib
```

Only paths starting with “%currDir”, “%homeDir”, or “%libDir”, respectively, will be affected by shortcuts. Here are some example path specifications with their shortcut-expanded versions:

```
    %currDir/someFile.txt           => /some/dir/someFile.txt
    %homeDir/someFile.txt           => /home/me/someFile.txt
    %libDir/someFile.txt            =>
                          /opt/454/apps/amplicons/config/lib/someFile.txt
    someFile.txt                    => /some/dir/someFile.txt
    %otherDir/someFile.txt          => /some/dir/%otherDir/someFile.txt
    data/%currDir/someFile.txt      => /some/dir/data/%currDir/someFile.txt
```

The last example does not expand the %currDir shortcut because it does not appear at the beginning of the path specification. The second to last example interprets ‘%otherDir’ literally, and resolves the given path relative to the currDir value of /some/dir, because %otherDir is not one of the defined shortcuts.

Absolute paths (*i.e.*, paths that begin with ‘/’) may also be used. Such paths are entirely unaffected by the currDir and by shortcuts.

To see the values of the shortcut prefixes, use the ‘show environment’ command.

11.3.2.7 Multiplexing Help

```
The GS Amplicon Variant Analyzer (AVA) software provides a number of
mechanisms for multiplexing reads, allowing multiple amplicons from
the same or different samples to be simultaneously sequenced within
a PTP region.

The simplest demultiplexing method relies on the sequence specific primer
regions of the amplicons.  If an experiment calls for measuring multiple
distinct amplicons from the same sample, those amplicons may be mixed
together in a PTP region.  The project setup allows different amplicons to
be associated with different samples, so it is also possible to multiplex
reads from different samples, providing the samples are constructed such
that each sample is comprised of reads from different amplicons.

However, if a user wants to sequence reads from different samples but the
same amplicons, the sequence specific primer information for the amplicons
is no longer sufficient for demultiplexing the reads to their appropriate
samples.  To allow multiplexing of samples with the same amplicon in a PTP
region, the Multiplex Identifier (MID) approach is supported, in which
bases are added adjacent to the sequence specific primer in order to label
an amplicon’s sample.

Two types of chemistry are supported.  With ‘adaptor’ chemistry the MIDs
are added (typically via ligation) to the amplified sample DNA.  The MIDs
start out directly linked to the sequencing adaptors and are ultimately
sandwiched between those adaptors and the target sequence amplification
primers.  The advantage of this chemistry is that it allows the use of
amplification primers that are completely neutral with respect to the MIDs
that may be ultimately employed.  The disadvantage is that an extra step
is required to augment the sequences with MIDs.  Alternatively, the default
‘amplicon’ chemistry may be used, in which the MIDs are constituent parts
of the amplification primers.  These amplicon chemistry primers are
synthesized to contain the sequencing adaptor, the MID, and the
target-specific amplification primer as a single construct (one each for
the 5’ and 3’ ends of the target sequence).  The advantage of this
chemistry is that the sample amplification step itself labels the reads
with MIDs as well as provides the necessary sequencing adaptors.  This
single step approach may reduce the chance of sample misassignments due
to laboratory errors.  The principal disadvantage is that separate
amplification primers need to be synthesized for each MID, and a commitment
to specific MIDs must be made prior to sample amplification.

With the ‘amplicon’ chemistry, MIDs are technically part of the amplicon
primer, but if they were encoded as such in a project, the user would have
to enter as many versions of an amplicon as there are samples to be
demultiplexed in a given region.  For simplicity, the AVA software allows
the specification of amplicons in a manner that is independent of whether
MIDs are employed, and provides a separate ‘multiplexer’ formalism that
describes the MID to sample relationships.  The AVA software automatically
combines, for the user, the MIDs of a multiplexer with the primers of an
amplicon and applies the multiplexer’s MID-sample relationships to
determine the sample to which a given read belongs.  This facilitates
project setup since multiple amplicons can share the same MID to sample
relationship information, with that information being defined just once in
a single multiplexer.

This also allows the MID specification (encapsulated in the multiplexer) to
be shared across multiple read data, in the event that the MID-sample
relationships are replicated in more than one read data of the experiment.

The use of multiplexers provides the following benefits:
    1) Separation of amplicon specification from the complexities of MIDs
    2) The sharing of MID to sample relationships across multiple amplicons
    3) The sharing of such information across multiple read data

The ‘associate’ command provides the ability to define both MID-based and
non-MID-based multiplexing relationships.  Run ‘help associate’ for more
details on how to create these multiplexing relationships.  For more
information on creating multiplexers, and their associated constituents,
run ‘help create multiplexer’, ‘help create mid’, and
‘help create midGroup’.
```

11.4 AVA-CLI Command Usage Statements

This section provides the verbatim content of the online help files for each individual command, providing the full command usage statement. Each is accessed by typing "help <command>" in the CLI (as described in section 11.3.1).

11.4.1 associate

```
assoc[iate] -sam[ple] <sample name>
            -amp[licon] <amplicon name> [-ofRef <reference sequence name>]
            [-file <file> [-format <format>]]

assoc[iate] -sam[ple] <sample name>
            -readData <read data name> | -readGroup <read group name>
            [-file <file> [-format <format>]]

assoc[iate] -sam[ple] <sample name>
            -amp[licon] <amplicon name> [-ofRef <reference sequence name>]
            -readData <read data name> | -readGroup <read group name>
            [-file <file> [-format <format>]]

assoc[iate] -mul[tiplexer] <multiplexer name>
            [-primer1Mid <primer1Mid name>
                [-ofPrimer1MidGroup <primer1MidGroup name>]]
            [-primer2Mid <primer2Mid name>
                [-ofPrimer2MidGroup <primer2MidGroup name>]]
            -checkMid <boolean>
            [-file <file> [-format <format>]]

assoc[iate] -mul[tiplexer] <multiplexer name>
            [-primer1Mid <primer1Mid name>
                [-ofPrimer1MidGroup <primer1MidGroup name>]]
            [-primer2Mid <primer2Mid name>
                [-ofPrimer2MidGroup <primer2MidGroup name>]]
            -checkMid <boolean>
            -sam[ple] <sample name>
            [-file <file> [-format <format>]]

assoc[iate] -mul[tiplexer] <multiplexer name>
            -amp[licon] <amplicon name> [-ofRef <reference sequence name>]
            -readData <read data name> | -readGroup <read group name>
            [-file <file> [-format <format>]]

assoc[iate] -mul[tiplexer] <multiplexer name>
            -readData <read data name> | -readGroup <read group name>
            [-file <file> [-format <format>]]

assoc[iate] -mul[tiplexer] <multiplexer name>
            [-primer1Mid <primer1Mid name>
                [-ofPrimer1MidGroup <primer1MidGroup name>]]
            [-primer2Mid <primer2Mid name>
                [-ofPrimer2MidGroup <primer2MidGroup name>]]
            -checkMid <boolean>
            -sam[ple] <sample name>
            -amp[licon] <amplicon name> [-ofRef <reference sequence name>]
            -readData <read data name> | -readGroup <read group name>
            [-file <file> [-format <format>]]
```

```
The 'associate' command is used to associate records in many-to-many
relationships.  Such relationships can exist between samples, amplicons,
read data, muliplexers, and MIDs.  When a particular association is made,
any more general associations that would be logically implied by the
original association will automatically be created (e.g., associating the
triplet of a sample, amplicon, and read data will implicitly create the
pairwise sample-amplicon association as well).

In any of the command forms above where '-amplicon' is being specified,
the '-ofRef' option can be used to disambiguate amplicons with the same
name but which are from different reference sequences.

The '-amplicon' option may be specified as a "*" to allow multiple
amplicons to be associated with a single command.  If a "*" is passed as
the '-amplicon' option value, with no '-ofRef' option, the "*" is
interpreted to indicate that all of the amplicons known in the project at
the time of the invocation will be involved in the association.  If both
the "*" value and '-ofRef' option are used, then all amplicons in the
project at the time of invocation that are of the given reference sequence
will be involved in the association.
```

```
In a similar manner to the ‘-ofRef’ option for amplicons, the
‘-ofPrimer1MidGroup’ and ‘-ofPrimer2MidGroup’ options can be used to
disambiguate ‘-primer1Mid’ and ‘-primer2Mid’ specifications, respectively.

The ‘-primer1Mid’ and ‘-primer2Mid’ options may also be specified as a
“*”.  If no ‘-ofPrimer1MidGroup’ or ‘-ofPrimer2MidGroup’ option is
supplied, the “*” refers to all the MIDs of the project.  If a MID group
is specified, the “*” refers to only the MIDs of that MID group.

The ‘-checkMid’ option is used to verify that the MIDs associated with the
primer1 side of the multiplexer are all mutually compatible as are the
primer2 side MIDs.  The definition of compatibility is the same as that
used by the ‘-checkMidGroup’ option of the ‘create mid’ command (defined
sequences are compatible if they are of the same length and are
non-identical, with undefined zero-length sequences allowed).  This option
must be ‘true’ or ‘false’, and defaults to ‘true’ if not provided.

Read data can be specified in a command as either a specific read data
using the ‘-readData’ option or as a collection of read data using the
‘-readGroup’ option.  When ‘-readGroup’ is used, it is as if the command is
being run multiple times (once for each particular read data in the read
group at the time of invocation).  For example, if readGroup1 has 3
read data in it (readData1, readData2, and readData3), running the command:
    associate -sample sample1 -readGroup readGroup1
and running the commands:
    associate -sample sample1 -readData readData1
    associate -sample sample1 -readData readData2
    associate -sample sample1 -readData readData3
would have the same net effect.  In those situations where identical
associations can correctly be made with each of the read data in a read
group, the ‘-readGroup’ allows the command to be more concise.

Explanations of the various command forms are as follows:

Run ‘help general tabularCommands’ for information about the ‘-file’
option.

assoc[iate] -sam[ple] <sample name>
            -amp[licon] <amplicon name> [-ofRef <reference sequence name>]
            [-file <file> [-format <format>]]

When only a sample and amplicon are specified, an association is created
between them.  The ‘-ofRef’ option can be used to disambiguate amplicons
with the same name that refer to different reference sequences.  If a “*”
is passed as the ‘-amplicon’ option value, with no ‘-ofRef’ option, all
amplicons known in the project are associated with the sample.  If both the
“*” value and ‘-ofRef’ option are used, then all amplicons of the given
reference sequence are associated with the sample.

assoc[iate] -sam[ple] <sample name>
            -readData <read data name> | -readGroup <read group name>
            [-file <file> [-format <format>]]

When a sample and read data are specified, associations are created between
the sample itself, all of the current amplicons associated with the sample,
and the read data.  For example, running
    associate -sample sample1 -readData read1
will associate “sample1” and all of its amplicons at the time of invocation
with “read1”.

Similarly, when a sample and read group are specified, associations are
created between the sample itself, all of the current amplicons associated
with the sample, and all of the read data in the read group.  For example,
running
    associate -sample sample1 -readGroup group1
will associate “sample1” and all of its amplicons at the time of invocation
with all of the read data in “group1”.

assoc[iate] -sam[ple] <sample name>
            -amp[licon] <amplicon name> [-ofRef <reference sequence name>]
            -readData <read data name> | -readGroup <read group name>
            [-file <file> [-format <format>]]

If a sample, an amplicon, and read data are specified, an association is
created between the sample itself, the amplicon, and the read data.  The
‘-ofRef’ option can be used to disambiguate amplicons with the same name
that refer to different reference sequences.  If a “*” is passed as the
‘-amplicon’ option value,  with no ‘-ofRef’ option, all amplicons known
in the project are associated with the sample and read data.  If both the
“*” value and ‘-ofRef’ option are used, then all amplicons of the given
reference sequence are associated with the sample and read data.  If a read
group is specified instead of a single read data, the sample and
amplicon(s) are associated with all of the read data in the read group.
```

In general, creating a triplet association between an amplicon, sample and some read data implicitly creates a simultaneous paired association between the amplicon and sample. If an amplicon is already associated with some sample on a particular read data set, any attempt to associate the amplicon with a different sample on the same read data set will be ignored (but with a warning) since the demultiplexing constraints only allow amplicons to be associated with individual samples in the context of any individual read data set (an amplicon may be associated with different samples on different read data sets, and different amplicons may be associated with different samples on the same read data, however).

```
assoc[iate] -mul[tiplexer] <multiplexer name>
            [-primer1Mid <primer1Mid name>
                [-ofPrimer1MidGroup <primer1MidGroup name>]]
            [-primer2Mid <primer2Mid name>
                [-ofPrimer2MidGroup <primer2MidGroup name>]]
            -checkMid <boolean>
            [-file <file> [-format <format>]]
```

When some combination of MIDs and a multiplexer are specified, the MIDs will be associated with the multiplexer. For either of the MID options (‘-primer1Mid’ and ‘-primer2Mid’), all of the MIDs in the project may be specified at the same time by using a “*” for the value. The ‘-ofPrimer1MidGroup’ and ‘-ofPrimer2MidGroup’ options can be used to restrict the “*” set of MIDs to those of a particular MID group (or to disambiguate MIDs with the same name from different MID groups). If it becomes necessary to temporarily associate inconsistent MIDs on a primer MID side, ‘-checkMid’ can be set to ‘false’.

Although it might be a more common case to use the next command below to simultaneously associate a multiplexer, MIDs, and a sample in a single operation (letting the implied multiplexer-MID associations be made automatically), the above command can be useful to specify MIDs that are not explicitly tied to samples. In particular, if compatible MIDs have been used in recent experiments, but are not present in this experiment, associating them to the multiplexer without samples can help prevent potential contaminants from being assigned to samples of this experiment.

```
assoc[iate] -mul[tiplexer] <multiplexer name>
            [-primer1Mid <primer1Mid name>
                [-ofPrimer1MidGroup <primer1MidGroup name>]]
            [-primer2Mid <primer2Mid name>
                [-ofPrimer2MidGroup <primer2MidGroup name>]]
            -checkMid <boolean>
            -sam[ple] <sample name>
            [-file <file> [-format <format>]]
```

When some combination of MIDs, a multiplexer, and a sample are specified, the sample is associated with a particular MID configuration of the multiplexer. There are restrictions on how the MID options (‘-primer1Mid’ and ‘-primer2Mid’) can be used that depend on the ‘-encoding’ type of the multiplexer. If the encoding type is ‘both’, it is required that both MID options be provided in order to associate the sample with a pair of MIDs. If the encoding type is ‘either’, ‘primer1’, or ‘primer2’, it is only necessary to supply one of the MID options at a time. In the ‘either’ case, specifying both options at the same time is allowed. In the ‘primer1’ and ‘primer2’ cases, the MID option of the proper type must be used (*e.g.*, the ‘primer1’ encoding type requires that the ‘-primer1Mid’ option be used). Any implied multiplexer-MID associations that were not explicitly set up previously will automatically be created as a consequence of this command. A sample may be associated with more than one MID configuration, but each MID configuration may only map to a single sample (*e.g.*, in an ‘primer1’ configuration, sample1 may be associated with both MID1 and MID2 on the primer1 side, but those MIDs could not simultaneously be associated with a different sample2).

```
assoc[iate] -mul[tiplexer] <multiplexer name>
            -amp[licon] <amplicon name> [-ofRef <reference sequence name>]
            -readData <read data name> | -readGroup <read group name>
            [-file <file> [-format <format>]]
```

When a multiplexer, amplicon and read data are specified, the amplicon will be associated with the read data-multiplexer context. As a result, the amplicon automatically becomes associated with each of the samples associated with the multiplexer (creating any missing sample-amplicon relationships along the way). A “*” may be provided with the ‘-amplicon’ option to indicate that all amplicons in the project should be associated. The ‘-ofRef’ option can be used, if necessary, to disambiguate amplicons with the same name or to restrict the “*” set of amplicons to those of the specified reference sequence. The association may be made with a single read data using the ‘-readData’ option, or the associations can be made at once for all of the read data within a read group using the ‘-readGroup’ option. Multiplexers may be associated with several different read data sets. Each of those read data-multiplexer contexts are allowed to have

```
different amplicon associations, but the internal MID-sample associations
remain the same in each of those contexts.  If the  multiplexer is not
already associated with the specified read data when this command is
invoked, those read data-multiplexer associations are created
automatically.

assoc[iate] -mul[tiplexer] <multiplexer name>
            -readData <read data name> | -readGroup <read group name>
            [-file <file> [-format <format>]]

When a multiplexer and a read data are specified, the association between
the pair is created.  This provides a read data-multiplexer context that is
ready to accept amplicon associations (which get processed through the
multiplexer's MID configuration to get distributed to all the associated
samples).  This command is performed automatically as a consequence of the
more specific command above that also specifies an amplicon.

assoc[iate] -mul[tiplexer] <multiplexer name>
            [-primer1Mid <primer1Mid name>
                [-ofPrimer1MidGroup <primer1MidGroup name>]]
            [-primer2Mid <primer2Mid name>
                [-ofPrimer2MidGroup <primer2MidGroup name>]]
            -checkMid <boolean>
            -sam[ple] <sample name>
            -amp[licon] <amplicon name> [-ofRef <reference sequence name>]
            -readData <read data name> | -readGroup <read group name>
            [-file <file> [-format <format>]]

It is also possible to specify all of the entities in a relationship at
once.  Provided that all of the individual entities have already been
created, this command creates a specific read data-multiplexer-amplicon
context where the multiplexer has an MID configuration that allows valid
mapping to samples.  This allows the AVA software to analyze reads at the
amplicon level and properly demultiplex them to their appropriate samples.
All of the necessary implied relationships in the command would be created
automatically.  A string of commands of this type might be used to define
the associations for an entire project.  This would not be a good strategy
when typing in commands by hand, but it could be convenient for setup
scripts that are programmatically generated using nested loops and tabular
commands.
```

11.4.2 close

```
close

Closes the current project.  This releases the lock held on the project
(unless it was opened "readOnly") and discards any unsaved changes.  A
project must be created or opened before executing commands that require an
open project.
```

11.4.3 computation

```
comp[utation] <computation command>

comp[utation] start
comp[utation] stop
comp[utation] status
comp[utation] loadDetectedVariants

The 'computation' command is used to control and query for information
about computations on the currently open project.

The following computation commands are available.  Run
'help computation <computation command>' for more detailed information.

start                   Starts a computation on the currently open project.
stop                    Stops a running computation on the currently
                            open project.
status                  Prints the status of computation on the currently
                            open project.
loadDetectedVariants    Loads variants into the currently open project that
                            were automatically detected during computation.
```

11.4.3.1 computation start

```
comp[utation] start

Starts a computation on the currently open project.
```

11.4.3.2 computation stop

```
comp[utation] stop

Stops a running computation on the currently open project.
```

11.4.3.3 computation status

```
comp[utation] status

Prints the status of computation on the currently open project.  If a
computation is currently running, "running" will be printed.  If no
computation is currently running, "stopped" will be printed.
```

11.4.3.4 computation loadDetectedVariants

```
comp[utation] loadDetectedVariants

Loads variants into the currently open project that were automatically
detected (i.e., not in list of predefined project Variants, but
automatically discovered by the software) during computation.
```

11.4.4 create

```
create <entity type> <other arguments>

The ‘create’ command is used to create new entities.  The type of entity to
create is determined by the ‘<entity type>’ argument.  The
‘<other arguments>’ are determined by the entity type.  For example, to
create a project, you can run ‘create project /path/to/new/project’.  This
will create a new project at ‘/path/to/new/project’.  To create a new
amplicon, you can run ‘create amplicon “MyAmplicon”’.

The following entities are available for creation.  Run
‘help create <entity type>’ for more detailed information.

amplicon        Creates an amplicon in the currently open project.
mid             Creates a new MID in the currently open project.
midGroup        Creates a new MID group in the currently open project.
multiplexer     Creates a new multiplexer in the currently open project.
project         Creates a new project.
readGroup       Creates a read group in the currently open project.
reference       Creates a reference sequence in the currently open project.
sample          Creates a sample in the currently open project.
variant         Creates a variant in the currently open project.
```

11.4.4.1 create amplicon

```
create amp[licon] <new amplicon name> [-orUpdate]
                                      [-ofRef <reference name>]
                                      [-annot[ation] <annotation>]
                                      [-ref[erence] <reference name>]
                                      [-primer1 <primer 1 sequence>]
                                      [-primer2 <primer 2 sequence>]
                                      [-start <target start index>]
                                      [-end <target end index>]
                                      [-checkPri[merMatch] <boolean>]
                                      [-file <file> [-format <format>]]

create amp[licon] -name <new amplicon name>
                  [-orUpdate]
                  [-ofRef <reference name>]
                  [-annot[ation] <annotation>]
                  [-ref[erence] <reference name>]
                  [-primer1 <primer 1 sequence>]
                  [-primer2 <primer 2 sequence>]
                  [-start <target start index>]
                  [-end <target end index>]
                  [-checkPri[merMatch] <boolean>]
                  [-file <file> [-format <format>]]

Creates a new amplicon in the currently open project.  In the first form,
the non-option argument is used as the name of the new amplicon.  In the
second, a name must be explicitly specified in option form.  If the
‘-orUpdate’ flag is given, an amplicon is only created if it does not
already exist.  If it already exists, the amplicon is merely updated.  The
‘-ofRef’ option can be used to disambiguate amplicons with the same name in
this case.  The remainder of the options are not required but can be used
to set properties of the new amplicon.

    -annotation         The annotation.
    -primer1            The primer 1 sequence.  This must be a nucleotide
                            sequence string conforming to IUPAC
                            nomenclature.  Any ambiguous symbols are
                            considered ‘N’s.
    -primer2            The primer 2 sequence.  This must be a nucleotide
                            sequence string conforming to IUPAC
                            nomenclature.  Any ambiguous symbols are
                            considered ‘N’s.
    -start              The index of the target start position, or a ‘*’
                            to indicate the position should be
                            automatically assigned.
    -end                The index of the target end position, or a ‘*’
                            to indicate the position should be
                            automatically assigned.
    -checkPrimerMatch   Whether the system should check for a match between
                            the reference sequence and the primers in the
                            bases flanking the target region.  This must be
                            ‘true’ or ‘false’, and defaults to ‘true’.

The start and end options indicate the positional range of the amplified
target as measured from the first base of the associated reference
sequence.  In the case that the primer sequences are included in the
reference sequence, the system can automatically assign these positions by
```

```
finding matches of primer1 and the reverse complement of primer2 and
assigning the start and end positions to be just inside these matches.
Either, or both, of the start and end positions may be specified as a '*'
to request this search.  If one position is provided and the other is a
'*', then one position will be constrained as given and the search will
proceed on the other position.  If no such matching pair, or more than one
matching pair can be found, then an error is generated.  N's in either the
reference or primer sequences count as matches, but any match that involves
greater than 50% N's will be rejected.  Any other substitutions,
insertions, or deletions are not permitted.  Using a '*' for either the
start or end implies the checkPrimerMatch option and requires exact matches
of both primers in the reference sequence.  If the primers are not included
in the reference or if the primers contain bases that don't exactly match
the reference, the checkPrimerMatch option should be specified as 'false'
to prevent an error from being generated, and both start and end positions
should be explicitly provided.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.4.2 create mid

```
create mid <new mid name> [-orUpdate]
                          [-seq[uence] <sequence>]
                          [-annot[ation] <annotation>]
                          [-midGroup <midGroup>]
                          [-checkMidGroup <boolean>]
                          [-file <file> [-format <format>]]

create mid -name <new mid name>
           [-orUpdate]
           [-seq[uence] <sequence>]
           [-annot[ation] <annotation>]
           [-midGroup <midGroup>]
           [-checkMidGroup <boolean>]
           [-file <file> [-format <format>]]

Creates a new MID in the currently open project.  In the first form,
the non-option argument is used as the name of the new MID.  In the
second, a name must be explicitly specified in option form.  If the
'-orUpdate' flag is given, an MID is only created if it does not
already exist.  If it already exists, the MID is merely updated.  The
remainder of the options are not required but can be used to set
properties of the new MID.

    -annotation         The annotation.
    -sequence           The MID sequence.  This must be a non-zero length
                            nucleotide sequence string containing only the
                            bases A, C, T and G.
    -midGroup           The MID group of the MID, if it belongs to a
                            group.  This must be a pre-existing group,
                            created using the 'create midGroup' command.
    -checkMidGroup      Whether the system should check for compatibility
                            between the new MID sequence and other pre-
                            existing MID sequences belonging to the same
                            MID group.  This must be 'true' or 'false',
                            and defaults to 'true'.

The name of the MID must be unique within the MID group it belongs to
(or unique within the project if the MID is not assigned to an MID group).

The rules for '-checkMidGroup' compatibility are as follows:

- An MID with an undefined sequence is considered compatible with any MID
  group, under the assumption that its compatibility will eventually be
  assessed when a defined sequence gets assigned to the MID.
- An MID with a defined sequence must have the same length as other
  defined MID sequences within an MID group to be compatible with the
  group. If the new MID sequence is the first defined sequence added to
  the MID group, the required sequence length for subsequent MIDs of the
  group with be the length of that first defined MID sequence.
- An MID with a defined sequence must not be identical (ignoring case)
  with any other defined, pre-existing MID sequence of the same MID group.

If it becomes necessary to edit existing MIDs in a way that temporarily
leaves the MIDs in a group in an inconsistent state (such as changing the
lengths of sequences in an MID group), '-checkMidGroup' should be set to
'false'.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.4.3 create midGroup

```
create midGroup <new midGroup name> [-orUpdate]
                                    [-annot[ation] <annotation>]
                                    [-file <file> [-format <format>]]

create midGroup -name <new midGroup name>
                [-orUpdate]
                [-annot[ation] <annotation>]
                [-file <file> [-format <format>]]

Creates a new MID group in the currently open project.  In the first form,
the non-option argument is used as the name of the new MID group.  In the
second, a name must be explicitly specified in option form.  If the
'-orUpdate' flag is given, an MID group is only created if it does not
already exist.  If it already exists, the MID group is merely updated.
The remainder of the options are not required but can be used to set
properties of the new MID group.

    -annotation     The annotation.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.4.4 create multiplexer

```
create mul[tiplexer] <new multiplexer name>
                     [-orUpdate]
                     [-enc[oding] <encoding>]
                     [-annot[ation] <annotation>]
                     [-file <file> [-format <format>]]

create mul[tiplexer] -name <new multiplexer name>
                     [-orUpdate]
                     [-enc[oding] <encoding>]
                     [-annot[ation] <annotation>]
                     [-file <file> [-format <format>]]

Creates a new multiplexer in the currently open project.  In the first
form, the non-option argument is used as the name of the new multiplexer.
In the second, a name must be explicitly specified in option form.  If the
'-orUpdate' flag is given, a multiplexer is only created if it does not
already exist.  If it already exists, the multiplexer is merely updated.
The remainder of the options are not required but can be used to set
properties of the new multiplexer.

    -annotation         The annotation.
    -encoding           The MID layout type for the multiplexer, where
                            the choices are both, either, primer1, and
                            primer2.

The four '-encoding' types have the following definitions:

    both                 Both primer 1 and primer 2 MIDs are present and
                             necessary to determine the sample for each
                             read.
    either               Both primer 1 and primer 2 MIDs are present, but
                             either one is sufficient to determine the
                             sample.  For a given read, the MID at the
                             5' end, in the read's orientation, is used
                             to determine the sample.
    primer1              MIDs are only present adjacent to primer 1.
    primer2              MIDs are only present adjacent to primer 2.

Although a multiplexer can be initially created without specifying the
'-encoding' type, the '-encoding' type must be set before MIDs and
samples can be associated with the multiplexer.

If '-orUpdate' is used to change the '-encoding' type of a multiplexer,
then all pre-existing sample associations for the multiplexer will be
removed and certain pre-existing associations with MIDs may also be
removed. Specifically, if the '-encoding' type is changed to 'either'
and the numbers of already associated primer 1 and primer 2 MIDs are
not equal, then both sets of MID associations will be removed.  If the
'-encoding' type is changed to 'primer1', then any associated primer 2
MIDs will be dissociated and if the type is changed to 'primer2', any
associated primer 1 MIDs will be dissociated.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.4.5 create project

```
create proj[ect] <path for new project> [-name <new project name>]
                                        [-annot[ation] <annotation>]
                                        [-file <file> [-format <format>]]

create proj[ect] -path <path for new project>
                 [-name <new project name>]
                 [-annot[ation] <annotation>]
                 [-file <file> [-format <format>]]

Creates a new project.  In the first form, the non-option argument is
used as the path at which the new project will be created.  In the
second, a path must be explicitly specified in option form. The remainder
of the options are not required but can be used to set properties of the
new project.

When a new project is created, the previously open project is closed if
necessary, and the new project becomes the open project.

    -name           The name of the project.
    -annotation     The annotation describing the project.


Unlike with the creation of new projects from the gsAmplicon graphical user
interface (GUI), the ‘create project’ command does not initialize new
projects with any default contents.  To initialize a project with the same
default contents as it would have if created by the GUI, the following
command should be run subsequent the ‘create project’ command:

     utility execute %libDir/newProjectInit.ava

Run ‘help general tabularCommands’ for information about the ‘-file’
option.

Run ‘help general filePaths’ for more information about the interpretation
of relative paths when using the ‘-file’ option or specifying the path for
the new project.
```

11.4.4.6 create readGroup

```
create readGroup <new read group name> [-orUpdate]
                                       [-annot[ation] <annotation>]
                                       [-file <file> [-format <format>]]

create readGroup -name <new read group name>
                 [-orUpdate]
                 [-annot[ation] <annotation>]
                 [-file <file> [-format <format>]]

Creates a new read group in the currently open project.  In the first form,
the non-option argument is used as the name of the new read group.  In the
second, a name must be explicitly specified in option form.  If the
‘-orUpdate’ flag is given, a read group is only created if it does not
already exist.  If it already exists, the read group is merely updated.
The remainder of the options are not required but can be used to set
properties of the new read group.

    -annotation     The annotation.

Run ‘help general tabularCommands’ for information about the ‘-file’
option.
```

11.4.4.7 create reference

```
create ref[erence] <new reference name> [-orUpdate]
                                        [-annot[ation] <annotation>]
                                        [-seq[uence] <sequence>]
                                        [-file <file> [-format <format>]]

create ref[erence] -name <new reference name>
                   [-orUpdate]
                   [-annot[ation] <annotation>]
                   [-seq[uence] <sequence>]
                   [-file <file> [-format <format>]]

Creates a new reference sequence in the currently open project.  In the
first form, the non-option argument is used as the name of the new
reference sequence.  In the second, a name must be explicitly specified in
option form.  If the '-orUpdate' flag is given, a reference sequence is
only created if it does not already exist.  If it already exists, the
reference sequence is merely updated.  The remainder of the options are not
required but can be used to set properties of the new reference sequence.

    -annotation     The annotation.
    -sequence       The nucleotide sequence string.  This sequence must use
                        IUPAC nomenclature.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.4.8 create sample

```
create sam[ple] <new sample name> [-orUpdate]
                                  [-annot[ation] <annotation>]
                                  [-file <file> [-format <format>]]

create sam[ple] -name <new sample name>
                [-orUpdate]
                [-annot[ation] <annotation>]
                [-file <file> [-format <format>]]

Creates a new sample in the currently open project.  In the first form, the
non-option argument is used as the name of the new sample.  In the second,
a name must be explicitly specified in option form.  If the '-orUpdate'
flag is given, a sample is only created if it does not already exist.  If
it already exists, the sample is merely updated.  The remainder of the
options are not required but can be used to set properties of the new
sample.

    -annotation     The annotation.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.4.9 create variant

```
create var[iant] <new variant name> [-orUpdate]
                                    [-ofRef <reference name>]
                                    [-annot[ation] <annotation>]
                                    [-ref[erence] <reference name>]
                                    [-pat[tern] <pattern>]
                                    [-stat[us] <status>]
                                    [-checkPat[tern] <boolean>]
                                    [-file <file> [-format <format>]]

create var[iant] -name <new variant name>
                 [-orUpdate]
                 [-ofRef <reference name>]
                 [-ref[erence] <reference name>]
                 [-annot[ation] <annotation>]
                 [-pat[tern] <pattern>]
                 [-stat[us] <status>]
                 [-checkPat[tern] <boolean>]
                 [-file <file> [-format <format>]]

Creates a new variant in the currently open project.  In the first form,
the non-option argument is used as the name of the new variant.  In the
second, a name must be explicitly specified in option form.  If the
‘-orUpdate’ flag is given, a variant is only created if it does not
already exist.  If it already exists, the variant is merely updated.  The
‘-ofRef’ option can be used to disambiguate variants with the same name in
this case.  The remainder of the options are not required but can be used
to set properties of the new variant.

    -annotation     The annotation.
    -reference      The name of the reference sequence to which the
                        variant refers.
    -pattern        The pattern that defines the nature of this variation.
    -status         The putative status.  This can be one of “accepted”,
                        “rejected”, or “putative”
    -checkPattern   Whether the system should check if the variant’s
                        pattern is syntactically correct and consistent
                        with the variant’s reference sequence. The
                        reference sequence must itself be set and have
                        a non-empty nucleotide sequence for this option
                        to take effect.  This value given must be ‘true’
                        or ‘false’, and defaults to ‘true’.

Run ‘help general tabularCommands’ for information about the ‘-file’
option.
```

11.4.5 dissociate

```
dissoc[iate] -sam[ple] <sample name>
             -amp[licon] <amplicon name> [-ofRef <reference sequence name>]
             [-file <file> [-format <format>]]

dissoc[iate] -sam[ple] <sample name>
             -readData <read data name>
             [-file <file> [-format <format>]]

dissoc[iate] -sam[ple] <sample name>
             -amp[licon] <amplicon name> [-ofRef <reference sequence name>]
             -readData <read data name>
             [-file <file> [-format <format>]]

dissoc[iate] -mul[tiplexer] <multiplexer name>
             [-primer1Mid <primer1Mid name>
                 [-ofPrimer1MidGroup <primer1MidGroup name>]]
             [-primer2Mid <primer2Mid name>
                 [-ofPrimer2MidGroup <primer2MidGroup name>]]
             [-file <file> [-format <format>]]

dissoc[iate] -mul[tiplexer] <multiplexer name>
             [-primer1Mid <primer1Mid name>
                 [-ofPrimer1MidGroup <primer1MidGroup name>]]
             [-primer2Mid <primer2Mid name>
                 [-ofPrimer2MidGroup <primer2MidGroup name>]]
             -sam[ple] <sample name>
             [-file <file> [-format <format>]]

dissoc[iate] -mul[tiplexer] <multiplexer name>
             -sam[ple] <sample name>
             [-file <file> [-format <format>]]

dissoc[iate] -mul[tiplexer] <multiplexer name>
             -amp[licon] <amplicon name> [-ofRef <reference sequence name>]
             -readData <readData name>
             [-file <file> [-format <format>]]

dissoc[iate] -mul[tiplexer] <multiplexer name>
             -readData <readData name>
             [-file <file> [-format <format>]]
```

The 'dissociate' command is used to dissociate records in many-to-many relationships. If a general relationship is dissociated, the more specific associations that depend on it automatically will be dissociated as well (*e.g.*, dissociating a sample from a read data will also result in the dissociation of the sample's read data-sample-amplicon associations). General relationships that are included as part of specific relationships are not, however, automatically dissociated (*e.g.*, when dissociating a read data-sample-amplicon relationship, the more general sample-amplicon relationship that it includes will not be dissociated).

In any of the command forms above where '-amplicon' is being specified, the '-ofRef' option can be used to disambiguate amplicons with the same name but which are from different reference sequences.

The '-amplicon' option may be specified as a "*" to allow multiple amplicons to be dissociated with a single command. In the context of a command where '-amplicon' and '-sample' are both specified, the "*" is interpreted to indicate that all of the amplicons of the sample should be dissociated. In the context of a command where '-amplicon', '-multiplexer', and '-readData' are all specified, the "*" is interpreted to indicate that all of the amplicons associated with the multiplexer in the context of that read data should be dissociated. In either case, the '-ofRef' option can still be used to restrict the "*" selection of amplicons to be that subset of amplicons belonging to the indicated reference sequence.

In a similar manner to the '-ofRef' option for amplicons, the '-ofPrimer1MidGroup' and '-ofPrimer2MidGroup' options can be used to disambiguate '-primer1Mid' and '-primer2Mid' specifications, respectively.

The '-primer1Mid' and '-primer2Mid' options may also be specified as a "*". If no '-ofPrimer1MidGroup' or '-ofPrimer2MidGroup' option is supplied, the "*" refers to all the MIDs of the project. If a MID group is specified, the "*" refers to only the MIDs of that MID group.

Explanations of the various command forms are as follows:

Run 'help general tabularCommands' for information about the '-file' option.

```
dissoc[iate] -sam[ple] <sample name>
             -amp[licon] <amplicon name>
             [-ofRef <reference sequence name>]
             [-file <file> [-format <format>]]
```

```
If a sample and an amplicon are specified, they are dissociated.  The
‘-ofRef’ option can be used to disambiguate amplicons with the same name
that refer to different reference sequences.  If a “*” is passed as the
‘-amplicon’ option value, with no ‘-ofRef’ option, all amplicons of the
sample are dissociated.  If both a “*” and the ‘-ofRef’ option are used,
then all amplicons of the sample belonging to the indicated reference
sequence will be dissociated.
```

```
dissoc[iate] -sam[ple] <sample name>
             -readData <read data name>
             [-file <file> [-format <format>]]
```

```
If a sample and a read data are specified, the sample itself, all
amplicons of the sample currently associated with the read data, and the
read data are dissociated.
```

```
dissoc[iate] -sam[ple] <sample name>
             -amp[licon] <amplicon name>
             [-ofRef <reference sequence name>]
             -readData <read data name>
             [-file <file> [-format <format>]]
```

```
If a sample, amplicon, and read data are specified, the sample itself, the
amplicon, and the read data are dissociated.  The ‘-ofRef’ option can be
used to disambiguate amplicons with the same name that refer to different
reference sequences.  If a “*” is passed as the ‘-amplicon’ option value,
all amplicons of the sample are dissociated.  This is identical to using
the invocation form with only the sample and read data specified.
```

```
dissoc[iate] -mul[tiplexer] <multiplexer name>
             [-primer1Mid <primer1Mid name>
                 [-ofPrimer1MidGroup <primer1MidGroup name>]]
             [-primer2Mid <primer2Mid name>
                 [-ofPrimer2MidGroup <primer2MidGroup name>]]
             [-file <file> [-format <format>]]
```

```
If some combination of MIDs and a multiplexer are specified, the MIDs will
be dissociated from the multiplexer.  Any samples associated with those
MIDs via the multiplexer will be dissociated as well.  Note that a
multiplexer may be used on more than one read data, and MID dissociations
will impact sample associations on all of those read data at once.  Also
note, that depending on the pre-existing sample associations and
encoding type of the multiplexer (both, either, primer1, or primer2),
dissociating an MID might impact more than one sample (e.g., if the
multiplexer encoding is ‘both’ and there is a sample1 associated with
primer1Mid mid1 and primer2Mid mid2 and there is a sample2 associated with
primer1Mid mid1 and primer2Mid mid3, dissociating primer1 mid1 from the
multiplexer will cause both samples to be dissociated).
```

```
dissoc[iate] -mul[tiplexer] <multiplexer name>
             [-primer1Mid <primer1Mid name>
                 [-ofPrimer1MidGroup <primer1MidGroup name>]]
             [-primer2Mid <primer2Mid name>
                 [-ofPrimer2MidGroup <primer2MidGroup name>]]
             -sam[ple] <sample name>
             [-file <file> [-format <format>]]
```

```
If some combination of MIDs, a multiplexer, and a sample are specified,
the sample will be dissociated from the specific MID association, but the
MID associations with the multiplexer will be left intact.  The
‘-primer1Mid’ and ‘-primer2Mid’ options are constrained by the encoding
type of the multiplexer.  Since this form of the command is expecting to
dissociate specific sample-MID associations, it must be given an
appropriate combination of MID options that are compatible with the
encoding type.  If the encoding type is ‘both’, ‘-primer1Mid’ and
‘-primer2Mid’ options must both be specified along with the sample.
If the encoding type is ‘either’, it is permissible to provide both MIDs
or just a single MID along with the specified sample.
```

```
dissoc[iate] -mul[tiplexer] <multiplexer name>
             -sam[ple] <sample name>
             [-file <file> [-format <format>]]
```

```
If a multiplexer and a sample are specified, the sample gets dissociated
from all MID combinations that have been used to associate the sample
with the multiplexer.  The pre-existing multiplexer-MID associations are
left intact.
```

```
dissoc[iate] -mul[tiplexer] <multiplexer name>
             -amp[licon] <amplicon name> [-ofRef <reference sequence name>]
             -readData <readData name>
             [-file <file> [-format <format>]]
```

If a multiplexer, amplicon and readData are specified, the amplicon is dissociated from the specific read data-multiplexer context. If the amplicon is simultaneously associated with the same multiplexer on a different read data, that relationship will be left intact. A "*" may be provided with the '-amplicon' option to indicate that all amplicons associated with the read data-multiplexer should be dissociated. The '-ofRef' option can be used, if necessary, to disambiguate among amplicons with the same name or to restrict the "*" set of amplicons to those of the specified reference sequence. Severing the relationship of an amplicon with a read data-multiplexer simultaneously dissociates the amplicon from all of the samples associated with the multiplexer in that read data context. The general sample-amplicon relationships, however, remain intact.

```
dissoc[iate] -mul[tiplexer] <multiplexer name>
             -readData <readData name>
             [-file <file> [-format <format>]]
```

If a multiplexer and a read data are specified, the multiplexer will be dissociated from that specific read data. The internal relationships of the multiplexer such as MID and sample associations remain intact, but any amplicons that were associated with the specific read data-multiplexer will be dissociated. If the multiplexer is simultaneously associated with other specific read data, those associations remain unchanged.

11.4.6 exit

```
exit [<return code>]
```

Exits the command interpreter. By default, 0 is used as the return code for the command interpreter process. If a return code is provided as an argument, it is used instead.

11.4.7 list

```
list <entity type> <other arguments>
```

The 'list' command is used to list information about project entities or the project itself. The type of entity to list is determined by the '<entity type>' argument. The '<other arguments>' are determined by the entity type. For example, to list all amplicons in the currently open project, you can run 'list amplicon'.

The following entities are available for listing. Run 'help list <entity type>' for more detailed information.

amplicon	Lists amplicons in the currently open project.
mid	Lists MIDs in the currently open project.
midGroup	Lists MID groups in the currently open project.
multiplexer	Lists multiplexers in the currently open project.
project	Lists information about the currently open project.
readData	Lists read data in the currently open project.
readGroup	Lists read groups in the currently open project.
reference	Lists reference sequences in the currently open project.
sample	Lists samples in the currently open project.
variant	Lists variants in the currently open project.

11.4.7.1 list amplicon

```
list amp[licon] [-outputFile <file>] [-format <table format>]

Lists all of the amplicons in the currently open project.  The listing is
printed in the form of a table.  The table has columns for the following.

    Name            The name of the amplicon.
    Annotation      The annotation for the amplicon.
    Reference       The reference sequence to which the amplicon refers.
    Primer1         The first primer.
    Primer2         The second primer.
    Start           The index of the start of the target in the reference
                        sequence.
    End             The index of the end of the target in the reference
                        sequence.

If no '-outputFile' option is given, the table is printed in a
tab-delimited format to the standard output of the interpreter.  An output
file of "-" has the same effect.  If an output file is given, the table is
written to that file.  Run 'help general filePaths' for more information
about specifying files.

The '-format' option controls the format of the printed table.  If "tsv", a
tab-delimited format is used.  If "csv", a comma-delimited format is used.
By default, the tab-delimited format is used, unless an output file is
given with a ".csv" extension.
```

11.4.7.2 list mid

```
    list mid [-outputFile <file>] [-format <table format>]

    Lists all of the MIDs in the currently open project.  The listing
    is printed in the form of a table.  The table has columns for the
    following.

        Name            The name of the MID.
        Annotation      The annotation for the MID.
        Sequence        The nucleotide sequence of the MID.
        MidGroup        The MID group to which the MID belongs.

    If no '-outputFile' option is given, the table is printed in a
    tab-delimited format to the standard output of the interpreter.  An output
    file of "-" has the same effect.  If an output file is given, the table is
    written to that file.  Run 'help general filePaths' for more information
    about specifying files.

    The '-format' option controls the format of the printed table.  If "tsv", a
    tab-delimited format is used.  If "csv", a comma-delimited format is used.
    By default, the tab-delimited format is used, unless an output file is
    given with a ".csv" extension.
```

11.4.7.3 list midGroup

```
list midGroup [-outputFile <file>] [-format <table format>]

Lists all of the MID groups in the currently open project.  The listing is
printed in the form of a table.  The table has columns for the following.

    Name            The name of the MID group.
    Annotation      The annotation for the MID group.

If no '-outputFile' option is given, the table is printed in a
tab-delimited format to the standard output of the interpreter.  An output
file of "-" has the same effect.  If an output file is given, the table is
written to that file.  Run 'help general filePaths' for more information
about specifying files.

The '-format' option controls the format of the printed table.  If "tsv", a
tab-delimited format is used.  If "csv", a comma-delimited format is used.
By default, the tab-delimited format is used, unless an output file is
given with a ".csv" extension.
```

11.4.7.4 list multiplexer

```
list mul[tiplexer] [-outputFile <file>] [-format <table format>]

Lists all of the multiplexers in the currently open project.  The listing
is printed in the form of a table.  The table has columns for the
following.

    Name            The name of the multiplexer.
    Annotation      The annotation for the multiplexer.
    Encoding        The encoding type of the multiplexer (both, either,
                        primer1, or primer2).

If no ‘-outputFile’ option is given, the table is printed in a
tab-delimited format to the standard output of the interpreter.  An output
file of “-” has the same effect.  If an output file is given, the table is
written to that file.  Run ‘help general filePaths’ for more information
about specifying files.

The ‘-format’ option controls the format of the printed table.  If “tsv”, a
tab-delimited format is used.  If “csv”, a comma-delimited format is used.
By default, the tab-delimited format is used, unless an output file is
given with a “.csv” extension.
```

11.4.7.5 list project

```
list proj[ect] [-outputFile <file>] [-format <table format>]

Lists data about the currently open project.  The listing is
printed in the form of a table.  The table has columns for the following.

    Path            The directory path to the project.
    Name            The name for the project.
    Annotation      The annotation for the project.

If no ‘-outputFile’ option is given, the table is printed in a
tab-delimited format to the standard output of the interpreter.  An output
file of “-” has the same effect.  If an output file is given, the table is
written to that file.  Run ‘help general filePaths’ for more information
about specifying files.

The ‘-format’ option controls the format of the printed table.  If “tsv”, a
tab-delimited format is used.  If “csv”, a comma-delimited format is used.
By default, the tab-delimited format is used, unless an output file is
given with a “.csv” extension.
```

11.4.7.6 list readData

```
list readData [-outputFile <file>] [-format <table format>]

Lists all of the read data in the currently open project.  The listing is
printed in the form of a table.  The table has columns for the following.

    Name            The name of the read data.
    Annotation      The annotation for the read data.
    ReadGroup       The read group to which the read data belongs.
    SymLink         Whether the read data is symbolically linked into the
                        project.
    Active          Whether the read data is active in the project.
    SffDir          The SFF directory from which the read data was
                        imported.
    SffName         The name of the SFF file of the read data.

If no ‘-outputFile’ option is given, the table is printed in a
tab-delimited format to the standard output of the interpreter.  An output
file of “-” has the same effect.  If an output file is given, the table is
written to that file.  Run ‘help general filePaths’ for more information
about specifying files.

The ‘-format’ option controls the format of the printed table.  If “tsv”, a
tab-delimited format is used.  If “csv”, a comma-delimited format is used.
By default, the tab-delimited format is used, unless an output file is
given with a “.csv” extension.
```

11.4.7.7 list readGroup

```
list readGroup [-outputFile <file>] [-format <table format>]

Lists all of the read groups in the currently open project.  The listing is
printed in the form of a table.  The table has columns for the following.

    Name            The name of the read group.
    Annotation      The annotation for the read group.

If no ‘-outputFile’ option is given, the table is printed in a
tab-delimited format to the standard output of the interpreter.  An output
file of “-” has the same effect.  If an output file is given, the table is
written to that file.  Run ‘help general filePaths’ for more information
about specifying files.

The ‘-format’ option controls the format of the printed table.  If “tsv”, a
tab-delimited format is used.  If “csv”, a comma-delimited format is used.
By default, the tab-delimited format is used, unless an output file is
given with a “.csv” extension.
```

11.4.7.8 list reference

```
    list ref[erence] [-outputFile <file>] [-format <table format>]

    Lists all of the reference sequences in the currently open project.  The
    listing is printed in the form of a table.  The table has columns for the
    following.

        Name            The name of the reference.
        Annotation      The annotation for the reference.
        Sequence        The nucleotide sequence of the reference.

    If no ‘-outputFile’ option is given, the table is printed in a
    tab-delimited format to the standard output of the interpreter.  An output
    file of “-” has the same effect.  If an output file is given, the table is
    written to that file.  Run ‘help general filePaths’ for more information
    about specifying files.

    The ‘-format’ option controls the format of the printed table.  If “tsv”, a
    tab-delimited format is used.  If “csv”, a comma-delimited format is used.
    By default, the tab-delimited format is used, unless an output file is
    given with a “.csv” extension.
```

11.4.7.9 list sample

```
    list sam[ple] [-outputFile <file>] [-format <table format>]

    Lists all of the samples in the currently open project.  The listing is
    printed in the form of a table.  The table has columns for the following.

        Name            The name of the sample.
        Annotation      The annotation for the sample.

    If no ‘-outputFile’ option is given, the table is printed in a
    tab-delimited format to the standard output of the interpreter.  An output
    file of “-” has the same effect.  If an output file is given, the table is
    written to that file.  Run ‘help general filePaths’ for more information
    about specifying files.

    The ‘-format’ option controls the format of the printed table.  If “tsv”, a
    tab-delimited format is used.  If “csv”, a comma-delimited format is used.
    By default, the tab-delimited format is used, unless an output file is
    given with a “.csv” extension.
```

11.4.7.10 list variant

```
list var[iant] [-outputFile <file>] [-format <table format>]

Lists all of the variants in the currently open project.  The listing is
printed in the form of a table.  The table has columns for the following.

    Name            The name of the variant.
    Annotation      The annotation for the variant.
    Reference       The reference sequence to which the variant refers.

If no ‘-outputFile’ option is given, the table is printed in a
tab-delimited format to the standard output of the interpreter.  An output
file of “-” has the same effect.  If an output file is given, the table is
written to that file.  Run ‘help general filePaths’ for more information
about specifying files.

The ‘-format’ option controls the format of the printed table.  If “tsv”, a
tab-delimited format is used.  If “csv”, a comma-delimited format is used.
By default, the tab-delimited format is used, unless an output file is
given with a “.csv” extension.
```

11.4.8 load

```
load -readGroup <read group name>
     -sffDir <SFF directory>
     -sffName <SFF file name>
     [-symLink <boolean>]
     [-alias <alias prefix for command interpreter>]
     [-file <file> [-format <format>]]

load -readGroup <read group name>
     -analysisDir <analysis directory>
     -sffName <SFF file name>
     [-symLink <boolean>]
     [-alias <alias prefix for command interpreter>]
     [-file <file> [-format <format>]]

load -readGroup <read group name>
     -sffDir <SFF directory>
     -regions <comma-separated region list>
     [-symLink <boolean>]
     [-filePrefix <SFF file prefix>]
     [-alias <alias prefix for command interpreter>]
     [-file <file> [-format <format>]]

load -readGroup <read group name>
     -analysisDir <analysis directory>
     -regions <comma-separated region list>
     [-symLink <boolean>]
     [-filePrefix <SFF file prefix>]
     [-alias <alias prefix for command interpreter>]
     [-file <file> [-format <format>]]

The ‘load’ command is used load read data into the currently open project.
The different options combinations for running the ‘load’ command provide
different ways of specifying what read data to load.

For all forms of invocation, the read group into which the read data will
be loaded must be provided using the ‘-readGroup’ option.

The ‘-symLink’ option defaults to false, but may be specified as true.
When specified as true, the read data files are not actually copied into
the area of the disk that stores the Amplicon Project: a symbolic link
to the data is created instead.

The location of the read data files can be specified with either the
‘-sffDir’ or ‘-analysisDir’ options.  Use the ‘-sffDir’ option to specify a
directory that directly contains read data files (.sff files).  Use the
‘-analysisDir’ option to specify an analysis directory.

In addition to the location of the read data files, the specific read data
to load must also be specified with the ‘-sffName’ or ‘-regions’ options.
Use the ‘-sffName’ option to specify the name of the SFF file to load.  Use
the ‘-regions’ option to specify the regions to load.  Regions must be
specified in a comma separated list with no intervening spaces.  For
example, ‘-regions 1,2,4’, specifies that regions 1, 2, and 4 should be
loaded.

If the ‘-regions’ option is used to specify the read data files to load,
a ‘-filePrefix’ option may be provided to restrict the loading to only
```

```
    those regions whose SFF file names begin with a certain prefix.  For
    example, in a given SFF directory there may be two region 1 files,
    TEST01.sff and REAL01.sff.  If you specify '-regions 1', both of these
    files will be loaded.  However, if you specify '-regions 1 -filePrefix
    REAL', only the later file will be loaded.

    An alias may be provided that allows loaded read data to be referenced by
    subsequent commands.  For example, if we run
    'load -readGroup MyGroup -sffDir some/path/sff -regions 1,2 -alias read',
    we can subsequently refer to the imported region read data as "read01" and
    "read02".  For example, we can run:
         'assoc -readData read01 -sample sample1 -amplicon amplicon1'
    (assuming sample1 and amplicon1 exist).  The alias is constructed by taking
    the value passed to the '-alias' option and appending two digits specifying
    the region.  This option facilitates the creation of scripts that load from
    analysis directories, wherein the regions of interest are known in
    advance, but the actual SFF file names are not known (since they are
    automatically given names by the pipeline software).

    Here are some examples of valid 'load' invocations.

    load -readGroup Group1 -sffDir /data/sff -sffName TEST01.sff

        This will load the read data in /data/sff/TEST01.sff into the read
        group named "Group1" of the currently open project.

    load -readGroup Group1 -analysisDir /data/analysis1 \
                           -regions 1,2,4 -alias Read

        This will load the read data of regions 1, 2, and 4 inside the analysis
        directory /data/analysis1 into the read group named "Group1" of the
        currently open project.  Subsequent commands will be able to refer to
        the read data as "Read01", "Read02", and "Read04".

    load -readGroup Group1 -analysisDir /data/analysis1 \
                           -regions 2 -filePrefix TEST -alias Read

        This will load the read data of region 2 inside the analysis directory
        /data/analysis1 into the read group named "Group1" of the currently
        open project.  Only SFF files with prefix "TEST" in the analysis will
        be considered.

    Run 'help general tabularCommands' for information about the '-file'
    option.

    Run 'help general filePaths' for more information about the interpretation
    of relative paths when using the '-file', '-analysisDir' or '-sffDir'
    options.
```

11.4.9 open

```
open <project path> [-control <control mode>]

Opens a project at a given path.  When a project is opened, the previously
open project is closed if necessary.

    -control        The control mode.  This can be one of "preempt" or
                        "readOnly".  By default, this command attempts to
                        acquire control of the project, which is required
                        to modify and run computations on the project.  If
                        another application is already in control, the
                        attempt fails, and an error is reported.  If the
                        control mode is set to "preempt", this command
                        will preempt the control of the other application
                        and take control for itself.  If the control
                        mode is set to "readOnly", then control is not taken,
                        and attempts to save modifications to this project will
                        fail.

 Note: If a previous instance of the command line or graphical user
 interface had the project open and was prematurely terminated, it may
 erroneously appear to the system that the project is currently under the
 control of another program instance.   In this case, in order to obtain
 control over the project, the "-control preempt" option will need to
 be used.

Run 'help general filePaths' for more information about the interpretation
of relative paths if used when specifying the project path.
```

11.4.10 remove

```
remove <entity type> <other arguments>

The ‘remove’ command is used to remove entities.  The type of entity to
remove is determined by the ‘<entity type>’ argument.  The ‘<other
arguments>’ are determined by the entity type.  For project records, the
‘<other arguments>’ is generally the name of the record to remove.

The following entities are available for removing.  Run
‘help remove <entity type>’ for more detailed information.

amplicon        Removes an amplicon from the currently open project.
mid             Removes an MID from the currently open project.
midGroup        Removes an MID group from the currently open project.
multiplexer     Removes a multiplexer from the currently open project.
readData        Removes a read data from the currently open project.
readGroup       Removes a read group from the currently open project.
reference       Removes a reference sequence from the currently open
                    project.
sample          Removes a sample from the currently open project.
variant         Removes a variant from the currently open project.
```

11.4.10.1 remove amplicon

```
remove amp[licon] <amplicon name> [-ofRef <reference sequence name>]
                                  [-file <file> [-format <format>]]

remove amp[licon] -name <amplicon name>
                  [-ofRef <reference sequence name>]
                  [-file <file> [-format <format>]]

Removes an amplicon.  In the first form, the non-option argument is used as
the name of the amplicon to remove.  In the second, a name must be
explicitly specified in option form.

Amplicons are allowed to have duplicate names as long as the reference
sequences to which they refer are distinct.  The ‘-ofRef’ argument can be
used to refer to such amplicons.  For example, if we have two amplicons
named “MyAmp”, but one of them refers to “ReferenceSequence1” and the other
to “ReferenceSequence2”, we can use the ‘-ofRef’ option to distinguish
them.  We can run ‘remove amplicon “MyAmp” -ofRef “ReferenceSequence1”’ to
remove the former amplicon.

If the amplicon name is given as the character ‘*’ then all amplicons will
be removed.  If the ‘-ofRef’ option is also supplied, then all the
amplicons of just that reference sequence will be removed.

Run ‘help general tabularCommands’ for information about the ‘-file’
option.
```

11.4.10.2 remove mid

```
remove mid <mid name> [-ofMidGroup <midGroup>]
                      [-file <file> [-format <format>]]

remove mid -name <mid name>
           [-ofMidGroup <midGroup>]
           [-file <file> [-format <format>]]

Removes an MID.  In the first form, the non-option argument is used as
the name of the MID to remove.  In the second, a name must be
explicitly specified in option form.

MIDs are allowed to have duplicate names as long as they belong to
distinct MID groups.  The ‘-ofMidGroup’ argument can be used to refer to
such MIDs.  For example, if we have two MIDs named “MyMID”, but one of
them is a member of MID group “MID_Group1” and the other is a member of
MID group “MID_Group2”, we can use the ‘-ofMidGroup’ option to distinguish
them.  We can run ‘remove mid “MyMID” -ofMidGroup “MID_Group1”’ to remove
the former MID.

If the MID name is given as the character ‘*’ then all MIDs will
be removed.  If the ‘-ofMidGroup’ option is also supplied, then all the
MIDs of just that MID group will be removed.

Removing MIDs also results in the removal of any associations in which
they are participants.

Run ‘help general tabularCommands’ for information about the ‘-file’
option.
```

11.4.10.3 remove midGroup

```
remove midGroup <read group name> [-file <file> [-format <format>]]
remove midGroup -name <read group name> [-file <file> [-format <format>]]

Removes an MID group.  In the first form, the non-option argument is used
as the name of the read group to remove.  In the second, a name must be
explicitly specified in option form.

If an MID group is removed, then all the MIDs of that group are
also removed.

If the MID group name is given as the character '*' then all MID groups
will be removed.  This would remove all the MIDs that belong to MID groups
from the project at the same time, but it would leave behind any MIDs that
do not have an MID group assignment.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.10.4 remove multiplexer

```
remove mul[tiplexer] <multiplexer name> [-file <file> [-format <format>]]
remove mul[tiplexer] -name <multiplexer name>
                     [-file <file> [-format <format>]]

Removes a multiplexer.  In the first form, the non-option argument
is used as the name of the multiplexer to remove.  In the second, a
name must be explicitly specified in option form.

If a multiplexer is removed, then all the associations that include that
multiplexer (such as muliplexer-readData and multiplexer-MID associations)
are removed at the same time.

If the multiplexer name is given as the character '*' then all the
multiplexers will be removed along with the associations in which they
participate.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.10.5 remove readData

```
remove readData <read data name> [-file <file> [-format <format>]]
remove readData -name <read data name> [-file <file> [-format <format>]]

Removes a read data.  In the first form, the non-option argument is used as
the name of the read data to remove.  In the second, a name must be
explicitly specified in option form.

If the read data name is given as the character '*' then all read
data will be removed.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.10.6 remove readGroup

```
remove readGroup <read group name> [-file <file> [-format <format>]]
remove readGroup -name <read group name> [-file <file> [-format <format>]]

Removes a read group.  In the first form, the non-option argument is used
as the name of the read group to remove.  In the second, a name must be
explicitly specified in option form.

If a read group is removed, then all the read data of that group are
also removed.

If the read group name is given as the character '*' then all read groups
will be removed.  This would effectively remove all the read data from
the project at the same time.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.10.7 remove reference

```
remove ref[erence] <reference name> [-file <file> [-format <format>]]
remove ref[erence] -name <reference name> [-file <file> [-format <format>]]

Removes a reference sequence.  In the first form, the non-option argument
is used as the name of the reference sequence to remove.  In the second, a
name must be explicitly specified in option form.

If a reference sequence is removed, then all the amplicons and variants
associated with that reference sequence are removed at the same time.

If the reference sequence name is given as the character '*' then all
the reference sequences will be removed.  This would effectively remove
all the amplicons and variants at the same time.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.10.8 remove sample

```
    remove sam[ple] <sample name> [-file <file> [-format <format>]]
remove sam[ple] -name <sample name> [-file <file> [-format <format>]]

Removes a sample.  In the first form, the non-option argument is used as
the name of the sample to remove.  In the second, a name must be explicitly
specified in option form.

If the sample name is given as the character '*' then all samples will be
removed.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.10.9 remove variant

```
remove var[iant] <variant name> [-ofRef <reference sequence name>]
                                [-file <file> [-format <format>]]

remove var[iant] -name <variant name>
                 [-ofRef <reference sequence name>]
                 [-file <file> [-format <format>]]

Removes a variant.  In the first form, the non-option argument is used as
the name of the variant to remove.  In the second, a name must be
explicitly specified in option form.

Variants are allowed to have duplicate names as long as the reference
sequences to which they refer are distinct.  The '-ofRef' argument can be
used to refer to such variants.  For example, if we have two variants named
"MyVar", but one of them refers to "ReferenceSequence1" and the other to
"ReferenceSequence2", we can use the '-ofRef' option to distinguish them.
We can run 'remove amplicon "MyVar" -ofRef "ReferenceSequence1"' to remove
the former variant.

If the variant name is given as the character '*' then all variants will
be removed.  If the '-ofRef' option is also supplied, then all the variants
of just that reference sequence will be removed.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.11 rename

```
rename <entity type> <other arguments>

The 'rename' command is used to rename entities.  The type of entity to
rename is determined by the '<entity type>' argument.  The '<other
arguments>' are determined by the entity type.  For project records, the
'<other arguments>' are generally the name of the record to rename followed
by the new name for the record.  For example, running 'rename amplicon
"Amp1" "Amp2"' will rename the amplicon named "Amp1" to "Amp2".

The following entities are available for renaming.  Run
'help rename <entity type>' for more detailed information.

amplicon        Renames an amplicon in the currently open project.
mid             Renames an MID in the currently open project.
midGroup        Renames an MID group in the currently open project.
multiplexer     Renames a multiplexer  in the currently open project.
project         Renames the currently open project.
readData        Renames a read data in the currently open project.
readGroup       Renames a read group in the currently open project.
reference       Renames a reference sequence in the currently open project.
sample          Renames a sample in the currently open project.
variant         Renames a variant in the currently open project.
```

11.4.11.1 rename amplicon

```
rename amp[licon] <name> <new name> [-ofRef <reference sequence name>]
                                    [-file <file> [-format <format>]]

rename amp[licon] -name <name>
                  -newName <new name>
                  [-ofRef <reference sequence name>]
                  [-file <file> [-format <format>]]

Renames an amplicon.  Amplicons are allowed to have duplicate names as long
as the reference sequences to which they refer are distinct.  The '-ofRef'
argument can be used to refer to such amplicons.  For example, if we have
two amplicons named "MyAmp", but one of them refers to "ReferenceSequence1"
and the other to "ReferenceSequence2", we can use the '-ofRef' option to
distinguish them.  We can run 'rename amplicon "MyAmp" "MyAmp2" -ofRef
"ReferenceSequence1"' to rename the former amplicon.

Instead of using arguments to specify the name and new name, the '-name'
and '-newName' options can be used.  This is useful when running this as a
tabular command.  Run 'help general tabularCommands' for information about
tabular commands and the '-file' option.
```

11.4.11.2 rename mid

```
rename mid <name> <new name> [-ofMidGroup <midGroup>]
                             [-file <file> [-format <format>]]

rename mid -name <name>
           -newName <new name>
           [-ofMidGroup <midGroup>]
           [-file <file> [-format <format>]]

Renames an MID. MIDs are allowed to have duplicate names as long as they
belong to distinct MID groups.  The '-ofMidGroup' argument can be used to
refer to such MIDs.  For example, if we have two MIDs named "MyMID", but
one of them is a member of MID group "MID_Group1" and the other is a
member of MID group "MID_Group2", we can use the '-ofMidGroup' option to
distinguish them.  We can run:
       rename mid "MyMID" "MyMid2" -ofMidGroup "MID_Group1"
to update the former MID.

Instead of using arguments to specify the name and new name, the '-name'
and '-newName' options can be used.  This is useful when running this as a
tabular command.  Run 'help general tabularCommands' for information about
tabular commands and the '-file' option.
```

11.4.11.3 rename midGroup

```
rename midGroup <name> <new name> [-file <file> [-format <format>]]

rename midGroup -name <name>
                -newName <new name>
                [-file <file> [-format <format>]]

Renames an MID group.

Instead of using arguments to specify the name and new name, the ‘-name’
and ‘-newName’ options can be used.  This is useful when running this as a
tabular command.  Run ‘help general tabularCommands’ for information about
tabular commands and the ‘-file’ option.
```

11.4.11.4 rename multiplexer

```
rename multiplexer <name> <new name> [-file <file> [-format <format>]]

rename multiplexer -name <name>
                   -newName <new name>
                   [-file <file> [-format <format>]]

Renames an multiplexer.

Instead of using arguments to specify the name and new name, the ‘-name’
and ‘-newName’ options can be used.  This is useful when running this as a
tabular command.  Run ‘help general tabularCommands’ for information about
tabular commands and the ‘-file’ option.
```

11.4.11.5 rename project

```
rename project <new name>

Renames the currently open project.
```

11.4.11.6 rename readData

```
rename readData <name> <new name> [-file <file> [-format <format>]]

rename readData -name <name>
                -newName <new name>
                [-file <file> [-format <format>]]

Renames a read data.

Instead of using arguments to specify the name and new name, the ‘-name’
and ‘-newName’ options can be used.  This is useful when running this as a
tabular command.  Run ‘help general tabularCommands’ for information about
tabular commands and the ‘-file’ option.
```

11.4.11.7 rename readGroup

```
rename readGroup <name> <new name> [-file <file> [-format <format>]]

rename readGroup -name <name>
                 -newName <new name>
                 [-file <file> [-format <format>]]

Renames a read group.

Instead of using arguments to specify the name and new name, the ‘-name’
and ‘-newName’ options can be used.  This is useful when running this as a
tabular command.  Run ‘help general tabularCommands’ for information about
tabular commands and the ‘-file’ option.
```

11.4.11.8 rename reference

```
rename ref[erence] <name> <new name> [-file <file> [-format <format>]]

rename ref[erence] -name <name>
                   -newName <new name>
                   [-file <file> [-format <format>]]

Renames a reference sequence.

Instead of using arguments to specify the name and new name, the ‘-name’
and ‘-newName’ options can be used.  This is useful when running this as a
tabular command.  Run ‘help general tabularCommands’ for information about
tabular commands and the ‘-file’ option.
```

11.4.11.9 rename sample

```
rename sam[ple] <name> <new name> [-file <file> [-format <format>]]

rename sam[ple] -name <name>
                -newName <new name>
                [-file <file> [-format <format>]]

Renames a sample.

Instead of using arguments to specify the name and new name, the ‘-name’
and ‘-newName’ options can be used.  This is useful when running this as a
tabular command.  Run ‘help general tabularCommands’ for information about
tabular commands and the ‘-file’ option.
```

11.4.11.10 rename variant

```
rename var[iant] <name> <new name> [-ofRef <reference sequence name>]
                                   [-file <file> [-format <format>]]

rename var[iant] -name <name>
                 -newName <new name>
                 [-ofRef <reference sequence name>]
                 [-file <file> [-format <format>]]

Renames an variant.  Variants are allowed to have duplicate names as long
as the reference sequences to which they refer are distinct.  The ‘-ofRef’
argument can be used to refer to such variants.  For example, if we have
two variants named “MyVar”, but one of them refers to “ReferenceSequence1”
and the other to “ReferenceSequence2”, we can use the ‘-ofRef’ option to
distinguish them.  We can run ‘update variant “MyVar” -ofRef
“ReferenceSequence1”’ to update the former variant.

Instead of using arguments to specify the name and new name, the ‘-name’
and ‘-newName’ options can be used.  This is useful when running this as a
tabular command.  Run ‘help general tabularCommands’ for information about
tabular commands and the ‘-file’ option.
```

11.4.12 report

```
report <report type> <other arguments>

The 'report' command is used to generate reports about the currently open
project.  The type of report is determined by the '<report type>' argument.
The '<other arguments>' are determined by the report type.

The following report types are available.  Run
'help report <report type>' for more detailed information.

variantHits     The variant hits in the currently open project.
```

11.4.12.1 report variantHits

```
report variantHits [-outputFile <file>] [-format <table format>]

Reports variant hits.  Variant hits are reported in the form of a table.
The table has columns for the following.

    Reference Name
    Variant Name
    Variant Status
    Variant Pattern
    Sample Name
    Forward Hits
    Forward Denom
    Reverse Hits
    Reverse Denom
    Read Type

Data are provided for a Variant of a given Reference Sequence if there
are reads of a Sample that span the region of variation as described
by the Variant Pattern.  The number of forward and reverse reads that
span the region are reported in the Forward Denom and Reverse Denom
columns, respectively.  The number of these reads that have the variation
are given in the Forward Hits and Reverse Hits columns.  The Hit / Denom
ratio provides an estimate of the Variant frequency in the Sample.
Two rows of data are given for each Variant based on the Read Type,
which is either Consensus or Individual.

If no '-outputFile' option is given, the table is printed in a
tab-delimited format to the standard output of the interpreter.  An output
file of "-" has the same effect.  If an output file is given, the table is
written to that file.  Run 'help general filePaths' for more information
about specifying files.

The '-format' option controls the format of the printed table.  If "tsv", a
tab-delimited format is used.  If "csv", a comma-delimited format is used.
By default, the tab-delimited format is used, unless an output file is
given with a ".csv" extension.

Here are some examples.

report variantHits

    Reports the variant hits table to the standard output of the command
    interpreter in a tab-delimited format.

report variantHits -outputFile /reports/hits.csv

    Reports the variant hits table to the /reports/hits.csv file in a
    comma-delimited format.

report variantHits -outputFile -

    Reports the variant hits table to the standard output of the command
    interpreter in a tab-delimited format.
```

11.4.13 save

```
save

This command takes no arguments.  It saves the currently open project,
committing any modifications made since opening the project or since the
last save.  If no project is currently open, an error is reported.
```

11.4.14 set

```
set <parameter name> <value>

Sets the value of a parameter to a given value.  The following parameter
are available.  Run 'help set <parameter name>' for more detailed
information.

verbose     Sets the verbose mode.
onErrors    Sets the behavior of the interpreter when errors are
                encountered.
currDir     Sets the current directory.
```

11.4.14.1 set verbose

```
set verbose <"true" or "false">

Sets the value of the 'verbose' parameter.  If the 'verbose' parameter is
set to 'true', extra information is provided about the commands that are
executed.  This may be useful to help debug scripts.
```

11.4.14.2 set onErrors

```
set onErrors <"stop" or "continue">

Sets the value of the 'onErrors' parameter.  If 'onErrors' is set to
'stop', the command interpreter will stop the current running script
if an error is encountered.  If 'onErrors' is set to 'continue', the
command interpreter will abort the command that caused the error but
will continue running and executing subsequent commands.

In the case that the interpreter stops due to an error, if it is running
a "top level" script (i.e., one that was not started from another script
with 'utility execute'), then the command interpreter will exit.  If the
running script was started from another script using the 'utility execute'
command, then control will be returned to the calling script and the
'utility execute' command in the calling script will be treated as if it
encountered an error.  The behavior in the calling script then depends on
how the 'onErrors' parameter is set in its environment.  If set to
'continue', the calling script will continue running the commands
subsequent to the 'utility execute'; otherwise, it will stop the calling
script:  this same rule is applied again in the case that the calling
script was itself invoked via a 'utility execute' from another script.

Run 'help utility execute' for information about how one script can execute
another script.
```

11.4.14.3 set currDir

```
set currDir <path>

Sets the current directory used to resolve relative file paths.  If the
indicated path does not exist or is not a directory, a warning is shown.

Run 'help general filePaths' for more information about file paths.
```

11.4.15 show

```
show <show command> <other arguments>

The ‘show’ command is used to show various information about the
interpreter.

The following show commands are available.  Run ‘help show <show command>’
for more detailed information.

environment             Shows the environment that defines the behavior of
                            the interpreter.
```

11.4.15.1 show environment

```
show env[ironment]

Shows the current environment in which commands are being run.  Here is
some example output:

libDir=/opt/454/apps/amplicons/config/lib
currDir=/home/me/data
homeDir=/home/me
verbose=false
onErrors=stop
project=MyProject (/home/me/data/MyProject)

The first three lines show the values of the path variables that may be
used in resolving relative file paths.  Run ‘help general filePaths’ for
more information about file paths.

The next line shows whether verbose mode is turned on.
Run ‘help set verbose’ for more information about this value.

The next line shows the behavior when errors are encountered.
Run ‘help set onErrors’ for more information about this value.

The next line shows the currently open project, indicating the project
name and location.  This is the project that will be affected by any
project-related commands.
```

11.4.16 update

```
update <entity type> <other arguments>

The ‘update’ command is used to update properties of entities.  For
example, you can update the annotation of an amplicon by running,
‘update amplicon “My Amplicon” -annotation “New annotation”’.

The following entities are available for updating.  Run
‘help update <entity type>’ for more detailed information.

amplicon        Updates an amplicon in the currently open project.
mid             Updates an MID in the currently open project.
midGroup        Updates an MID group in the currently open project.
multiplexer     Updates a multiplexer in the currently open project.
project         Updates the currently open project.
readData        Updates a read data in the currently open project.
readGroup       Updates a read group in the currently open project.
reference       Updates a reference sequence in the currently open project.
sample          Updates a sample in the currently open project.
variant         Updates a variant in the currently open project.
```

11.4.16.1 update amplicon

```
update amp[licon] <new amplicon name> [-ofRef <reference sequence name>]
                                      [-annot[ation] <annotation>]
                                      [-primer1 <primer 1 sequence>]
                                      [-primer2 <primer 2 sequence>]
                                      [-start <target start index>]
                                      [-end <target end index>]
                                      [-checkPri[merMatch] <boolean>]
                                      [-file <file> [-format <format>]]

update amp[licon] -name <new amplicon name>
                  [-ofRef <reference sequence name>]
                  [-annot[ation] <annotation>]
                  [-primer1 <primer 1 sequence>]
                  [-primer2 <primer 2 sequence>]
                  [-start <target start index>]
                  [-end <target end index>]
                  [-checkPri[merMatch] <boolean>]
                  [-file <file> [-format <format>]]
```

Updates an amplicon in the currently open project. In the first form, the non-option argument is used as the name of the amplicon to update. In the second, a name must be explicitly specified in option form. Amplicons are allowed to have duplicate names as long as the reference sequences to which they refer are distinct. The '-ofRef' argument can be used to refer to such amplicons. For example, if we have two amplicons named "MyAmp", but one of them refers to "ReferenceSequence1" and the other to "ReferenceSequence2", we can use the '-ofRef' option to distinguish them. We can run 'update amplicon "MyAmp" -ofRef "ReferenceSequence1"' to update the former amplicon.

The remainder of the options are not required but are used to set properties of the amplicon.

```
    -ofRef              The name of the reference sequence to which the
                            amplicon refers to help disambiguate amplicons
                            with the same name.
    -annotation         The annotation.
    -primer1            The primer 1 sequence.  This must be a nucleotide
                            sequence string conforming to IUPAC
                            nomenclature.  Any ambiguous symbols are
                            considered 'N's.
    -primer2            The primer 2 sequence.  This must be a nucleotide
                            sequence string conforming to IUPAC
                            nomenclature.  Any ambiguous symbols are
                            considered 'N's.
    -start              The index of the target start position, or a '*'
                            to indicate the position should be
                            automatically assigned.
    -end                The index of the target end position, or a '*'
                            to indicate the position should be
                            automatically assigned.
    -checkPrimerMatch   Whether the system should check for a match between
                            the reference sequence and the primers in the
                            bases flanking the target region.  This must be
                            'true' or 'false', and defaults to 'true'.
```

The start and end options indicate the positional range of the amplified target as measured from the first base of the associated reference sequence. In the case that the primer sequences are included in the reference sequence, the system can automatically assign these positions by finding matches of primer1 and the reverse complement of primer2 and assigning the start and end positions to be just inside these matches. Either, or both, of the start and end positions may be specified as a '*' to request this search. If one position is provided and the other is a '*', then one position will be constrained as given and the search will proceed on the other position. If no such matching pair, or more than one matching pair can be found, then an error is generated. N's in either the reference or primer sequences count as matches, but any match that involves greater than 50% N's will be rejected. Any other substitutions, insertions, or deletions are not permitted. Using a '*' for either the start or end implies the checkPrimerMatch option and requires exact matches of both primers in the reference sequence. If the primers are not included in the reference or if the primers contain bases that don't exactly match the reference, the checkPrimerMatch option should be specified as 'false' to prevent an error from being generated, and both start and end positions should be explicitly provided.

Run 'help general tabularCommands' for information about the '-file' option.

11.4.16.2 update mid

```
update mid <mid name> [-ofMidGroup <midGroup>]
                      [-seq[uence] <sequence>]
                      [-annot[ation] <annotation>]
                      [-midGroup <midGroup>]
                      [-checkMidGroup <boolean>]
                      [-file <file> [-format <format>]]

update mid -name <mid name>
           [-ofMidGroup <midGroup>]
           [-seq[uence] <sequence>]
           [-annot[ation] <annotation>]
           [-midGroup <midGroup>]
           [-checkMidGroup <boolean>]
           [-file <file> [-format <format>]]

Updates an MID in the currently open project.  In the first form,
the non-option argument is used as the name of the MID to update.  In the
second, a name must be explicitly specified in option form.  MIDs are
allowed to have duplicate names as long as they belong to distinct MID
groups.  The '-ofMidGroup' argument can be used to refer to such MIDs.
For example, if we have two MIDs named "MyMID", but one of them is a member
of MID group "MID_Group1" and the other is a member of MID group
"MID_Group2", we can use the '-ofMidGroup' option to distinguish them.  We
can run 'update mid "MyMID" -ofMidGroup "MID_Group1"' to update the former
MID.

The remainder of the options are not required but can be used to set
properties of the MID.

    -annotation         The annotation.
    -sequence           The MID sequence.  This must be a non-zero length
                            nucleotide sequence string containing only the
                            bases A, C, T and G.
    -midGroup           The MID group of the MID if it belongs to a group.
                            This must be a pre-existing group such as one
                            created using the 'create midGroup' command.
    -checkMidGroup      Whether the system should check for compatibility
                            between the new MID sequence and other pre-
                            existing MID sequences belonging to the same
                            MID group.  This must be 'true' or 'false',
                            and defaults to 'true'.

The name of the MID must be unique within the MID group it belongs to
(or unique within the project if the MID is not assigned to an MID group).

The rules for '-checkMidGroup' compatibility are as follows:

- An MID with an undefined sequence is considered compatible with any MID
  group, under the assumption that its compatibility will eventually be
  assessed when a defined sequence gets assigned to the MID.
- An MID with a defined sequence must have the same length as other
  defined MID sequences within an MID group to be compatible with the
  group. If the new MID sequence is the first defined sequence added to
  the MID group, the required sequence length for subsequent MIDs of the
  group with be the length of that first defined MID sequence.
- An MID with a defined sequence must not be identical (ignoring case)
  with any other defined, pre-existing MID sequence of the same MID group.

If it becomes necessary to edit existing MIDs in a way that temporarily
leaves the MIDs in a group in an inconsistent state (such as changing the
lengths of sequences in an MID group), '-checkMidGroup' should be set to
'false'.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.16.3 update midGroup

```
update midGroup <midGroup name> [-annot[ation] <annotation>]
                                [-file <file> [-format <format>]]

update midGroup -name <midGroup name>
                [-annot[ation] <annotation>]
                [-file <file> [-format <format>]]

Updates an MID group in the currently open project.  In the first form,
the non-option argument is used as the name of the MID group to update.
In the second, a name must be explicitly specified in option form.  The
remainder of the options are not required but can be used to set
properties of the MID group.

    -annotation     The annotation.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.16.4 update multiplexer

```
update mul[tiplexer] <multiplexer name> [-enc[oding] <encoding>]
                                        [-annot[ation] <annotation>]
                                        [-file <file> [-format <format>]]

update mul[tiplexer] -name <multiplexer name>
                     [-enc[oding] <encoding>]
                     [-annot[ation] <annotation>]
                     [-file <file> [-format <format>]]

Updates a multiplexer in the currently open project.  In the first form,
the non-option argument is used as the name of the multiplexer to update.
In the second, a name must be explicitly specified in option form.

The remainder of the options are not required but can be used to set
properties of the new multiplexer.

    -annotation         The annotation.
    -encoding           The MID layout type for the multiplexer, where
                            the choices are both, either, primer1, and
                            primer2.

The four '-encoding' types have the following definitions:

    both                 Both primer 1 and primer 2 MIDs are present and
                             necessary to determine the sample for each
                             read.
    either               Both primer 1 and primer 2 MIDs are present, but
                             either one is sufficient to determine the
                             sample.  For a given read, the MID at the
                             5' end, in the read's orientation, is used
                             to determine the sample.
    primer1              MIDs are only present adjacent to primer 1.
    primer2              MIDs are only present adjacent to primer 2.

If the multiplexer was initially created without specifying the
'-encoding' type, the '-encoding' type must be set using the 'update
multiplexer' command before MIDs or MID<->Sample associations can be
created using the multiplexer.

If the multiplexer already has a defined '-encoding' type and that type is
changed, then all pre-existing sample associations for the multiplexer
will be removed and certain pre-existing associations with MIDs may also
be removed. Specifically, if the '-encoding' type is changed to 'either'
and the numbers of already associated primer 1 and primer 2 MIDs are
not equal, then both sets of MID associations will be removed.  If the
'-encoding' type is changed to 'primer1', then any associated primer 2
MIDs will be dissociated and if the type is changed to 'primer2', then any
associated primer 1 MIDs will be dissociated.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.16.5 update project

```
update proj[ect] [-annotation <annotation>]

Updates the currently open project.  The options specify what properties of
the project to update.

    -annotation     The annotation describing the project.
```

11.4.16.6 update readData

```
update readData <read data name> [-annot[ation] <annotation>]
                                 [-readGroup <read group name>]
                                 [-active <boolean>]
                                 [-originalPath <original path>]
                                 [-file <file> [-format <format>]]

update readData -name <read data name>
                [-annot[ation] <annotation>]
                [-readGroup <read group name>]
                [-active <boolean>]
                [-originalPath <original path>]
                [-file <file> [-format <format>]]

Updates a read data in the currently open project.  In the first form,
the non-option argument is used as the name of the read group to update.
In the second, a name must be explicitly specified in option form.  The
remainder of the options are not required but are used to set properties of
the read data.

    -annotation     The annotation.
    -readGroup      The name of the read group to which this read data
                        belongs.
    -active         The active status of the read data.  This can be one of
                        “true” or “false”.
    -originalPath   The original path of the read data.  The project
                        remembers the original path from which the read
                        data was imported.  This is used to update that
                        path.

Run ‘help general tabularCommands’ for information about the ‘-file’
option.
```

11.4.16.7 update readGroup

```
update readGroup <read group name> [-annot[ation] <annotation>]
                                   [-file <file> [-format <format>]]

update readGroup -name <read group name>
                [-annot[ation] <annotation>]
                [-file <file> [-format <format>]]

Updates a read group in the currently open project.  In the first form, the
non-option argument is used as the name of the read group to update.  In
the second, a name must be explicitly specified in option form.  The
remainder of the options are not required but are used to set properties of
the read group.

    -annotation     The annotation.

Run ‘help general tabularCommands’ for information about the ‘-file’
option.
```

11.4.16.8 update reference

```
update ref[erence] <reference name> [-annot[ation] <annotation>]
                                    [-seq[uence] <sequence>]
                                    [-file <file> [-format <format>]]

update ref[erence] -name <reference name>
                   [-annot[ation] <annotation>]
                   [-seq[uence] <sequence>]
                   [-file <file> [-format <format>]]

Updates a reference sequence in the currently open project.  In the
first form, the non-option argument is used as the name of the reference
sequence to update.  In the second, a name must be explicitly specified in
option form.  The remainder of the options are not required but are used to
set properties of the reference sequence.

    -annotation     The annotation.
    -sequence       The nucleotide sequence string.  This sequence must use
                        IUPAC nomenclature.

Run ‘help general tabularCommands’ for information about the ‘-file’
option.
```

11.4.16.9 update sample

```
update sam[ple] <sample name> [-annot[ation] <annotation>]
                              [-file <file> [-format <format>]]

update sam[ple] -name <sample name>
                [-annot[ation] <annotation>]
                [-file <file> [-format <format>]]

Updates a sample in the currently open project.  In the first form, the
non-option argument is used as the name of the sample to update.  In the
second, a name must be explicitly specified in option form.  The remainder
of the options are not required but are used to set properties of the
sample.

    -annotation     The annotation.

Run ‘help general tabularCommands’ for information about the ‘-file’
option.
```

11.4.16.10 update variant

```
update var[iant] <variant name> [-annot[ation] <annotation>]
                                [-ref[erence] <reference sequence name>]
                                [-pat[tern] <pattern>]
                                [-stat[us] <status>]
                                [-checkPat[tern] <boolean>]
                                [-file <file> [-format <format>]]

update var[iant] -name <new variant name>
                 [-ref[erence] <reference sequence name>]
                 [-annot[ation] <annotation>]
                 [-pat[tern] <pattern>]
                 [-stat[us] <status>]
                 [-checkPat[tern] <boolean>]
                 [-file <file> [-format <format>]]

Updates a variant in the currently open project.  In the first form, the
non-option argument is used as the name of the variant to update.  In the
second, a name must be explicitly specified in option form.  Variants are
allowed to have duplicate names as long as the reference sequences to
which they refer are distinct.  The '-ofRef' argument can be used to refer
to such variants.  For example, if we have two variants named "MyVar", but
one of them refers to "ReferenceSequence1" and the other to
"ReferenceSequence2", we can use the '-ofRef' option to distinguish them.
We can run 'update variant "MyVar" -ofRef "ReferenceSequence1"' to update
the former variant.

The remainder of the options are not required but are used to set
properties of the variant.

    -annotation     The annotation.
    -reference      The name of the reference sequence to which the
                        variant refers.
    -pattern        The pattern that defines the nature of this variation.
    -status         The putative status.  This can be one of "accepted",
                        "rejected", or "putative"
    -checkPattern   Whether the system should check if the variant's
                        pattern is syntactically correct and consistent
                        with the variant's reference sequence. The
                        reference sequence must itself be set and have
                        a non-empty nucleotide sequence for this option
                        to take effect.  This value given must be 'true'
                        or 'false', and defaults to 'true'.

Run 'help general tabularCommands' for information about the '-file'
option.
```

11.4.17 utility

```
util[ity] <utility command> <other arguments>

The 'utility' command is used to execute general utility commands.  For
example, running 'utility clone /data/clone1' will clone the currently open
project to /data/clone1.

The following utility commands are available.  Run
'help utility <utility command>' for more detailed information.

validateNames           Validates the record names in the currently open
                            project.
validateForComputation  Validates that the currently open project is ready
                            for computation.
makeSetupScript         Makes a setup script that, if run, would attempt to
                            recreate the currently open project.
clone                   Clones the currently open project.
execute                 Executes a script file.
```

11.4.17.1 utility validateNames

```
util[ity] validateNames [-fix] [-fixPrefix <prefix>] [-fixSuffix <suffix>]

Validates that the names of records in the currently open project conform
to the requirements of the command interpreter.  Since commands use record
names to identify records, duplicate record names can cause ambiguity.
Additionally, names that are empty or consist entirely of whitespace
can cause syntactic difficulties in certain situations.

This command exists to support the manipulation of projects that
were created using the Graphical User Interface, where the same naming
constraints are not currently applied.  Any project created or edited
with the Command Line Interface will automatically be compatible with
the Graphical User Interface.  This command provides a mechanism to
ensure that the reverse is true as well.

In its default form, this command will report an error if there are
records that will cause ambiguity or syntactic problems if they are
encountered by other commands.  If there are no problematic names, this
command does nothing.

If the ‘-fix’ option is supplied, this command will construct unique,
non-empty names for the records that would have caused problems.  The
constructed names will have the word “FIX_” prepended to them and have
an underscore followed by a number appended to them for uniqueness.  The
prefix is added to provide a marker that this command has modified the
name of the record.

The ‘-fixPrefix’ option specifies a custom string that will be prepended to
modified records instead of the default, “FIX_”.  Setting this option
implies ‘-fix’.

The ‘-fixSuffix’ option specifies a custom string that will appended to
modified records before the number is appended for uniqueness instead of
the default, “-”.  Setting this option implies ‘-fix’ as well.

For example, suppose you have a project with 3 samples all named “MyAmp”.
Running ‘utility validateNames’ will report an error.  Running
‘utility validateNames -fix’ will rename the amplicons to be “FIX_MyAmp_1”,
“FIX_MyAmp_2”, and “FIX_MyAmp_3”.  Running
‘utility validateNames -fix -fixPrefix “FLAG” -fixSuffix “#” will rename
the amplicons to be “FLAG_MyAmp#1”, “FLAG_MyAmp#2”, and “FLAG_MyAmp#3”.

Note that since amplicons and variants can be distinguished by the
reference sequence to which they refer, it is possible to have multiple
amplicons or variants with the same name but different reference sequences.
Such records will not be modified by this command, unless they are empty.
However, amplicons or variants with the same name and reference are
ambiguous and will be modified.
```

11.4.17.2 utility validateForComputation

```
util[ity] validateForComputation [-silent <boolean>]

Validates that the currently open project is ready for computation.  The
project is valid for computation if the following criteria are met.

- Reference sequences have a sequence that is at least 1 base.
- Amplicons refer to valid reference sequences and have target start and
  end coordinates that are contained within said reference sequence.
- Read data files are available that are associated with at least
  one Sample and one or more valid Amplicons.
- Optionally, Variants that refer to valid reference sequences and
  have non-empty patterns that are valid with respect to said reference
  sequence.

If some criteria are not met, warnings are reported describing the
problems, and an error is reported for the operation.  If ‘-silent’ is set
to be true, no warnings are reported, but an error is still reported for
the operation.  If all criteria are met, this command has no effect.
```

11.4.17.3 utility makeSetupScript

```
util[ity] makeSetupScript [-outputFile <file>]

Makes a setup script that, if run with the command interpreter, would
attempt to recreate the currently open project.  Note that it will usually
not be possible to run this script after creating it, since the project
already exists in the same location.

If no '-outputFile' is given, the script is printed to the standard output
of the interpreter.  An output file of "-" has the same effect.  If an
output file is given, the script is written to that file.

Run 'help general filePaths' for more information about specifying files.
```

11.4.17.4 utility clone

```
util[ity] clone <clone project path>
                    [-projectName <clone project name>]
                    [-projectAnnotation <clone project annotation>]
                    [-copyReadData <boolean>]
                    [-scriptOnly <file>]

Clones the currently open project.  The project will be cloned to the path
given as the argument to this command.  By default, the read data and all
project records that depend on the read data will be excluded from the
clone.

By providing the '-projectName' option, the name of the clone project can
be set.  By default, the project name is set to the base name of the clone
project path.

By providing the '-projectAnnotation' option, the annotation of the clone
project can be set.  By default, the annotation of the clone project will
be set to be the same as the currently open project.

If '-copyReadData' is set to "true", the read data and all project records
that depend on the read data will be included in the clone.  If set to
"false" (the default), they will not be included.

If the '-scriptOnly' option is provided, no project will actually be
created.  Instead, a script will be created that, if run by the command
interpreter, will perform the clone.  This allows you to adapt and
customize the clone before actually performing it.  If "-" is provided as
the file, the script will be written to the standard output of the command
interpreter.

Note that the currently cloned project will be cloned as it currently
exists in the command interpreter.  This means that unsaved changes are
propagated to the clone.

Run 'help general filePaths' for more information about the interpretation
of relative paths when using the '-scriptOnly' option or specifying the
clone project path.
```

11.4.17.5 utility execute

```
util[ity] exec[ute] <script path>
                    [-withCurrDir <path>]
                    [-onMissingScript <”ignore”, “warn”, or “error”>]
```

```
Executes a script file.  This allows useful sequences of commands to be
grouped and reused easily.  For example, it may be helpful to create a
script file that creates standard amplicons.  Executing this script will
create the amplicons in the currently open project.

The execution will inherit the environment of the caller.  For example, if
the verbose option is set in the caller, it will also be set in the script.
Modification of the environment by the called script will not be propagated
back to the caller.  For example, suppose the verbose option is set to
“true” in script A at the time it executes script B.  Script B then sets
the verbose option to “false”.  Commands in script B will execute with a
“false” verbose option, but once the execution of script B is complete,
subsequent commands in script A will run with a “true” verbose option.

There are two important exceptions to this policy: the currently open
project and the current directory (currDir).

The currently open project is global.  For example, if script A executes
script B and script B opens a project, that project will continue to be
open in script A.

The current directory for the execution of a script is set by default to
the directory that contains the script itself.  For example, execution of
the script at “someDir/someScript.ava” will run with a current directory of
“someDir”.  This default behavior allows a set of related scripts to
refer to each other using relative paths, independent of where the scripts
are actually installed.  It also allows the commands of a script to easily
refer to a tabular file with the ‘-file’ option in a relative manner when
that tabular file is installed in a location relative to the script.

However, this current directory for the script execution can be modified by
using the ‘-withCurrDir’ option.  If the ‘-withCurrDir’ option is given,
the path passed to it will be used instead. In particular, to invoke a
script that will run with the same current directory as the calling script,
simply do:

     utility execute someDir/someScript.ava -withCurrDir %currDir

In this example, the shortcut path %currDir expands to the current
directory of the calling script, thereby setting the current directory
of the called script to be the same as that of the calling script.  Run
‘help general filePaths’ for more information on the use of relative paths
and other available shortcut paths.

The use of ‘-withCurrDir’ has no effect on the current directory of the
calling script itself.

By providing the ‘-onMissingScript’ option, the behavior of the command, if
the file specified by the script path cannot be found, is customized.  If
set to “ignore”, a missing script will be ignored completely.  If set to
“warn”, a warning will be shown.  If set to “error” (the default), an error
will be reported.

Run ‘help set onErrors’ for information about how errors are handled within
an executed script.
```

11.5 Creating and Computing a Project with the AVA-CLI

There are many different ways to use the AVA-CLI to create or perform computations on a project. These include running `doAmplicon` in interactive mode, typing commands individually, using a program to automatically generate a script that can be piped into the `doAmplicon` command, or manually authoring a script file and invoking `doAmplicon` to execute the commands in that file. The command line syntax for these different options are specified in section 11.3.2.1.

This section provides examples of ways one can piece together a script of commands for setting up a Project. This example Project uses the same underlying data as the example discussed in section 10. The differences for this particular example are that a separate Reference Sequence is being entered for each EGFR exon (rather than stitching them together as an artificial reference); and that all four Read Data files are being utilized (rather than just the region 3 file). Since the sections below often digress to illustrate the variety of commands available to accomplish a specific task, they are not meant to be followed as literal step-by-step instructions for Project creation. However, section 11.5.15 presents an integrated script that can be supplied to the `doAmplicon` command to perform the entire example Project setup, computation and processing (provided you have access to the same sff files and you edit the paths appropriately so the script can find them).

11.5.1 Setting CLI Parameters

You can use the "set" command to change some of the CLI environment parameters within a script (see section 11.4.14 for the usage statement). The "set" command allows you to change the value of three parameters (verbose, onErrors, and currDir).

```
set verbose false
set verbose true
set onErrors stop
set onErrors continue
set currDir <path>
```

Setting verbose to true enables additional logging to enhance troubleshooting capabilities. Each command is logged as it is executed. This is particularly useful for commands that are dynamically synthesized as a side effect of reading tabular input (see section 11.3.2.3). However, whether or not verbose mode is set to true, the CLI will report detailed locations (including, as appropriate, the file name of the script, the line of the script, and the line of any external file or table being read) when it encounters errors.

The onErrors parameter controls how the command interpreter handles any errors it encounters. If onErrors is set to stop (the default, unless running the interpreter in "interactive mode"), the command interpreter will halt and exit with a non-zero exit code when an error is encountered. If onErrors is set to continue, the command that encountered an error will be aborted, but the command interpreter will continue running and execute the rest of the commands in the script.

Because changes to a Project's definition are not permanent until a "save" command is executed, setting the onErrors parameter set to "stop" allows the creation of "transactional" scripts that leave the Project in a consistent state and do not modify the Project definition unless all commands complete without error. This is simply achieved by placing the save command after all the commands that modify the project and with onErrors set to stop: if any of the commands fail, the script will halt, the save command will not be executed, and the Project definition will remain in its original state.

Setting the currDir parameter controls how relative file paths are interpreted by the commands to the CLI. For more information on file paths, see section 11.3.2.6.

11.5.2 Creating a New Project

The first step in creating a Project is setting up the Project directory structure that will store the Project configuration, data, and results. This is done using the "create project" command (see section 11.4.4.2 for the usage statement). Here is an example Project creation command:

```
create project /data/ampProjects/EGFR_CLI \
    -name EGFR_CLI \
    -annotation "CLI Example Project Creation Test"
```

Note the backslash character ("\") used to indicate line continuation. This allows you to control the format of the command over multiple lines, to improve the readability of long commands. This command could also have been presented as one continuous line without the backslashes. Note also the multi-word annotation included within double-quotes. Double-quotes allow spaces and unusual characters to be included in argument values. See section 11.3.2.2 for other specifics on how commands are formatted and parsed. Finally, note that creation of the directory structure for the Project occurs at the time that the "create project" command is executed and is the one aspect of Project definition that will not be reverted if you choose not to save before exiting the CLI.

The "create project" command can be used only once for any given Project. To continue the set up or other work on a Project that has been previously created, use the "open" command (see section 11.4.9 for the usage statement). To open an existing Project, you simply type the Project path after the open command, *e.g.*:

```
open /data/ampProjects/EGFR_CLI
```

Note that this is the actual path to the Project and not the name of the Project. The last part of the Project path and the Project name often coincide because the default name for a Project can be based on the Project directory (see section 10.2.2 for an example of this), but the Project path and the Project name can also diverge (such as if the Project is moved to a new location, perhaps for reasons related to disk space, in which case the Project name would stay the same but the Project path would change). If you try to open a Project that is already open by someone else and you are in interactive mode, a warning will appear and will give you the option to preempt control or to continue with read-only access. If you are using an open command in a script in non-interactive mode, the open command will fail and throw an error that will halt your script (unless onErrors is set to continue).

If you want to intentionally open a Project in read-only mode, you can use the "-control readonly" parameter as part of your open statement. You can also explicitly set the control to "-control preempt" to seize control of the Project you are opening even if someone else is working with it.

11.5.3 Creating References

The next step is to add Reference Sequences since they are necessary for the full specification of Amplicons. This is done using the "create reference" command (see section 11.4.4.7 for the usage statement). Multiple Reference Sequences can be created in a single invocation of the "create reference" command by saving to a file a table containing all the specific Reference Sequence features, and calling the "create reference" command on that file using the "-file" option. The file containing the reference features may be in either tab-separated value (tsv) or comma-separated value (csv) formats, but the file is assumed to be in the tsv format unless the "-format" option is set to "csv" or the file name ends in the suffix ".csv". See section 11.3.2.3 for more details on using tabular files as input.

Generally speaking, any of the commands that support tabular input work by combining the given command line option values with the parameters specified in the table headers and associated values found in the table, to synthesize a set of command options that are the union of both sets of values. This allows all the parameters of the table to be nested within a constant set of options specified on the command line, reducing the chance of error that might occur when manually creating a table with the repeated constant values within the table itself. The example below is the simplest example of using a file as input, in which all the parameters to the "create reference" command are, in fact, coming from the supplied file. See section 11.5.7 for an example of a command that combines options on the command lines with values specified in a file.

Below is an example tsv-formatted table that could be used to create the 5 Reference Sequences for the EGFR Project. Note that the double quotes around the field names and values aren't strictly necessary for this particular example, but quotes would be necessary if the values contained the value delimiter character of the format being used. As explained in section 11.3.2.3, the syntax of the tabular input files follows the standard tsv and csv formatting conventions, not the syntax rules of the CLI itself.

Due to the limitations of this printed document, the "Sequence" entry for each Reference Sequence appears on a separate group of lines, when they are in fact on a single line and are separated from the "Annotation" entry only by a tab character. A similar situation occurs in various other file listings, throughout this Manual section.

```
“Name”    “Annotation”      “Sequence”
“EGFR_Exon_18”  “EGFR_Exon_18”
     “GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACACCC
AGTGGAGAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGAAACTGAATTCAAAAA
GATCAAAGTGCTGGGCTCCGGTGCGTTCGGCACGGTGTATAAGGTAAGGTCCCTGGCAC
AGGCCTCTGGGCTGGGCCGCAGGGCCTCTCATGGTCTGGTGGGG”
“EGFR_Exon_19”  “EGFR_Exon_19”
     “TCACAATTGCCAGTTAACGTCTTCCTTCTCTCTCTGTCATAGGGACTCTGGAT
CCCAGAAGGTGAGAAAGTTAAAATTCCCGTCGCTATCAAGGAATTAAGAGAAGCAACAT
CTCCGAAAGCCAACAAGGAAATCCTCGATGTGAGTTTCTGCTTTGCTGTGTGGGGGTCC
ATGGCTCTGAACCTCAGGCCCACCTTTTCTC”
“EGFR_Exon_20”  “EGFR_Exon_20”
     “CCACACTGACGTGCCTCTCCCTCCCTCCAGGAAGCCTACGTGATGGCCAGCGT
GGACAACCCCCACGTGTGCCGCCTGCTGGGCATCTGCCTCACCTCCACCGTGCAGCTCA
TCACGCAGCTCATGCCCTTCGGCTGCCTCCTGGACTATGTCCGGGAACACAAAGACAAT
ATTGGCTCCCAGTACCTGCTCAACTGGTGTGTGCAGATCGCAAAGGTAATCAGGGAAGG
GAGATACGGGGAGGGGAGATAAGGAGCCAGGATC”
“EGFR_Exon_21”  “EGFR_Exon_21”
     “TCTTCCCATGATGATCTGTCCCTCACAGCAGGGTCTTCTCTGTTTCAGGGCAT
GAACTACTTGGAGGACCGTCGCTTGGTGCACCGCGACCTGGCAGCCAGGAACGTACTGG
TGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCTGGCCAAACTGCTGGGTGCG
GAAGAGAAAGAATACCATGCAGAAGGAGGCAAAGTAAGGAGGTGGCTTTAGGTCAGCCA
GCAT”
“EGFR_Exon_22”  “EGFR_Exon_22”
     “CACTGCCTCATCTCTCACCATCCCAAGGTGCCTATCAAGTGGATGGCATTGGA
ATCAATTTTACACAGAATCTATACCCACCAGAGTGATGTCTGGAGCTACGGTGAGTCAT
AATCCTGATGCTAATGAGTTTGTACTGAGGCCAAGCTGG”
```

The header of the table shows the names of the parameters you want to supply to the "create reference command". In order to use the table to create the Reference Sequences in the Project, it can be saved as a file (*e.g.* "EGFR_CLI_references.txt") in the directory from which you plan to run the script, and cited as argument under the -file option of the "create reference" command:

```
create reference –file EGFR_CLI_references.txt
```

When you use a file as input to a command, the AVA-CLI uses the command and any of its other arguments (besides the "-file" argument) as a prefix command and applies them to each non-header line in the file, such that the contents of each row are converted into their command line option form. Thus, the first Reference Sequence line in our example file gets converted into the following command:

```
create reference EGFR_Exon_18 –annotation EGFR_Exon_18
–sequence
GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACACCCAGTGGA
GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGAAACTGAATTCAAAAAGATCAA
AGTGCTGGGCTCCGGTGCGTTCGGCACGGTGTATAAGGTAAGGTCCCTGGCACAGGCCT
CTGGGCTGGGCCGCAGGGCCTCTCATGGTCTGGTGGGG
```

Tabular input to commands can also be used without saving the table contents to a separate file: you can include the tables directly in your script using the "here" format (which is discussed in section 11.3.2.3). The symbols "- <<" after "-file" indicate that a table in "here" format follows, starting and ending with its "terminator" (in this case, "HERE_TERMINATOR"). The terminator text used to indicate the end of the table should obviously not be found in the table contents. The AVA-CLI will treat the lines following the command as if they were read from a separate file, until it encounters the "HERE_TERMINATOR" text at the beginning of a line.

```
create reference -file - << HERE_TERMINATOR
"Name"    "Annotation"      "Sequence"
"EGFR_Exon_18"  "EGFR_Exon_18"
     "GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACACCC
AGTGGAGAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGAAACTGAATTCAAAAA
GATCAAAGTGCTGGGCTCCGGTGCGTTCGGCACGGTGTATAAGGTAAGGTCCCTGGCAC
AGGCCTCTGGGCTGGGCCGCAGGGCCTCTCATGGTCTGGTGGGG"
"EGFR_Exon_19"  "EGFR_Exon_19"
     "TCACAATTGCCAGTTAACGTCTTCCTTCTCTCTCTGTCATAGGGACTCTGGAT
CCCAGAAGGTGAGAAAGTTAAAATTCCCGTCGCTATCAAGGAATTAAGAGAAGCAACAT
CTCCGAAAGCCAACAAGGAAATCCTCGATGTGAGTTTCTGCTTTGCTGTGTGGGGGTCC
ATGGCTCTGAACCTCAGGCCCACCTTTTCTC"
"EGFR_Exon_20"  "EGFR_Exon_20"
     "CCACACTGACGTGCCTCTCCCTCCCTCCAGGAAGCCTACGTGATGGCCAGCGT
GGACAACCCCCACGTGTGCCGCCTGCTGGGCATCTGCCTCACCTCCACCGTGCAGCTCA
TCACGCAGCTCATGCCCTTCGGCTGCCTCCTGGACTATGTCCGGGAACACAAAGACAAT
ATTGGCTCCCAGTACCTGCTCAACTGGTGTGTGCAGATCGCAAAGGTAATCAGGGAAGG
GAGATACGGGGAGGGGAGATAAGGAGCCAGGATC"
"EGFR_Exon_21"  "EGFR_Exon_21"
     "TCTTCCCATGATGATCTGTCCCTCACAGCAGGGTCTTCTCTGTTTCAGGGCAT
GAACTACTTGGAGGACCGTCGCTTGGTGCACCGCGACCTGGCAGCCAGGAACGTACTGG
TGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCTGGCCAAACTGCTGGGTGCG
GAAGAGAAAGAATACCATGCAGAAGGAGGCAAAGTAAGGAGGTGGCTTTAGGTCAGCCA
GCAT"
"EGFR_Exon_22"  "EGFR_Exon_22"
     "CACTGCCTCATCTCTCACCATCCCAAGGTGCCTATCAAGTGGATGGCATTGGA
ATCAATTTTACACAGAATCTATACCCACCAGAGTGATGTCTGGAGCTACGGTGAGTCAT
AATCCTGATGCTAATGAGTTTGTACTGAGGCCAAGCTGG"
HERE_TERMINATOR
```

The use of regular files or "here" files to input data is equivalent, and the choice is up to the user. Regular files can prove especially useful if the elements of the Project definition are supplied to you from a third-party (such as from the client of a sequencing center). On the other hand, "here" files allow you to encapsulate all the data except for the actual Read Data files into a single, relatively portable script. High-throughput environments with a workflow system might generate such scripts, thereby automating the project creation process.

The "create reference" command will normally fail if you try to create a Reference Sequence with a name that already exists. However, there are situations where it may be legitimate to attempt to do so. For example, a script may have correctly created Reference Sequences and then reached a save point before terminating early due to some error in the script; in such a case, it would be useful to be able to fix the problem in the script and just run it again without it throwing errors due to the Reference Sequences previously saved in the system. The optional flag "-orUpdate" allows this: if you try to create a Reference Sequence under a name that already exists in the Project, the "-orUpdate" flag converts the operation into an update, and the annotation and/or sequence provided in the "create" command are used to update the existing Reference Sequence. Without the flag, the name collision would throw an error.

This "-orUpdate" flag is available for most of the create commands, including those for Amplicons, Samples, and Variants. The flag is discussed in more detail for the Amplicon creation example (section 11.5.4).

11.5.4 Creating Amplicons

Now that there are Reference Sequences in the system, Amplicons can be completely specified for the Project. This is done using the "create amplicon" command (see section 11.4.4.1 for the usage statement). To add multiple Amplicons, it is best to use tabular input, as shown for the Reference Sequences, in section 11.5.3. (To best illustrate each command and parameters, the rest of this Project setup tutorial will show all tabular inputs using inline "here" files.)

```
create amplicon -file - << HERE_TERMINATOR
“Name” “Annotation”  “Reference”   “Primer1”     “Primer2”     “Start”       “End”
“EGFR_18_1”   “Amplifies EGFR_Exon_18 from 23 to 66”    “EGFR_Exon_18”
       “GACCCTTGTCTCTGTGTTCTTG”    “CCTCAAGAGAGCTTGGTTGG”      “23”   “66”
“EGFR_18_2”   “Amplifies EGFR_Exon_18 from 60 to 136”   “EGFR_Exon_18”
       “AGCCTCTTACACCCAGTGGA”      “CCTTATACACCGTGCCGAAC”      “60”   “136”
“EGFR_18_3”   “Amplifies EGFR_Exon_18 from 123 to 197”  “EGFR_Exon_18”
       “TGAATTCAAAAAGATCAAAGTG”    “CCCCACCAGACCATGAGA” “123”  “197”
“EGFR_19_1”   “Amplifies EGFR_Exon_19 from 23 to 115”   “EGFR_Exon_19”
       “TCACAATTGCCAGTTAACGTCT”    “GATTTCCTTGTTGGCTTTCG”      “23”   “115”
“EGFR_19_2”   “Amplifies EGFR_Exon_19 from 67 to 183”   “EGFR_Exon_19”
       “TCTGGATCCCAGAAGGTGAG”      “GAGAAAAGGTGGGCCTGAG”“67”   “183”
“EGFR_20_1”   “Amplifies EGFR_Exon_20 from 20 to 108”   “EGFR_Exon_20”
       “CCACACTGACGTGCCTCTC”“GCATGAGCTGCGTGATGAG”“20”   “108”
“EGFR_20_2”   “Amplifies EGFR_Exon_20 from 102 to 194”  “EGFR_Exon_20”
       “GCATCTGCCTCACCTCCAC”“GCGATCTGCACACACCAG” “102”  “194”
“EGFR_20_3”   “Amplifies EGFR_Exon_20 from 153 to 244”  “EGFR_Exon_20”
       “GGCTGCCTCCTGGACTATGT”      “GATCCTGGCTCCTTATCTCC”      “153”  “244”
“EGFR_21_1”   “Amplifies EGFR_Exon_21 from 23 to 113”   “EGFR_Exon_21”
       “TCTTCCCATGATGATCTGTCCC”    “GACATGCTGCGGTGTTTTC”“23”   “113”
“EGFR_21_2”   “Amplifies EGFR_Exon_21 from 111 to 215”  “EGFR_Exon_21”
       “GGCAGCCAGGAACGTACT” “ATGCTGGCTGACCTAAAGC”“111”  “215”
“EGFR_22_1”   “Amplifies EGFR_Exon_22 from 21 to 132”   “EGFR_Exon_22”
       “CACTGCCTCATCTCTCACCA”      “CCAGCTTGGCCTCAGTACA”“21”   “132”
HERE_TERMINATOR
```

The Amplicons in this example are fully specified: the Reference Sequences used happen to have enough context to include both the Primer1 and Primer2 matches, and the exact "Start" and "End" target coordinates are known and specified. If the exact coordinates of the targets were not known, asterisks ("*") could be used as wild cards for the Start and End fields. This would force the application to try and determine the target coordinates by finding perfect matches for the primers, just as the GUI does (see section 10.2.4). "N" characters are counted as matches as long as they don't make up more than 50% of the match. The results of an automatic primer match must yield one and only one pair of primer matches. If no pair match is found or if more than one match is found for a primer, an error is generated.

The software carries out the primer search, even if the asterisk notation is not used, to validate the target coordinates you supplied manually. This could lead to an error in the cases where your primers have an intentional mismatch with the reference, or where your primers are not included as part of the Reference Sequences. In such cases you would need to supply the correct target coordinates manually, and disable the automatic verification of the target coordinates using the "-checkPrimerMatch false" option to the "create amplicon" command.

The "create amplicons" command also has an "-orUpdate" flag like the one discussed for the create reference example (section 11.5.3), which can be used if a script terminates prematurely but after creating and saving one or more Amplicons: this flag will prevent errors due to "pre-existing" Amplicon names if you re-run the script (after fixing it), and will update the existing Amplicons with the new data instead.

Unlike with the creation of Reference Sequences, however, Amplicons cannot always rely solely on the use of the "-orUpdate" flag because the uniqueness requirement for the naming of Amplicons is only at the level of each Reference Sequence, not for the whole Project. In the case where Amplicons with the same name have been defined relative to different Reference Sequences, the "-orUpdate" flag would not know which Amplicon it is intended for. Such a situation can be resolved by using an "-ofRef" parameter to specify the Reference Sequence of the intended Amplicon. Without an "-ofRef" to properly disambiguate identically named Amplicons when using an "-orUpdate", the "create amplicon" command will fail and throw an error.

11.5.5 Creating Variants

If known Variants exist, they can be added to the Project using the "create variant" command (see section 11.4.4.9 for the usage statement). As was the case for Amplicons (section 11.5.4), the Reference Sequence relative to which a Variant is defined must pre-exist in the project. An example using a "here" style table is below.

```
create variant -file - << HERE_TERMINATOR
"Name"      "Annotation"      "Reference" "Pattern"   "Status"
"15BP_DEL_93-107" "Pattern entered manually"    "EGFR_Exon_19"
      "d(93-107)" "accepted"
"HAP_97C_126A"    "Created from selections"     "EGFR_Exon_18"
      "s(97,C)s(126,A)" "accepted"
"SUB_A_to_C_97"   "Created from selections"     "EGFR_Exon_18"
      "s(97,C)"   "accepted"
"SUB_G_to_A_126"  "Created from selections"     "EGFR_Exon_18"
      "s(126,A)"  "accepted"
HERE_TERMINATOR
```

Another similarity with the "create amplicon" command is that a Project can also have multiple Variants of the same name, as long as they are defined relative to different Reference Sequences. So, the "create variant" command also has both the "-orUpdate" flag and the "-ofRef" parameter, which functions the same way as when used with the "create amplicon" command (section 11.5.4).

The "create variant" command has an additional option used to verify the "pattern" given for the Variants being created: "-checkPattern". When this option is set to "true" (the default value), the "pattern" you set for each Variant is validated in three different ways.

1. The pattern is first checked to make sure that it is syntactically correct.
2. Secondly, the coordinates of the pattern are checked to make sure that they actually exist within the Reference Sequence specified for the Variant.
3. Thirdly, any substitution code must actually create a difference at the specified position (thus, specifying s(10,C) when position 10 is already a C in the Reference Sequence would be an error).

If a check is conducted and any of the three validation criteria are not met, an error will be thrown. However, the check does not occur if the Variant does not have a Reference Sequence assigned to it, or if the Reference Sequence's nucleotide sequence is empty; this allows you to add incomplete Variant definitions or to add Variant place holders before you have specified your Reference Sequences, if you so desire, without causing the "create variant" command to fail. You can also disable the "-checkPattern" option by setting it to "false".

11.5.6 Creating Samples

Samples are easily created as they consist only of a name and an optional annotation. Samples are added using the "create sample" command (see section 11.4.4.8 for the usage statement). Below is an example creating 7 Samples, using a "here" table.

```
create sample -file - << HERE_TERMINATOR
"Name"    "Annotation"
"Sample1" "Sample1"
"Sample2" "Sample2"
"Sample3" "Sample3"
"Sample4" "Sample4"
"Sample5" "Sample5"
"Sample6" "Sample6"
"Sample7" "Sample7"
HERE_TERMINATOR
```

The "create sample" command has an "-orUpdate" flag like the one discussed for the "create reference" example, above (section 11.5.3).

11.5.7 Associating Samples with Amplicons

With the Amplicon and Samples defined, we can now associate them according to the requirements of the experiment. This is done using the "associate" command (see section 11.4.1 for the usage statement). For this example, the Samples are being used to pool Amplicons from shared Reference Sequences. You can create commands to process a single Sample at a time using tabular file input (shown as a "here" block below):

```
assoc -sample Sample1 -file - << HERE_TERMINATOR
"amplicon"      "ofRef"
"EGFR_20_1"     "EGFR_Exon_20"
"EGFR_20_2"     "EGFR_Exon_20"
"EGFR_20_3"     "EGFR_Exon_20"
HERE_TERMINATOR
```

This example also illustrates how command line options are combined with tabular contents. In this case, the single given command is actually translated into the three separate commands:

```
assoc -sample Sample1 -amplicon EGFR_20_1 -ofRef EGFR_Exon_20
assoc -sample Sample1 -amplicon EGFR_20_2 -ofRef EGFR_Exon_20
assoc -sample Sample1 -amplicon EGFR_20_3 -ofRef EGFR_Exon_20
```

Alternatively, the sample name can be used as a field in the file rather than as an argument on the command line, allowing multiple Sample – Amplicon associations to be established from a single command:

```
assoc -file - << HERE_TERMINATOR
“sample”  “amplicon”  “ofRef”
“Sample1” “EGFR_20_1” “EGFR_Exon_20”
“Sample1” “EGFR_20_2” “EGFR_Exon_20”
“Sample1” “EGFR_20_3” “EGFR_Exon_20”
“Sample2” “EGFR_18_1” “EGFR_Exon_18”
“Sample2” “EGFR_18_2” “EGFR_Exon_18”
“Sample2” “EGFR_18_3” “EGFR_Exon_18”
“Sample3” “EGFR_18_1” “EGFR_Exon_18”
“Sample3” “EGFR_18_2” “EGFR_Exon_18”
“Sample3” “EGFR_18_3” “EGFR_Exon_18”
“Sample4” “EGFR_19_2” “EGFR_Exon_19”
“Sample4” “EGFR_19_1” “EGFR_Exon_19”
“Sample5” “EGFR_20_2” “EGFR_Exon_20”
“Sample5” “EGFR_20_1” “EGFR_Exon_20”
“Sample5” “EGFR_20_3” “EGFR_Exon_20”
“Sample6” “EGFR_21_2” “EGFR_Exon_21”
“Sample6” “EGFR_21_1” “EGFR_Exon_21”
“Sample7” “EGFR_22_1” “EGFR_Exon_22”
HERE_TERMINATOR
```

Note the “ofRef” field, used as a safety measure to make sure that the correct Amplicon is being specified, as described above (section 11.5.4). In this particular example, however, the “ofRef” field is not actually necessary since all the Amplicon names are unique within the whole Project, and is shown only for illustrative purposes.

For more uniform Projects, where the same Amplicons are measured in each Sample, an asterisk (“*”) can be used as a shortcut for the Amplicon names. This associates all the Amplicons defined in the Project at that time point with the specified Sample. You can also combine the asterisk for Amplicons with an “ofRef” to selectively associate all and only the Amplicons derived from the “ofRef” Reference Sequence with the Sample specified.

Rather than directly associating Amplicons with Samples in advance, one might consider directly associating Samples with Amplicons and particular Read Data Sets. At such time, the corresponding Sample-Amplicon associations will be implicitly made. However, this requires that the Read Data Sets be first loaded into the Project as described in the section below (11.5.8).

11.5.8 Loading Read Data Sets

Read Data Sets are imported into the Project using the "load" command (see section 11.4.8 for the usage statement). The exact way to load the Read Data Sets into the Project depends on how the data is organized on the disk. In the case where individual SFF files are stored together in a given repository (*e.g.* /data/sffFiles/EGFR_sff_files) and you know the specific file names you need, you might load the files as shown below (shown in the "here" file format):

```
load –sffDir /data/sffFiles/EGFR_sff_files -file - << HERE_TERMINATOR
"SffName"         "ReadGroup"  "SymLink"   "Name"
"DGVS90J01.sff"   "ReadGrp_1"  "false"     "DGVS90J01"
"DGVS90J02.sff"   "ReadGrp_1"  "false"     "DGVS90J02"
"DGVS90J03.sff"   "ReadGrp_1"  "false"     "DGVS90J03"
"DGVS90J04.sff"   "ReadGrp_1"  "false"     "DGVS90J04"
HERE_TERMINATOR
```

Note that a Read Group can be specified as above even if it was not explicitly added to the Project beforehand: the load command will automatically create Read Groups of given names, as needed. To explicitly add a Read Group to the Project, use the "create readGroup" command (see section 11.4.4.6 for the usage statement). In the example above, if you had wanted to create the read group in advance you could have typed "`create readGroup ReadGrp_1`".

In our example, the 4 SFF files we want to load into the Project happen to be from the same sequencing Run. This allows us to use a more compact form of the command, with the "-filePrefix" option:

```
load –sffDir /data/sffFiles/EGFR_sff_files \
–readGroup ReadGrp_1 -filePrefix DGVS90J \
–regions 1,2,3,4 –symLink "false"
```

Specifying the file prefix and choosing which regions to load gives the command enough context so each file does not need to be specified individually. Using the file prefix also provides sufficient specificity to prevent sff files from another Run from being erroneously picked up during the load. If there were a firstRun01.sff and a secondRun01.sff in the directory and you specified region 1 without a file prefix, both files would be imported.

If the SFF files you want to import are not gathered together in a repository but are instead still located in their respective Run analysis directories, the situation may be a bit different: while you might know what Runs and regions you want to import, you may not know the specific file prefix to use. This is because a Run's SFF file names are assigned during analysis by the pipeline software. To get around this, one may set up an "alias" for the loaded files which you can use to refer to the files by region without knowing their actual names. In this next example, we use the alias mechanism, and further use the "analysisDir" option to specify a Run analysis directory:

```
 load –analysisDir /data/sequencingRuns/EGFR_Run_Dir/EGFR_Analysis_Dir/ \
–readGroup ReadGrp_1 –regions 1,2,3,4 \
–symLink false –alias EGFR_reads
```

In the command above, the alias has been set to "EGFR_reads". With this alias established, you would refer to the region 1 file as "EGFR_reads01" and to the region 4 file as "EGFR_reads04". The files are still loaded into the Project with their actual names, but the alias enables you to refer to them without knowing those names. With this facility, you could actually create the script for processing an Amplicon Project in advance of the completion of Pipeline analysis.

In most cases, when Read Data Sets are loaded into a Project, actual copies of the read data files are stored within the directory structure of the Project. This makes the Project directory portable so it can be moved to another location and maintain its integrity as a functional Project. However, you can save disk space and transfer time by setting the "symlink" parameter to "true" (it defaults to "false"). Enabling symlinking causes the Read Data Sets loaded to be stored as symbolic links pointing to the original read data files instead of creating physical copies of the data in the Project folder. If you use this option, your Projects may become nonfunctional if you either move the Project to a location where the symlinks can no longer reach the original data (such as to a different computational host) or if you move or delete the original data. Without access to the Read Data, you will be unable to rerun computations for the Project, and you will also be unable to view Flowgrams.

11.5.9 Associating Read Data Sets with Samples

With the Sample-Amplicon associations already made and the Read Data Sets loaded into the Project, the Samples with their associated Amplicons can now be associated to the proper Read Data Sets. This uses the "associate" command, described in section 11.5.7 where it was used to associate Samples with Amplicons (the usage statement is in section 11.4.1). In this case, the command must supply a Sample and a Read Data Set (or a Read Group).

```
assoc -file - << HERE_TERMINATOR
"readData"      "sample"
"DGVS90J01"     "Sample1"
"DGVS90J02"     "Sample2"
"DGVS90J03"     "Sample6"
"DGVS90J03"     "Sample7"
"DGVS90J03"     "Sample3"
"DGVS90J03"     "Sample5"
"DGVS90J03"     "Sample4"
HERE_TERMINATOR
```

When you make the association between a Sample and a Read Data Set, all the Amplicons associated with that Sample at that time are used to form three-way Read Data Set - Sample - Amplicon associations. This is equivalent to dragging a Sample, in the GUI, onto a Read Data Set in the Read Data Tree. These associations are used during Demultiplexing to determine which reads of a given Read Data Set belong to which Samples. As with dragging a Sample onto a Read Data Set in the GUI, when a Sample is associated with a Read Data Set, the Sample itself must already have an association with at least one Amplicon. An error is generated in the CLI if such a Sample-Amplicon association does not exist at the time you attempt to associate the Sample with the Read Data Set.

Rather than first creating Sample-Amplicon associations and using the Samples to then indirectly create Read Data Set – Sample – Amplicon associations, you can directly create these three-way associations by specifying all three entities in a single command. For example:

```
assoc -readData DGVS90J04 -sample Sample8 -amplicon EGFR_20_1
```

As this command only forms the association for the triad, the three individual entities must have been previously defined. It must be remembered that for a given Read Data Set, an Amplicon can only be assigned to one Sample; an "associate" command that would violate this constraint would return a warning message and the association will not be made. If the Sample and Amplicon specified in the "three-way association" form of the command, above, were not already associated with each other, that two-way association would simultaneously be made when the command executes.

In a Project where the same Sample (or list of Samples) is being measured against many Read Data Sets, the "associate" command can be simplified by the judicious use of Read Groups: associating a Sample (with its already associated Amplicons) to a Read Group forms the Read Data Set - Sample - Amplicon association triads with each Read Data Set present in the Read Group. For example:

```
assoc -readGroup ReadGrp_1 -sample Sample1
```

would associate Sample1 and all its currently associated Amplicons with each of the Read Data Sets of Read Group ReadGrp_1.

In a different context, where you may be measuring all the Amplicons defined in a Project for a Sample (or the same set of Amplicons for every Sample of the Project), you can more succinctly specify the triad association using the asterisk ("*") notation for the Amplicons. For example:

```
assoc -readGroup ReadGrp_1 -sample Sample8 -amplicon *
```

The command above associates every Amplicon currently defined in the Project with the Sample specified and then associates the Sample and these Amplicons with every Read Data Set in the Read Group. You can include an "ofRef" parameter in the association command to restrict the Amplicons being associated to those derived from a single particular Reference Sequence.

Note that the command usage above is appropriate for establishing the Read Data Set – Sample – Amplicon associations in projects in which MIDs are not used. For an example of a CLI script that creates such associations for an MID-based experiment, see section 11.6

11.5.10 Editing Object Properties

After you have finished entering various objects into a Project, you may find it necessary to edit properties and correct mistakes. There are several commands that let you alter the Project data after they have already been entered into the system.

11.5.10.1 Updating an Object

One way to edit an object is to use the "update" command (see section 11.4.16 for the usage statement). In the example Project, we find that the region 4 Read Data Set is actually empty. Since the "load" command used to import Read Data Sets defaults the "active" flag to true, we need to change that flag to false, for this Read Data Set:

```
update readData EGFR_reads04 -active false
```

The update command is of the form "update <entity type> <name of entity> [other options]" where the "other options" are used to set the values for properties appropriate for the entity type. The only object property that cannot be updated via the update command is the object's name. To change the name of an object, use the "rename" command (section 11.5.10.2, below).

When updating an Amplicon or a Variant, you can use an "ofRef" parameter to fully specify the entity of interest in cases where the multiple entities may share the same name (but are uniquely named in the context of their particular Reference Sequences). If an update command encounters an ambiguous Amplicon or Variant name, the command will fail and an error will be generated.

11.5.10.2 Renaming an Object

Although you can change most of the properties of an object using the update command (section 11.5.10.1), an object name change requires the "rename" command (see section 11.4.11 for the usage statement). For example:

```
rename sample –name Sample8 –newName Vial_XYZ
```

The general syntax of the rename command is: "rename <entity type> -name <existing name of entity> -newName <new name for entity>, whereby you provide the original name as the "name" parameter, and the name you want to change it to as the "newName" parameter. An alternative, more succinct, form of the command allows the "-name" and "-newName" option keywords to be left out with the old and new names being given positionally, as in:

```
rename sample Sample8 Vial_XYZ
```

As with the update command (section 11.5.10.1), if the Project contains Amplicons or Variants with duplicate names (but that are defined relative to different Reference Sequences), you must supply an "ofRef" parameter to specify which particular Amplicon or Variant you want to rename. If a rename command encounters an ambiguous Amplicon or Variant name, the command will fail and an error will be generated.

11.5.10.3 Removing an Object

If you find that you no longer have a need for an object, or if you find that an object you imported would require more work to edit than to create from scratch, you can use the "remove" command to eliminate it from the Project altogether (see section 11.4.10 for the usage statement). For example:

```
remove sample Sample8
```

Most of the remove commands have the syntax shown above, with the entity type and entity name as the only arguments to the command. Removing Amplicons and Variants, however, may also involve adding an extra "ofRef" parameter to resolve ambiguities where the Project contains Amplicons or Variants with duplicate names (but that are uniquely named relative to their particular Reference Sequences). If a remove command encounters an ambiguous Amplicon or Variant name, the command will fail and an error will be generated.

Be aware that removing an object can have a cascade of downstream consequences. If you delete a Read Data Set, the associations it has with any Sample-Amplicon sets will be severed (but the Samples will remain associated with the same Amplicons they had before the removal). Removing a Read Group will also remove all the Read Data Sets it contains (with the repercussions above). If you remove a Reference Sequence, you will also remove all the Amplicons and Variants associated with that Reference Sequence. Removing a Sample or an Amplicon severs any Read Data Set – Sample or Sample – Amplicon associations that it used to participate in. Removing a Variant does not impact other objects.

As for other commands, above, an asterisk ("*") can be used in place of an entity name to mean all instances of an entity type. This allows you to easily remove all the entities of the specified type. You can combine the "ofRef" parameter with an asterisk to remove only those entities (Amplicons or Variants) from the specified Reference Sequence.

11.5.10.4 Dissociating Relationships

Sometimes, the objects you have entered into the Project are all correct, but you may have made the wrong associations between some of them; or, you may have cloned a previous Project with similar objects, but the new Project structure may be slightly different. In cases such as these, you would want to dissolve the incorrect associations without removing the objects from the Project. The "dissociate" command serves that purpose (see section 11.4.5 for the usage statement).

This command is used to influence the three-way Read Data Set – Sample – Amplicon relationships (as displayed in the Project tab's Read Data tree of the GUI) and the Sample-Amplicon relationships (displayed in the Project tab's Samples tree of the GUI). As such, it has two general flavors: one primarily affects the Samples tree and the other, primarily the Read Data tree; but there can be changes in both trees as a result of either command type.

If you are primarily trying to influence the Sample-Amplicon relationships (as seen in the Samples tree), you should use the form of the command where you supply a Sample and an Amplicon as arguments. For example:

```
dissoc -sample Sample6 -amp EGFR_21_1
```

The command above removes the association between Sample6 and Amplicon EGFR_21_1 but does not remove either object from the Project. Additionally, while removing this association, any three-way Read Data Set – Sample – Amplicon associations for Sample 6 and EGR_21_1 would also be removed. In the GUI this would be reflected as elements being removed from the corresponding Read Data Sets of the Read Data tree.

If more than one Amplicons of the same name are associated with a Sample (but are uniquely named relative to their particular Reference Sequences), you must use the "-ofRef" parameter to specify the Amplicon to which you want to apply the dissociation. If you don't use the "-ofRef" parameter in this situation, an error will be generated.

You can use an asterisk ("*") as the Amplicon specifier in the command to dissociate all Amplicons from a Sample, with the concomitant dissociation of the related Read Data – Sample – Amplicon associations that may exist. The asterisk notation can be combined with the "-ofRef" parameter to dissociate from the Sample all the Amplicons that are defined relative to a specified Reference Sequence (but maintain the associations of Amplicons from other Reference Sequences).

If you are primarily trying to influence the associations of a particular Read Data Set with its Samples and Amplicons, you should use the form of the command where you specify a Read Data, a Sample, and optionally an Amplicon as arguments. For example:

```
dissoc -readdata DGVS90J02 -samp Sample2 -amp EGFR_18_1
```

The command above dissociates the Read Data Set - Sample – Amplicon association triplet among DGVS90J02, Sample2, and EGFR_18_1, but all three objects still individually remain in the project. Dissociation of a Read Data Set - Sample – Amplicon triplet does NOT influence the corresponding Sample – Amplicon paired association, which is maintained.

If more than one Amplicon of the same name are associated with a Sample (but are uniquely named relative to their particular Reference Sequences), you must use the "-ofRef" parameter to specify the Amplicon to which you want to apply the dissociation. If you don't use the "-ofRef" parameter in this situation, an error will be generated.

You can use an asterisk ("*") as the Amplicon specifier in the command to dissociate all the Read Data Set - Sample – Amplicon triplets based on a given Read Data Set and Sample. As before, the dissociation of these triplet associations has no effect on the corresponding pairwise Sample-Amplicon relationships (as viewed in the Samples tree of the GUI). The asterisk notation can be combined with the "-ofRef" parameter to dissociate only those Read Data - Sample – Amplicon triplets where the Amplicons are defined relative to a specified Reference Sequence (but maintain the associations for Amplicons from other Reference Sequences).

A shortened form of the Read Data-centric command allows you to omit the Amplicon specification and only supply a Read Data and a Sample. This is interpreted as if you had specified the Amplicon as an asterisk.

```
dissoc -readdata DGVS90J01 -samp Sample1
```

11.5.11 Computation

The CLI can also be used to trigger the computation of a Project once it has been set up properly.

11.5.11.1 Validating the Project before Computation

There are two utility commands that can be used to validate a Project before computation: "utility validateNames" and "utility validateForComputation".

The GUI does not currently enforce unique names for individual objects because there is an internal accounting process other than naming that keeps entities distinct from each other. In the CLI, however, all objects are created and manipulated using their names, so non-unique names can be a problem. Some degree of name duplication can be tolerated for Amplicon and Variant objects if the Reference Sequences for those objects can be combined with the object names to unambiguously identify individual objects (using the "-ofRef" parameter available in various commands). Any name duplication that cannot be disambiguated in this manner must be fixed. Additionally, the GUI can allow objects to have empty names, which also would cause problems for the CLI.

The "utility validateNames" command can be used to detect and correct naming problems in Projects (see section 11.4.17.1 for the usage statement). Running the command without any arguments will report an error if any problem names are encountered (irresolvable duplicates or empty names). The command does nothing if there are no errors to report.

You can also use the "-fix" flag with the "utility validateNames" command to enable it to correct the naming problems it finds (rather than using it just as a problem detection tool as above). The default fix is to put a "FIX_" prefix and an underscore ("_") suffix followed by a unique number. For example, two Reference Sequences with the same name "MyRef" would be converted to FIX_MyRef_1 and FIX_MyRef_2. You can specify alternate prefixes and suffix separators using the "-fixPrefix" and "-fixSuffix" options. The common prefix makes it easy to find these "fixed" names in the sorted Project Tab tables of the GUI and manually adjust them, if desired, to different unique names.

Even if the names are perfectly fine, there can be other problems with a Project that might impact its computation. The command "utility validateForComputation" checks for these problems (see section 11.4.17.2 for the usage statement). Specifically, the command verifies the following:

1. all Reference Sequences contain a sequence that is at least 1 base in length;
2. all Amplicons are associated to valid Reference Sequences and have target start and end coordinates that are contained within that Reference Sequence;
3. all Read Data files that are associated with at least one Sample and one or more valid Amplicons are available; and
4. if Variants are defined in the Project that are associated to valid Reference Sequences, they have non-empty patterns that are valid with respect to that Reference Sequence.

If any of these criteria aren't met, warnings are reported and the command throws an error; otherwise it does nothing. The warnings can be silenced by setting the "-silent" option to "true".

11.5.11.2 Managing the Computation

The "computation" command allows you to start or stop a computation, or check its status (see section 11.4.3 for the usage statement). You are only allowed to use the "computation start" and "computation stop" commands on Projects on which you have full control. If you were able to successfully do a simple "open" on a Project, or if you were able to open it with the "-control preempt" option, you have the appropriate level of control to start or stop a computation.

If you have read-only access to the Project, you cannot influence the course of the computation, but you can run the "computation status" command which will report "running" or "stopped", as appropriate.

Unlike with the GUI, a computation started by the CLI is not run as a separate background process, but rather as part of the CLI process itself. This means that if the CLI instance that started a computation is terminated (via a control-C, for instance) the Project computation will abruptly stop as well. Additionally, it means that the next step of the current CLI script (or the display of the next command prompt, if in interactive mode) will not be executed until the computation finishes. Thus, the "computation status" command for checking the status of a computation is mostly useful for checking on the status of computations that were initiated either from the GUI or from other CLI instances, but not for computations that a particular CLI instance started itself. Unlike the GUI, therefore, the CLI does not currently provide the means to obtain status reporting to determine the current stage of computation or the dispositions of the processed and queued computation steps. However, if a computation were started from a CLI instance, you could open the project in the GUI (perhaps in read only mode, so as not to seize control from the CLI instance that may have additional steps to perform after the Project computation is finished) and track its detailed progress there.

Likewise, because computations run as part of the CLI process, you can't execute a clean "computation stop" within the same instance of the CLI that you used to initiate the "computation start." The safest way to halt a computation is to start another instance of the CLI (or open the GUI), open the Project in preempt mode, and type "computation stop" (in the CLI) or press the stop button on the main Computation tab (in the GUI).

11.5.11.3 Loading Automatically Detected Variants

Once a computation is complete, any Variants that were part of the Project prior to the computation will have their frequency statistics updated. As part of the computation, the application also attempts to automatically detect potential variations in the data. In the GUI, you have the option of importing the automatically detected variants using a "Load" button on the main Variants tab, with the option to narrow down the set to import using a variety of filters. In the current version of the CLI, by contrast, no filters are provided and the import is an all or nothing proposition.

You can import the pool of automatically detected variants in the CLI by using the "computation loadDetectedVariants" command (see section 11.4.3.4 for the usage statement). Note that this command loads the automatically detected Variants into memory, but just like in the GUI, you must save the Project if you want the load to be permanent. Since the load is currently all or nothing, a good strategy may be to load the Variants but not save them, so you can run a report command (see section 11.5.12) on all the potential Variants for a Project without cluttering the GUI view of the Project with too many marginal Variants.

If you choose not to load the detected Variants after a computation triggered from the CLI (or you choose to load them but not save the Project), the Auto-Detected Variants will not be lost to you even if you exit the CLI: they will remain in the Project, in a queue to be loaded in a subsequent session. For instance, you could start a new instance of the CLI, reopen the Project and run the "computation loadDetectedVariants" command, and thus recover the Auto-Detected Variants you chose not to load or save the first time. Similarly, you could finish the computation in the CLI without loading (or with loading and not saving) the Auto-Detected Variants. Later, you can open the Project in the GUI, and you would be able to access the Auto-Detected Variants via the Load button on the main Variants tab.

11.5.12 Reporting

After your Project has finished computing you can open it with the GUI to explore the results and alignments. As usual, the Variants Frequency Table on the main Variants tab of the GUI can be manually exported by clicking on the "text file" button located next to it. The structure of this exported file is in the same two-dimensional geometry (Variants as the rows and Samples as the columns) as the table displayed in the GUI itself. Because of the two-dimensional structure, this format is not particularly amenable to high-throughput processing as one might want to do following a project Computation. The same information can also be generated in an automated fashion, and in a more useful linear form, using the "report variantHits" command from the CLI (see section 11.4.12.1 for the usage statement and output format). Although the GUI allows you to apply various filters to the data before exporting Sample-Variant data, the CLI currently only supports a bulk report of all the Variant statistics. You can generate the report in either tab-separated value (tsv) or comma-separated value (csv) formats. The report only includes Variants that are officially part of the Project, *i.e.* specifically defined Variants, or auto-detected Variants that were "loaded" using the "computation loadDetectedVariants" command (see section 11.5.11.3). If you have any unloaded automatically detected Variants, they will not be included in the output unless you use the "computation loadDetectedVariants" command prior to using "report variantHits."

With the CLI-generated report in hand you can do your own customized processing of the reported Variant frequencies. One suggestion would be to apply filter criteria on the reported data to highlight Variants with the most believable support; users can then focus on these "best candidates" and verify them by examination of the alignments in the GUI.

11.5.13 Finishing Touches

These few additional "Project management" commands can be useful when you have finished working with a Project and are ready to either move on to another one, or you want to shut down the CLI.

11.5.13.1 save

When you have gained control of a Project via a Project creation, a standard open, or an open in preempt mode, you have the freedom to save the modifications you make to that Project at any point you deem appropriate. This is done with the "save" command (see section 11.4.13 for the usage statement). If you close a Project (see section 11.5.13.2) that contains changes without first saving it, the unsaved changes will be discarded. Note that if you have made any modifications to the Project, you MUST run the "save" command to commit those changes to the Project prior to triggering a computation with the "computation start" command. If you trigger a computation on a Project that contains unsaved modifications, an error will be generated.

11.5.13.2 close

When you have finished making changes to the Project, you can close it using the "close" command (see section 11.4.2 for the usage statement). The close statement discards any unsaved changes to the Project and frees the Project lock so someone else can access it without needing to preempt control. If there are unsaved changes to the Project, the CLI will show a warning to let you know that some changes are being discarded. If you are in interactive mode, a "yes", "no", "cancel" prompt will be shown to allow you to avoid discarding the changes if you didn't really mean to close without saving. The CLI is still running after you use the close command, so you can open another Project and do more work.

11.5.13.3 exit

When you are ready to shut down the `doAmplicon` instance you are running, you can use the "exit" command (see section 11.4.6 for the usage statement). As with the close command (section 11.5.13.2, above), the exit command will warn you if you try to exit an Project that contains unsaved changes and, in interactive mode, will prompt you to decide if you want to save before exiting. Since the exit command actually shuts down the `doAmplicon` instance, it will terminate any chain of commands wherever it is introduced (*e.g.*, if you supply three scripts to be executed in succession by a `doAmplicon` instance and the first script has an exit in it, the other two scripts will not be executed).

11.5.14 Exporting from the Project

It can be convenient to use aspects, or the entire definition, of an existing Project as the basis for defining a new Project. Two utility commands are available ("utility makeSetupScript" and "utility clone") that provide the means to export most or all aspects of an existing Project definition at once; and a set of "list" commands are provided to facilitate more focused exports of Project setup data.

11.5.14.1 utility makeSetupScript

The "utility makeSetupScript" command is basically a means of making a backup of the setup of your currently open Project in the CLI (see section 11.4.17.3 for the usage statement). If you provide the command with an "-outputFile" option, the CLI writes a script to that file that contains all the commands that you would have to input to `doAmplicon` to regenerate your Project setup.

The script will contain commands to create the Project directory structure and to create all the Project objects and the required associations between them. Load commands are included to handle the import of the Read Data Sets. However, the script does not back up any computed results for the Project. Generally, you will not be able to run the script immediately after you generate it, because your Project will still be in place and the "create Project" command in the script will fail because you can't create a Project that already exists. The script is intended as a safety measure that you could use to reconstitute your Project should the Project directory become corrupted. Note that the "utility makeSetupScript" command exports the Project setup based on the state of the open Project that you have in memory when you run the command (including any unsaved changes). The load commands generated for the Read Data Sets will try to find the data in the original locations where the data was located when originally imported into the Project (whether via the CLI or the GUI). If that data has been moved to a new location, you will need to manually edit the setup script to reflect the new location. If the data is no longer available, the load commands will fail. Note that the script only backs up the Project setup; it does not backup of the actual Read Data Sets.

11.5.14.2 utility clone

The "utility clone" command is used to create an exact copy of a Project setup to another location (see section 11.4.17.4 for the usage statement). The command is used in the context of an open Project in the CLI, and you provide the location where you want the clone to be created (*e.g.*, "utility clone /data/clonedProjectDirectory"). The command copies the Project directory structure and objects with their associations to the new location, but does not copy the Read Data Sets (unless specifically called for by setting the "-copyReadData" parameter to "true").

A "-scriptOnly" option can be used to prevent the actual execution of the clone operation and instead write out the commands necessary to carry out the cloning process to a script that you can edit and use later as input to a `doAmplicon` command (this is similar to running the "utility makeSetupScript" command, except that the Read Data Sets are not copied by default). The clone operation uses the state of the open Project at the time the command is run, so any unsaved changes involving the Project setup will be included in the clone operation. The cloning operation does not include any computed results for the Project.

If you choose to copy the Read Data Sets as part of the clone command, the read data will be obtained from the open Project and then the "OriginalPath" of the Read Data Sets will be updated to reflect the true original source of the data. This allows the clone to occur even if the original source of the Read Data Sets has been moved or deleted. The read data import should only fail if there are disk space issues or if the Project being cloned uses symbolic links and the links are invalid because the data was moved or deleted.

11.5.14.3 list

Sometimes you may want to export only a specific subset of the Project setup data rather than backing up or cloning an entire Project. Examples may be that you want to reuse some Reference Sequences from one Project in a new Project; or that you want to recycle some Amplicons and/or Variants associated with those Reference Sequences, but you don't want to import any Samples or Sample – Amplicon relationships (*e.g.* because your new Project will have its own Samples, different from the ones of the existing Project). In these cases you can use the "list" command to output tabular data that is suitable to import into a new Project (see section 11.4.7 for the usage statement). The tables returned by the list command can have their format set to tab-separated values (tsv) or comma-separated values (csv). Using the "-outputFile" option, you can specify what file the table should be written to or you can allow the table to be written to standard output.

To import a list table into a new Project, use the corresponding create command for the data type in the file, and either provide the table file using the "-file" parameter or the contents of the file as part of a "here" format file (*e.g.*, "create ref -file listRefTable.tsv", where listRefTable.tsv was previously created from another project via "list ref -outputFile listRefTable.tsv").

The list commands can also be useful in interactive mode as a way to verify the content of the Project as you enter and edit Project objects. Even if you aren't in interactive mode, the list commands can be used in scripts to enhance logging in scripts when troubleshooting.

List commands are available to review and export the basic entities of the Project, but not the associations between Samples, Amplicons, and Read Data Sets managed by the associate command. To view or export those associations one must inspect or run the scripts generated by the "utility makeSetupScript" and "utility clone" commands.

11.5.15 Integrated Project Script

Below is the commented text of an example script that could be used to set up and compute a Project (provided you have access to the Read Data files). You would execute this script by saving it to a file such as "projectSetupScript.ava" and executing it using the CLI ("doAmplicon projectSetupScript.ava"). Alternatively, you could start the CLI in interactive mode and type or copy and paste the commands into the interface individually. Note that the '#' symbol at the beginning of a line is used to indicate a comment line that is ignored by the command interpreter. The data for the object types is supplied in tab-delimited "here" files. If you copy and paste the data, make sure that the space between fields in the "here" files remain tabs when you paste the data in the new location (in some combinations of applications the cut (or copy) and paste operation converts the tabs into spaces).

Due to the limitations of this printed document, certain lines of the script below appear on multiple lines. This occurs for certain tab-separated entries in the tables given as arguments to certain commands. Be aware that these should actually be single lines in the script.

```
# Script to create a project, compute it, and generate a report.
# Edit paths as necessary to conform to your system.

# Create the project architecture. Edit the path if you want to create the
# project in an alternate location.

create project /data/ampProjects/EGFR_CLI \
    -name EGFR_CLI \
    -annotation “CLI Example Project Creation Test”

# This command creates all the reference objects.

create reference -file - << HERE_TERMINATOR
“Name”        “Annotation”     “Sequence”
“EGFR_Exon_18”        “EGFR_Exon_18”
     “GACCCTTGTCTCTGTGTTCTTGTCCCCCCCAGCTTGTGGAGCCTCTTACACCCAGTGGAGAAGCTCCCAACCAAGC
TCTCTTGAGGATCTTGAAGGAAACTGAATTCAAAAAGATCAAAGTGCTGGGCTCCGGTGCGTTCGGCACGGTGTATAAGGTA
AGGTCCCTGGCACAGGCCTCTGGGCTGGGCCGCAGGGCCTCTCATGGTCTGGTGGGG”
“EGFR_Exon_19”        “EGFR_Exon_19”
     “TCACAATTGCCAGTTAACGTCTTCCTTCTCTCTCTGTCATAGGGACTCTGGATCCCAGAAGGTGAGAAAGTTAAAA
TTCCCGTCGCTATCAAGGAATTAAGAGAAGCAACATCTCCGAAAGCCAACAAGGAAATCCTCGATGTGAGTTTCTGCTTTGC
TGTGTGGGGGTCCATGGCTCTGAACCTCAGGCCCACCTTTTCTC”
“EGFR_Exon_20”        “EGFR_Exon_20”
     “CCACACTGACGTGCCTCTCCCTCCCTCCAGGAAGCCTACGTGATGGCCAGCGTGGACAACCCCCACGTGTGCCGCC
TGCTGGGCATCTGCCTCACCTCCACCGTGCAGCTCATCACGCAGCTCATGCCCTTCGGCTGCCTCCTGGACTATGTCCGGGA
ACACAAAGACAATATTGGCTCCCAGTACCTGCTCAACTGGTGTGTGCAGATCGCAAAGGTAATCAGGGAAGGGAGATACGGG
GAGGGGAGATAAGGAGCCAGGATC”
“EGFR_Exon_21”        “EGFR_Exon_21”
     “TCTTCCCATGATGATCTGTCCCTCACAGCAGGGTCTTCTCTGTTTCAGGGCATGAACTACTTGGAGGACCGTCGCT
TGGTGCACCGCGACCTGGCAGCCAGGAACGTACTGGTGAAAACACCGCAGCATGTCAAGATCACAGATTTTGGGCTGGCCAA
ACTGCTGGGTGCGGAAGAGAAAGAATACCATGCAGAAGGAGGCAAAGTAAGGAGGTGGCTTTAGGTCAGCCAGCAT”
“EGFR_Exon_22”        “EGFR_Exon_22”
     “CACTGCCTCATCTCTCACCATCCCAAGGTGCCTATCAAGTGGATGGCATTGGAATCAATTTTACACAGAATCTATA
CCCACCAGAGTGATGTCTGGAGCTACGGTGAGTCATAATCCTGATGCTAATGAGTTTGTACTGAGGCCAAGCTGG”
HERE_TERMINATOR

# This command creates the amplicon objects.

create amplicon -file - << HERE_TERMINATOR
“Name”        “Annotation”     “Reference”      “Primer1”        “Primer2”
“Start”       “End”
“EGFR_18_1”   “Amplifies EGFR_Exon_18 from 23 to 66”    “EGFR_Exon_18”
     “GACCCTTGTCTCTGTGTTCTTG” “CCTCAAGAGAGCTTGGTTGG”   “23”     “66”
“EGFR_18_2”   “Amplifies EGFR_Exon_18 from 60 to 136”   “EGFR_Exon_18”
     “AGCCTCTTACACCCAGTGGA”   “CCTTATACACCGTGCCGAAC”   “60”     “136”
“EGFR_18_3”   “Amplifies EGFR_Exon_18 from 123 to 197”  “EGFR_Exon_18”
     “TGAATTCAAAAAGATCAAAGTG” “CCCCACCAGACCATGAGA”     “123”    “197”
“EGFR_19_1”   “Amplifies EGFR_Exon_19 from 23 to 115”   “EGFR_Exon_19”
     “TCACAATTGCCAGTTAACGTCT” “GATTTCCTTGTTGGCTTTCG”   “23”     “115”
“EGFR_19_2”   “Amplifies EGFR_Exon_19 from 67 to 183”   “EGFR_Exon_19”
     “TCTGGATCCCAGAAGGTGAG”   “GAGAAAAGGTGGGCCTGAG”    “67”     “183”
“EGFR_20_1”   “Amplifies EGFR_Exon_20 from 20 to 108”   “EGFR_Exon_20”
     “CCACACTGACGTGCCTCTC”    “GCATGAGCTGCGTGATGAG”    “20”     “108”
“EGFR_20_2”   “Amplifies EGFR_Exon_20 from 102 to 194”  “EGFR_Exon_20”
     “GCATCTGCCTCACCTCCAC”    “GCGATCTGCACACACCAG”     “102”    “194”
“EGFR_20_3”   “Amplifies EGFR_Exon_20 from 153 to 244”  “EGFR_Exon_20”
     “GGCTGCCTCCTGGACTATGT”   “GATCCTGGCTCCTTATCTCC”   “153”    “244”
“EGFR_21_1”   “Amplifies EGFR_Exon_21 from 23 to 113”   “EGFR_Exon_21”
     “TCTTCCCATGATGATCTGTCCC” “GACATGCTGCGGTGTTTTC”    “23”     “113”
“EGFR_21_2”   “Amplifies EGFR_Exon_21 from 111 to 215”  “EGFR_Exon_21”
     “GGCAGCCAGGAACGTACT”     “ATGCTGGCTGACCTAAAGC”    “111”    “215”
“EGFR_22_1”   “Amplifies EGFR_Exon_22 from 21 to 132”   “EGFR_Exon_22”
     “CACTGCCTCATCTCTCACCA”   “CCAGCTTGGCCTCAGTACA”    “21”     “132”
HERE_TERMINATOR

# This command seeds the project with a few known variants.

create variant -file - << HERE_TERMINATOR
“Name”        “Annotation”     “Reference”      “Pattern”        “Status”
“15BP_DEL_93-107”  “Pattern entered manually”   “EGFR_Exon_19”   “d(93-107)”
       “accepted”
“HAP_97C_126A”     “Created from selections”    “EGFR_Exon_18”   “s(97,C)s(126,A)”
       “accepted”
“SUB_A_to_C_97”    “Created from selections”    “EGFR_Exon_18”   “s(97,C)”
       “accepted”
“SUB_G_to_A_126”   “Created from selections”    “EGFR_Exon_18”   “s(126,A)”
       “accepted”
HERE_TERMINATOR
```

```
# This command creates all the sample objects.

create sample -file - << HERE_TERMINATOR
“Name”        “Annotation”
“Sample1”     “Sample1”
“Sample2”     “Sample2”
“Sample3”     “Sample3”
“Sample4”     “Sample4”
“Sample5”     “Sample5”
“Sample6”     “Sample6”
“Sample7”     “Sample7”
HERE_TERMINATOR

# This command sets up the Sample-Amplicon associations.

assoc -file - << HERE_TERMINATOR
“sample”      “amplicon”       “ofRef”
“Sample1”     “EGFR_20_1”      “EGFR_Exon_20”
“Sample1”     “EGFR_20_2”      “EGFR_Exon_20”
“Sample1”     “EGFR_20_3”      “EGFR_Exon_20”
“Sample2”     “EGFR_18_1”      “EGFR_Exon_18”
“Sample2”     “EGFR_18_2”      “EGFR_Exon_18”
“Sample2”     “EGFR_18_3”      “EGFR_Exon_18”
“Sample3”     “EGFR_18_1”      “EGFR_Exon_18”
“Sample3”     “EGFR_18_2”      “EGFR_Exon_18”
“Sample3”     “EGFR_18_3”      “EGFR_Exon_18”
“Sample4”     “EGFR_19_2”      “EGFR_Exon_19”
“Sample4”     “EGFR_19_1”      “EGFR_Exon_19”
“Sample5”     “EGFR_20_2”      “EGFR_Exon_20”
“Sample5”     “EGFR_20_1”      “EGFR_Exon_20”
“Sample5”     “EGFR_20_3”      “EGFR_Exon_20”
“Sample6”     “EGFR_21_2”      “EGFR_Exon_21”
“Sample6”     “EGFR_21_1”      “EGFR_Exon_21”
“Sample7”     “EGFR_22_1”      “EGFR_Exon_22”
HERE_TERMINATOR

# This load command assumes that the data is sitting in an official analysis
# directory where the data is actually sitting in an sff subdirectory of the
# analysisDir. If you have data sitting in an alternate analysis directory,
# you can specify the analysis path

load   -analysisDir /data/sequencingRuns/EGFR_Run_Dir/EGFR_Analysis_Dir/ \
          -readGroup ReadGrp_1 -regions 1,2,3,4 \
          -symLink false -alias EGFR_reads

# If your read data is in a generic repository, rather than an official
# analysis directory, you can comment out the load above and replace it with
# one like the following where you have edited the sffDir path to point to the
# sff files on your system.

#load –sffDir /data/sffFiles/EGFR_sff_files \
#       –readGroup ReadGrp_1 -filePrefix DGVS90J \
#       –regions 1,2,3,4 –symLink “false”

# If you don’t have access to the specific Read Data for this project
# you have already set up as much of the project as you can and
# you won’t be able to run a computation on it. You should save the
# project setup here and exit. You can open the project in the
# GUI to see how the commands above have been translated into a
# project setup.

# save
# exit

# This command creates the associations between the Read Data and
# the Sample-Amplicon pairs. This command is only valid if you have
# imported the Read Data from an analysis directory using EGFR_reads
# as an alias. If you instead imported the Read Data from a
# repository using actual file names, you would need to change the aliases
# to actual file names (i.e. EGFR_reads01 to DGVS90J01).

assoc -file - << HERE_TERMINATOR
“readData”    “sample”
“EGFR_reads01”        “Sample1”
“EGFR_reads02”        “Sample2”
“EGFR_reads03”        “Sample6”
“EGFR_reads03”        “Sample7”
“EGFR_reads03”        “Sample3”
“EGFR_reads03”        “Sample5”
“EGFR_reads03”        “Sample4”
HERE_TERMINATOR
```

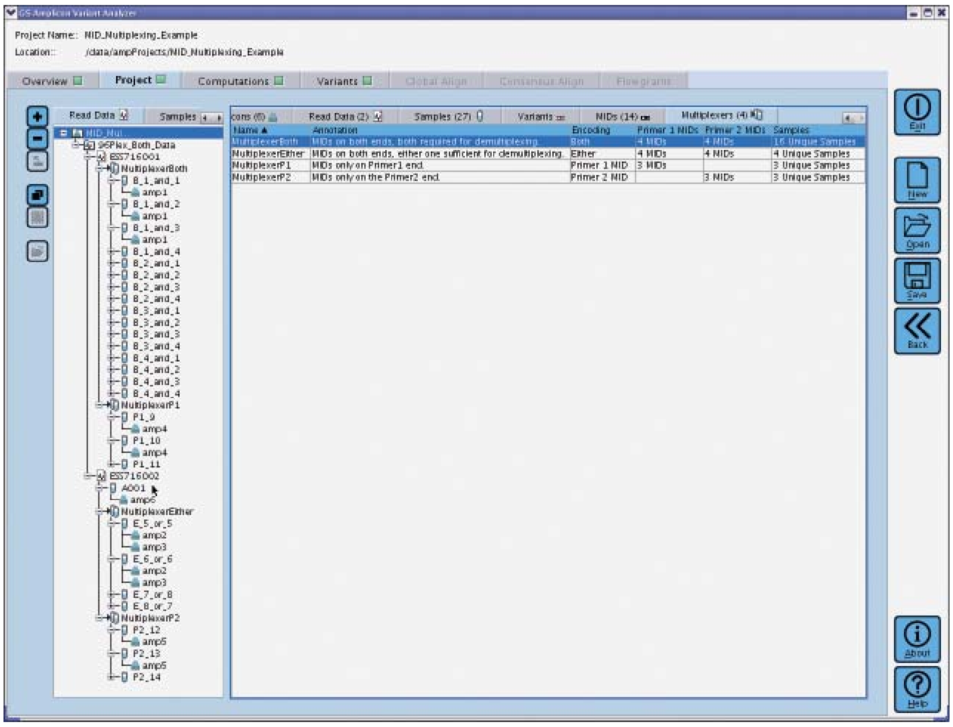
```
# The region 4 Read Data file is actually empty, so we should mark it as
# not active so it gets excluded from the analysis.

update readData EGFR_reads04 -active false

# You should save the project setup now.

save

# Run the validateNames utility to be on the safe side.

utility validateNames

# Check to see if there are any other problems with the project.

utility validateForComputation

# Trigger the start of the computation.

computation start

# Load the automatically detected variants discovered as part of the
# computation.

computation loadDetectedVariants

# Report the measured variant frequencies to a tab-delimited output file.

report variantHits -outputFile EGFR_variant_hits.txt

# Close the project without saving. This will prevent the automatically
# detected variants from being permanently added to the project.
# You will receive a warning about unsaved changes to the project.

close

# Exit the CLI. Your project setup and your computation results should now be
# able to be viewed if you open the project in the GUI.

exit
```

11.6 Creating and Computing an MID Project with the AVA-CLI

The EGFR example Project shown in detail in sections 10 (GUI version) and 11.5 (CLI version) does not display the usage of MIDs and Multiplexers. The example below briefly shows many of the special features of the AVA software that come into play when MIDs are used, and how they would be set up in a Project using the CLI.

A few things to remember are that in a Project that contains multiple Read Data sets, a given Multiplexer can be used for more than one Read Data set, or distinct Read Data sets can each have specific Multiplexer(s). It is also possible to associate more than one Multiplexer with the same Read Data set and to mix regular Samples with Multiplexers on a Read Data set. However, for a given Read Data set, an Amplicon can only be assigned to one entity (a regular Sample or a Multiplexer). Figure 11–1 shows the Read Data Tree and Multiplexers Definition Table for an example Project that is atypically complex for tutorial purposes, as it was intentionally constructed to illustrate a wide variety of Multiplexer features, whereby:

- 4 Multiplexers are used, showing all 4 encoding types (Both, Either, Primer 1 MID, and Primer2 MID)
- "MultiPlexerBoth" and "MultiplexerP1" are associated with Read Data ESS716O01 (as seen in the Tree), and "MultiplexerEither" and "MultiplexerP2" are associated with Read Data set ESS716O02
- the Tree nodes are partially expanded to reveal which Amplicons are being demultiplexed by which Multiplexer:
  - "MuliplexerBoth" is being used to assign a single amplicon "amp1" to 16 Samples
  - "MultiplexerP1" is being used to assign "amp4" to 3 different Samples
  - "MultiplexerEither" is being used to assign two different Amplicons ("amp2" and "amp3") to 4 different Samples
  - "MultiplexerP2" is being used to assign "amp5" to 3 different Samples
  - MIDs are not being used for "amp6" and it is being assigned to Sample "A001" on the basis of its template-specific primers without the benefit of a Multiplexer



Project Name:: MID_Nultiplexing_Example
Location:: /data/ampProjects/MID_Nultiplexing_Example

```
MID_Nul...
  96Plex_Both_Data
    ESS716O01
      MultiplexerBoth
        B_1_and_1
          amp1
        B_1_and_2
          amp1
        B_1_and_3
          amp1
        B_1_and_4
        B_2_and_1
        B_2_and_2
        B_2_and_3
        B_2_and_4
        B_3_and_1
        B_3_and_2
        B_3_and_3
        B_3_and_4
        B_4_and_1
        B_4_and_2
        B_4_and_3
        B_4_and_4
      MultiplexerP1
        P1_9
          amp4
        P1_10
          amp4
        P1_11
    ESS716O02
      A001
        amp6
      MultiplexerEither
        E_5_or_5
          amp2
          amp3
        E_6_or_6
          amp2
          amp3
        E_7_or_8
        E_8_or_7
      MultiplexerP2
        P2_12
          amp5
        P2_13
          amp5
        P2_14
```

cons (6) | Read Data (2) | Samples (27) | Variants | MIDs (14) | Multiplexers (4)

Name ▲	Annotation	Encoding	Primer 1 MIDs	Primer 2 MIDs	Samples
MultiplexerBoth	MIDs on both ends, both required for demultiplexing.	Both	4 MIDs	4 MIDs	16 Unique Samples
MultiplexerEither	MIDs on both ends, either one sufficient for demultiplexing.	Either	4 MIDs	4 MIDs	4 Unique Samples
MultiplexerP1	MIDs only on Primer1 end.	Primer 1 MID	3 MIDs		3 Unique Samples
MultiplexerP2	MIDs only on the Primer2 end.	Primer 2 MID		3 MIDs	3 Unique Samples

Figure 11–1: The Multiplexer Definition Table and Read Data Tree for the MID example Project described in this section. Note that this Project is atypically complex, as it serves to illustrate a wide variety of MID / Multiplexer features.

The usual CLI commands can be used to set up an MID Project, using the appropriate options. For example, the 'associate' command supports the definition of both MID-based and non-MID-based multiplexing relationships. Read section 11.4.1 (or run 'help associate') for more details on how to create these multiplexing relationships. For more information on creating Multiplexers and their associated constituents using the CLI, run 'help create multiplexer' (section 11.4.4.4), 'help create mid' (section 11.4.4.2), and 'help create midGroup' (section 11.4.4.3).

To display some of these commands in context, an example CLI script is provided below. This script would produce a Project setup that would match the one shown in Figure 11–1. Just as in non-MID project setup CLI scripts (as shown in the example in section 11.5), the standard objects such as References, Amplicons, and Samples must be created, and Read Data must be imported. The script shows these entities being loaded via the '-file' option to keep the script more succinct.

The "here" file contents that are shown are tab-delimited. The double-quoting of the "here" file contents isn't a requirement, but it is used here to help make it clear visually when particular fields have been deleted. These empty fields are generally interpreted as an intent to set an appropriate empty value for a field (such as an empty string for an annotation). However, the "associate" command is unique in this regard, as empty fields are interpreted as "ignore this field for this line". This allows different types of associations to be specified within a single table (*e.g.* in the script below, the associations between Multiplexers, MIDs, and Samples for all four different types of Multiplexers are shown in the same association "here" file).

In the example script, the associations of the Multiplexers, MIDs, and Samples are split logically from the association of the Multiplexers, Read Data, and Amplicons, but they could have been combined into a single association table. This would have resulted in a larger, more complicated table with more repetition of fields across lines, which would be more difficult to create error-free by manual typing. However, the large association table would be convenient when creating the setup script via programmatic means (such as using Perl scripts to construct the commands by consulting a database or spreadsheet of the experimental design).

This script also illustrates the use of the new "utility execute" command to load the "454Standard" MID Group via an existing default script called "create454StandardMIDs.ava", which is also used as part of the automatic project initialization functionality. Documentation of the "utility execute" command is available in section 11.4.17.5 and more information on automatic project initialization can be found in sections 12.4 and 12.5.

Due to the limitations of this printed document, certain lines of the script below appear on multiple lines. This occurs for certain tab-separated entries in the tables given as arguments to certain commands. Be aware that these should actually be single lines in the script.

11.6.1 Example MID Project Script

```
# Create the project
create project /data/ampProjects/MID_CLI_Example \
    -name MID_CLI_Example \
    -annotation “”

# Load the references from a tab-delimited file
# containing data lines for each reference
# following  the format of the header below
# (which should be included at the top of the file):
#
# “Name”  “Annotation”    “Sequence”
#
# For this example, ref1-ref6 should be defined.

create reference -file referencesFile.txt

# Load the amplicons from a tab-delimited file
# containing data lines for each amplicon
# following  the format of the header below
# (which should be included at the top of the file):
#
# “Name”  “Annotation”    “Reference”   “Primer1”   “Primer2” “Start”    “End”
#
# For this example, amp1-amp6 should be defined
# (where amp1 is from ref1, amp2 is from ref2, etc.).

create amplicon -file ampliconsFile.txt

# For this example, the following samples need to be created.

create sample -file - << HERE_TERMINATOR
“Name”
“A001”
“B_1_and_1”
“B_1_and_2”
“B_1_and_3”
“B_1_and_4”
“B_2_and_1”
“B_2_and_2”
“B_2_and_3”
“B_2_and_4”
“B_3_and_1”
“B_3_and_2”
“B_3_and_3”
“B_3_and_4”
“B_4_and_1”
“B_4_and_2”
“B_4_and_3”
“B_4_and_4”
“E_5_or_5”
“E_6_or_6”
“E_7_or_8”
“E_8_or_7”
“P1_9”
“P1_10”
“P1_11”
“P2_12”
“P2_13”
“P2_14”
HERE_TERMINATOR

# Create a readGroup for the readData.
create readGroup -name “96Plex_Both_Data”


# Load the readData from a tab-delimited file
# containing data lines for each readData set
# following  the format of the header below
# (which should be included at the top of the file):
#
# “SffDir”   “SffName”    “ReadGroup”   “SymLink”   “Name”
#
# For this example, two sff files named
# ESS716O01 and ESS716O02 are being loaded.

load -file readDataFile.txt

# Use the utility execute command to run the
# default script that loads a project with
# the 454Standard group of 14 MIDs.

utility execute %libDir/create454StandardMIDs.ava
```

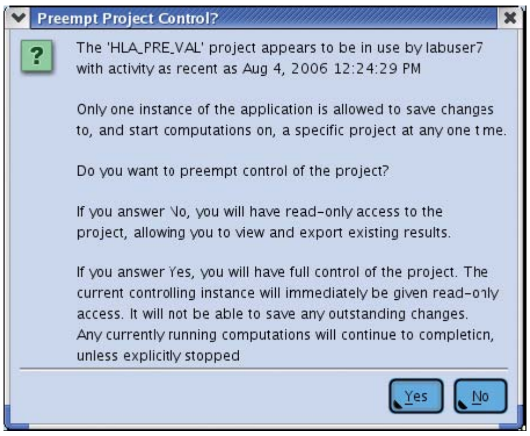
```
# For the sake of demonstrating functionality,
# this example assumes that you want to replace
# Mid9-Mid14 from the 454Standard group with
# your own custom set of 6 new MIDs.

# To simplify project, optionally remove the
# the 454Standard MIDs that are being replaced.
# Note: specifying the OfMidGroup option isn’t
# technically necessary as the MID Names are unique
# in the project.

remove mid -file - << HERE_TERMINATOR
“Name”    “OfMidGroup”
“Mid9”    “454Standard”
“Mid10”   “454Standard”
“Mid11”   “454Standard”
“Mid12”   “454Standard”
“Mid13”   “454Standard”
“Mid14”   “454Standard”
HERE_TERMINATOR

# Create a midGroup for the new MIDs.

create midGroup -name “CustomMids”

# Load the custom MID sequences from a tab-delimited file
# containing data lines for each MID
# following  the format of the header below
# (which should be included at the top of the file):
#
# “Name”  “Annotation”   “Sequence”“MidGroup”
#
# For this example, 6 MIDs are being defined
# with the names CMid9-CMid14.

create mid -file customMidFile.txt

# Create the four different multiplexers
# being used in the project.

create multiplexer -file - << HERE_TERMINATOR
“Name”    “Annotation”   “Encoding”
“MultiplexerBoth”    “”  “both”
“MultiplexerEither”  “”  “either”
“MultiplexerP1”      “”  “primer1”
“MultiplexerP2”      “”  “primer2”
HERE_TERMINATOR

# Set up the association of MIDs to samples
# in each of the multiplexers.
# Note: specifying the OfPrimer1MidGroup and OfPrimer2MidGroup
# options isn’t technically necessary as the MID Names are unique
# in the project.

assoc -file - << HERE_TERMINATOR
“Multiplexer”        “Primer1Mid”   “OfPrimer1MidGroup”  “Primer2Mid”
“OfPrimer2MidGroup”  “Sample”
“MultiplexerBoth”    “Mid1”     “454Standard”  “Mid1”   “454Standard”  “B_1_and_1”
“MultiplexerBoth”    “Mid1”     “454Standard”  “Mid2”   “454Standard”  “B_1_and_2”
“MultiplexerBoth”    “Mid1”     “454Standard”  “Mid3”   “454Standard”  “B_1_and_3”
“MultiplexerBoth”    “Mid1”     “454Standard”  “Mid4”   “454Standard”  “B_1_and_4”
“MultiplexerBoth”    “Mid2”     “454Standard”  “Mid1”   “454Standard”  “B_2_and_1”
“MultiplexerBoth”    “Mid2”     “454Standard”  “Mid2”   “454Standard”  “B_2_and_2”
“MultiplexerBoth”    “Mid2”     “454Standard”  “Mid3”   “454Standard”  “B_2_and_3”
“MultiplexerBoth”    “Mid2”     “454Standard”  “Mid4”   “454Standard”  “B_2_and_4”
“MultiplexerBoth”    “Mid3”     “454Standard”  “Mid1”   “454Standard”  “B_3_and_1”
“MultiplexerBoth”    “Mid3”     “454Standard”  “Mid2”   “454Standard”  “B_3_and_2”
“MultiplexerBoth”    “Mid3”     “454Standard”  “Mid3”   “454Standard”  “B_3_and_3”
“MultiplexerBoth”    “Mid3”     “454Standard”  “Mid4”   “454Standard”  “B_3_and_4”
“MultiplexerBoth”    “Mid4”     “454Standard”  “Mid1”   “454Standard”  “B_4_and_1”
“MultiplexerBoth”    “Mid4”     “454Standard”  “Mid2”   “454Standard”  “B_4_and_2”
“MultiplexerBoth”    “Mid4”     “454Standard”  “Mid3”   “454Standard”  “B_4_and_3”
“MultiplexerBoth”    “Mid4”     “454Standard”  “Mid4”   “454Standard”  “B_4_and_4”
“MultiplexerEither”  “Mid5”     “454Standard”  “Mid5”   “454Standard”  “E_5_or_5”
“MultiplexerEither”  “Mid6”     “454Standard”  “Mid6”   “454Standard”  “E_6_or_6”
“MultiplexerEither”  “Mid7”     “454Standard”  “Mid8”   “454Standard”  “E_7_or_8”
“MultiplexerEither”  “Mid8”     “454Standard”  “Mid7”   “454Standard”  “E_8_or_7”
“MultiplexerP1”      “CMid9”    “CustomMids”   “”  “”   “P1_9”
“MultiplexerP1”      “CMid10”   “CustomMids”   “”  “”   “P1_10”
“MultiplexerP1”      “CMid11”   “CustomMids”   “”  “”   “P1_11”
“MultiplexerP2”      “”  “”     “CMid12”       “CustomMids” “P2_12”
“MultiplexerP2”      “”  “”     “CMid13”       “CustomMids” “P2_13”
“MultiplexerP2”      “”  “”     “CMid14”       “CustomMids” “P2_14”
HERE_TERMINATOR
```

```
# Associate the non-MID sample directly
# with its amplicon and read data.

assoc -readData ESS716O02 -sample “A001” -amplicon “amp6” -ofRef “ref6”

# Associate the multiplexers with
# their read data and amplicons.

assoc -file - << HERE_TERMINATOR
“Multiplexer”        “readData” “amplicon”     “ofRef”
“MultiplexerBoth”    “ESS716O01”   “amp1”      “ref1”
“MultiplexerP1”      “ESS716O01”   “amp4”      “ref4”
“MultiplexerEither”  “ESS716O02”   “amp2”      “ref2”
“MultiplexerEither”  “ESS716O02”   “amp3”      “ref3”
“MultiplexerP2”      “ESS716O02”   “amp5”      “ref5”
HERE_TERMINATOR

save
```

12. Amplicon Variant Analyzer – Special Topics

12.1 Addressing Simultaneous Multiple Users' Access to an Amplicon Project

Only one instance of the GS Amplicon Variant Analyzer can be "in control" of a given Amplicon Project at any given time, *i.e.* be able to save changes or carry out/stop computations in the Project. This is important because if multiple users or instances of the software had the same project open simultaneously and each were used to edit the project, saving from either instance would overwrite the changes of the other. To help minimize this risk, the AVA software presents a message window at the time a project is opened, if it appears to be in use by another user or another instance of the software (Figure 12–1).



The 'HLA_PRE_VAL' project appears to be in use by labuser7 with activity as recent as Aug 4, 2006 12:24:29 PM

Only one instance of the application is allowed to save changes to, and start computations on, a specific project at any one time.

Do you want to preempt control of the project?

If you answer No, you will have read-only access to the project, allowing you to view and export existing results.

If you answer Yes, you will have full control of the project. The current controlling instance will immediately be given read-only access. It will not be able to save any outstanding changes. Any currently running computations will continue to completion, unless explicitly stopped

Figure 12–1: Message window alerting you that the Amplicon Project you are trying to open is already open in another instance of the AVA software. This message gives you the choice of opening the Project as Read-Only or to preempt control of the Project from the other instance.

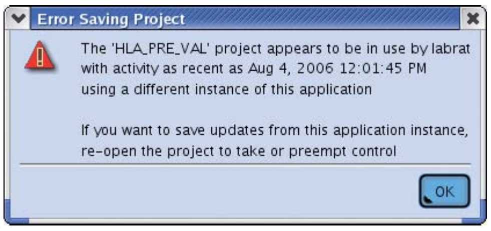
This message provides you with:

- the logon name of the other user
- the approximate time at which that user last interacted with the Project (such as by saving an update or running a computation)
- the choice to load the Project
  - but to operate in a "read-only" mode, or
  - to preempt control of the Project from the other user.

The goal of this message is to inform users of a potential conflict, not to enforce any policy with respect to these conflicts. Thus, although it is possible to preempt control from the other AVA instance, it would likely be better to first contact the indicated user to see if they are still working with the Project, or if they might have simply neglected to shutdown the application. Another possibility is that the other instance may have been terminated with a control-C or was otherwise unable to carry out a clean shutdown. In such a case it would merely appear that another instance is actively using the Project, when none actually is. If the other user cannot readily be contacted, the last activity indicator of the message may help you determine if it is safe to preempt control of the Project without unduly affecting the other user.

If you open the Project in Read-Only mode, you will be able to make changes internally to the application (define new Reference Sequences, *etc.*) and use all the features of the Variants, Global Align, Consensus Align, and Flowgrams Tabs, including the export of pre-existing results (.png snapshots, .xls spreadsheets, or .ace files). However, you will not be able to save any changes to disk, or to use them in new calculations (which also involves writing the results to disk): the Save button will be grayed-out, which constitutes the only visual clue to the Read-Only status of the Project. Features like defining new Variants from selections on the Global and Consensus Align Tabs, though available, would be of little use since you would not be able to save the newly defined Variants to the disk or obtain frequency calculations for them in the Variants tab. On the other hand, you would be able to observe (but not stop) a new computation of the Project if carried out by the currently controlling instance. If you have made changes to the Project and another user preempts control, the Save button may temporarily remain active (not grayed-out). If you then click on the Save button, you will be alerted to the transition to Read-Only mode, but you will not be able to save your changes. Clicking on either the Save, the Start computation or the Stop computation button, when in Read-Only mode, produces the following warning messages (Figure 12–2).

A

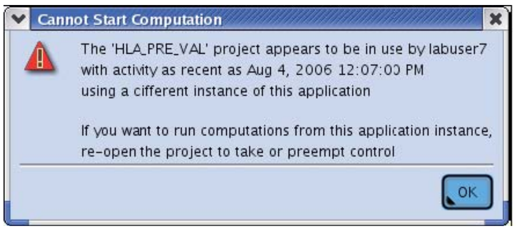


Error Saving Project

The 'HLA_PRE_VAL' project appears to be in use by labrat
with activity as recent as Aug 4, 2006 12:01:45 PM
using a different instance of this application

If you want to save updates from this application instance,
re-open the project to take or preempt control

B

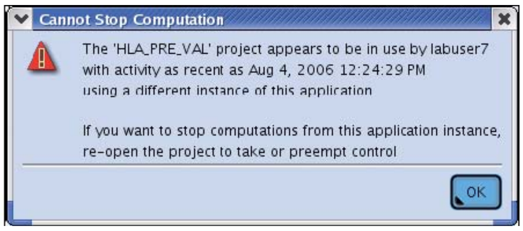


Cannot Start Computation

The 'HLA_PRE_VAL' project appears to be in use by labuser7
with activity as recent as Aug 4, 2006 12:07:03 PM
using a cifferent instance of this application

If you want to run computations from this application instance,
re-open the project to take or preempt control

C



Cannot Stop Computation

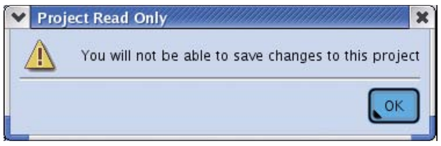
The 'HLA_PRE_VAL' project appears to be in use by labuser7
with activity as recent as Aug 4, 2006 12:24:29 PM
using a different instance of this application

If you want to stop computations from this application instance,
re-open the project to take or preempt control

Figure 12–2: Messages indicating that you do not have "control" of the Amplicon Project (*i.e.* you are operating in Read-Only mode) and that you cannot (A) Save the changes to a Project, or (B) Start or (C) Stop computations in the Project.

If you preempt control of the Project from another user, either at this point or when you first opened the Project (see Figure 12–1), the other user will be automatically and *transparently* transitioned to Read-Only mode. Not only will any of their unsaved changes become irremediably unsaveable (even trying to preempt control back from you would involve exiting the Project and thus the loss of unsaved changes), but no message will be sent to inform the other user that this transition to Read-Only occurred. The only visual clue to this state is that the Save button will remain grayed-out even after changes are made. The most obvious clue will be the reception of one of the messages of Figure 12–2, if the other ("preempted") user attempts to save Project changes or to start new computations (or stop your computation). Re-opening the Project would elicit the message of Figure 12–1, which identifies the person who currently has control over the Project.

In a related feature, if you attempt to open an Amplicon Project in a file-system on which you do not have writing permissions, the software displays the message shown in Figure 12–3, alerting you that the Project will open as Read-Only (assuming that you can actually read the files in that area of the file-system). Although the message specifies only the "Save" restriction, the computation restrictions apply as well.


Project Read Only

You will not be able to save changes to this project

Figure 12–3: Alert message indication that the Amplicon Project you are trying to open will open in Read-Only mode

12.2 Intelligent Variant Naming

As part of its computation, the AVA software identifies certain differences between the Consensi or Reads corresponding to each Amplicon and their cognate Reference Sequences, and proposes them as possible Variants (see section 9.4). Depending on the size and complexity of the Project (and on whether or not the sequences you are working with are hypervariable), the Project could end up containing hundreds of thousands of potential Variants. Because the number of Variants can be so high, the AVA software features an automatic "intelligent" process for assigning meaningful names to the Variants, The goal is to generate Variant names that are unique for a given Reference Sequence, are suitable for sorting, and are as informative as possible without being so long as to become unwieldy (above 25 characters).

The 4-tier naming convention described below is applied to the Auto-Detected Variants as well as to the Variants proposed manually by users via the "Declare project variant from current selections" functionality on the Global Align and Consensus Align tabs (see sections 9.6.3.3 and 9.7.3).

12.2.1 Tier 1 Naming

Tier 1 names are preferred, and come in two forms. The first form has the prototype "Position:From-Sequence/To-Sequence" and the second has the prototype "PositionA-PositionB:From-Sequence/To-Sequence". Complicated Variants may be described using multiple names of either form, separated by commas. This is the most precise naming scheme as it explicitly specifies each base requirement that defines the Variant. Also, starting the name with the base position (relative to the Reference Sequence), is convenient for sorting a Table of Variants.

Table 12–1, below, shows some examples of Tier 1 names and what they mean.

Tier 1 Variant Name	Interpretation of the Variant Name
10:A/C	position 10 has a change from an A to a C
10-12:ACT/ATG	position 10 must match the Reference Sequence (and remain an A), while positions 11 and 12 change from a CT to a TG
10-12:ACT/ANG	position 10 must remain an A, position 11 can be any base, and position 12 has a change from a T to a G
10-12:ACT/---	the bases ACT at positions 10-12 are deleted
10.5:-/A	an A is inserted between positions 10 and 11
10-11:A-T/ACT	a C is inserted between A and T, at positions 10 and 11; the A and the T must remain unchanged
10:A/C,45:T/G	haplotype change including an A changed to a C at position 10 and a T changed to a G at position 45

Table 12–1: Examples of Tier 1 "intelligent" Variant names

12.2.2 Tier 2 Naming

If the attempt to explicitly specify all the base changes with a Tier 1 name produces an identifier that is longer than 25 characters, Tier 2 naming takes over. Tier 2 names also come in two forms. The first form has the prototype “Position:Modifier[(count/value)]” and the second has the prototype “PositionA-PositionB:Modifier(count/value)”. The modifiers are REF (must match the Reference Sequence), DEL (Deletion), INS (Insertion), and SUB (Substitution). For the first prototype, the “count/value” is optional for single base matches (REF) or single base deletions (DEL). This naming scheme is often more compact than Tier 1 names, especially when stretches of bases can be collapsed into a base count (second prototype), yet it maintains the sorting advantage by starting names with the base position (relative to the Reference Sequence. However, exact base changes are not always stated explicitly.

Table 12–2 shows examples that demonstrate the basics of Tier 2 naming.

Tier 2 Variant Name	Interpretation of the Variant Name
10:REF	base at position 10 must match the reference sequence
10-49:REF(40)	the 40 bases from 10-49 must match the reference sequence
10:DEL	base at position 10 is deleted
10-49:DEL(40)	the 40 bases from 10-49 are deleted
10.5:INS(ACG)	the bases ACG are inserted between positions 10 and 11
10:SUB(G)	base at position 10 has been changed to a G
10:SUB(C),45:SUB(G)	haplotype change at positions 10 (changed to a C) and 45 (changed to a G)

Table 12–2: Examples of Tier 2 “intelligent” Variant names. Note that some of these (like “10:REF”) would not be used because their Tier 1 equivalent would be preferred. These are shown for illustrative purposes.

12.2.3 Tier 3 Naming

If the first and second tier attempts both produce names longer than 25 characters, the naming scheme resorts to Tier 3 naming, using the literal “pattern” used when defining a Variant manually (*i.e.* using the Variant Definition Syntax described in section 9.3.2.5.2). These are patterns such as d(10-50), s(10,C), or m(10-50)s(51,C)m(52-80). The Variant Definition Syntax can be more compact than the Tier 2 naming scheme because it uses single letter abbreviations for the change types (m, d, i, s), as opposed to the 3-letter abbreviations seen above (REF, DEL, INS, SUB). Also, Tier 3 naming does not specify the lengths of matches and deletions, and it concatenates haplotype codes without any separating characters like the comma used in Tier 2. However, these names are less convenient to sort because they start with an abbreviated change type rather than a Reference position.

12.2.4 Tier 4 Naming

If a Variant can not be assigned a name that is 25 characters or less using any of the first three tier naming schemes, the software resorts to a generic but unique name following the prototype "Var_number". These default Variant names resemble those created from scratch in the Variant Definition Table of the Variants sub-tab, on the Project tab (see section 9.3.2.5.2): when a new Variant is created in that table using the "Add" function, it initially has no Reference and no pattern assigned to it, so there is no useful information with which to construct a meaningful default name. The software then creates a generic name by obtaining a unique number from a counter and appending it to the prefix "Var_".

12.2.5 Naming Example

Table 12–3 shows how 4 different but related Variant Patterns end up being named by the naming scheme, showing an example of each Tier.

Final Naming Tier	Variant Pattern	Final Name
Tier 1	d(327)m(339-342)	327:A/-,339-342:AAGC/AAGC
Tier 2	d(327)m(339-343)	327:DEL,339-343:REF(5)
Tier 3	d(327-328)m(339-343)	d(327-328)m(339-343)
Tier 4	d(327-328)m(339-343)m(347)	Var_16

Table 12–3: Examples of final Variant names that could be used, for each of the 4 tiers schemes, based on a set of Variant patterns of increasing complexity. See text below for more details.

The Tier 1 example shows that the Variant pattern can be expressed as a name exactly 25 characters in length. Since this meets the length constraint, the Tier 1 name is used as this Variant's final name.

In the Tier 2 example, the Variant pattern from the Tier 1 example has been altered to extend the match range by an extra base. If this pattern were converted into a Tier 1 name, it would read: "327:A/-,339-343:AAGCA/AAGCA" (assuming base 343 of the Reference Sequence were an A). This name exceeds the 25 character limit by two characters, so the software rejects it and constructs a Tier 2 name. The Tier 2 final name, "327:DEL,339-343: REF(5)", has 22 characters, so it is adopted as the final name for this Variant.

The Tier 3 example Variant pattern is the same as the Tier 2 pattern except that it has an extra base in its deletion. If this pattern were expressed as a Tier 2 name, it would read: "327-328:DEL,339-343:REF(5)". This name has 26 characters so the software rejects it and constructs a Tier 3 name, using the Variant Definition Syntax: "d(327-328)m(339-343)". Since the Tier 3 name is only 20 characters it is adopted as the final name for the Variant.

In the Tier 4 example, the Variant pattern from the Tier 3 example is altered by the addition of an extra match constraint. Since Tier 3 names are the same as the Variant Pattern, and the pattern here already exceeds 25 characters (see Table 12–3), the software resorts to the final tier, and the generic "Var_16" is used as the final name.

12.3 Properties Windows for Global and Consensus Alignments

As described in sections 9.6.3.2 and 9.7.3, above, right-clicking on a nucleotide in the alignment on the Global Align or the Consensus Align tab opens a context-sensitive menu that includes a "properties" option. Selecting this option opens a "properties" window containing sequence-specific information for the Consensus or read on which you right-clicked. The specific information that is provided in this window depends on whether the sequence is a Consensus, a forward read or a reverse read.

12.3.1 When is the Properties Information Useful?

The multi-alignment views of the Global and Consensus Align tabs display only the non-Primer portions of the aligned reads. In certain situations, however, it may be useful to be able to examine the sequences flanking the aligned sequences (or indeed, the whole original sequencing read). For example, if a read appears to be aligned so poorly that you think it might be a contaminant, it might be useful to examine the part of the read that was considered as a Primer match by the alignment software, to determine if some mispriming might have occurred. Conversely, if a read appears to terminate too early in the alignment, one might want to check if the sequence is really that short, or if it was truncated by some unforeseen problem with the aligner, or if it was due to sequence quality issues. Alternatively, if a read exhibits a large deletion at the edge of an alignment, it would be useful to see the sequence from the read beyond the edge of the alignment, to determine if there is a significant match that supports the deletion or if the aligner arbitrarily placed some trailing bases far from their adjacent bases in the read when they actually could have been aligned closer.

The "properties" window of a Consensus or a read provides the necessary information for such verifications, *e.g.* the full original sequence read, as well as the aligned and flanking sub-regions of the read (see details below, section 12.3.2). These are all provided in FASTA format, which can be copied to the clipboard and used in external search or analysis programs. In particular, one could BLAST a sequence to determine its identity if it is either aligned so poorly that it looks like a contaminant; or if it has such specific variation compared to the Reference Sequence that it looks like it might be a homolog or a paralog of the intended Amplicon rather than a regular Variant of the Amplicon. One can also compare a sequence to dbSNP to see if a particular Variant has already been identified in the literature.

12.3.2 Content of the Three "Properties" Window Types

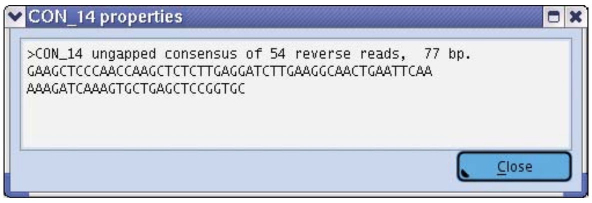
The "properties" windows for each of the sequence types (Consensus, forward Read, and reverse Read) each have their own specific content, all displayed is one or more FASTA sequences.

12.3.2.1 Properties Window for a Consensus

The Consensus properties window (Figure 12–4) simply displays a FASTA version of the Consensus sequence displayed in the alignment, minus any gaps. Since the Consensus is formed from the alignment of many trimmed reads, there is no flanking sequence to report. The definition line of the sequence is annotated to provide the number and orientation of the reads that went into constructing the Consensus, and the length of the sequence.

Although the lack of flanking information precludes this sequence from being used to troubleshoot certain issues, it is perfectly suited for use in a dbSNP search. If you choose one of the Consensi with the most member reads from your alignment (which is likely to be less noisy than a Consensus with fewer members), you can copy it to the clipboard and use it in a dbSNP search to see if your Variant is novel or not.

The Consensus properties window is only accessible on the Global Align tab when "Read Type" is set to "Consensus" (see section 9.6.3.2).



```
>CON_14 ungapped consensus of 54 reverse reads,  77 bp.
GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGCAACTGAATTCAA
AAAGATCAAAGTGCTGAGCTCCGGTGC
```

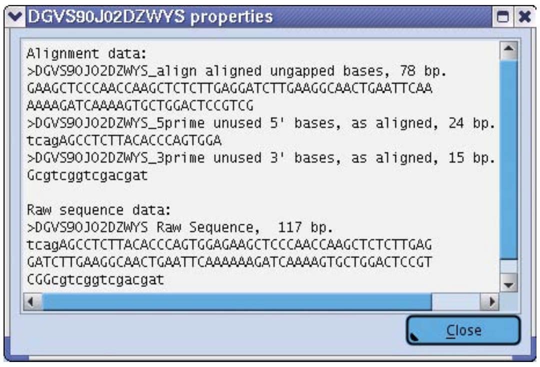
Figure 12–4: The Consensus properties window with the FASTA sequence of a Consensus and its annotated definition line

12.3.2.2 Properties Window for a Forward Read

The properties window of a forward read (Figure 12–5) contains up to 4 FASTA sequences. The window first displays a block of sequences based on the alignment data: the aligned portion of the Read, the unused 5'-flanking sequence, and the unused 3'-flanking sequence are provided as three separate FASTA sequences. Finally, the window displays the FASTA sequence of the entire read, obtained from the Read Data sff file. Note that the sequences can have mixed case characters: the lower case characters are used to represent the sequencing key and low quality read regions.

The flanking sequence information, along with an indication of where the sequence quality might be trailing off, can be used to troubleshoot alignment issues. The sequences are also available for copying so they can be used to query external databases.

The properties window for individual reads is accessible from the Consensus Align tab (section 9.7.3), as well as from the Global Align tab when "Read Type" is set to "Individual" (see section 9.6.3.2).

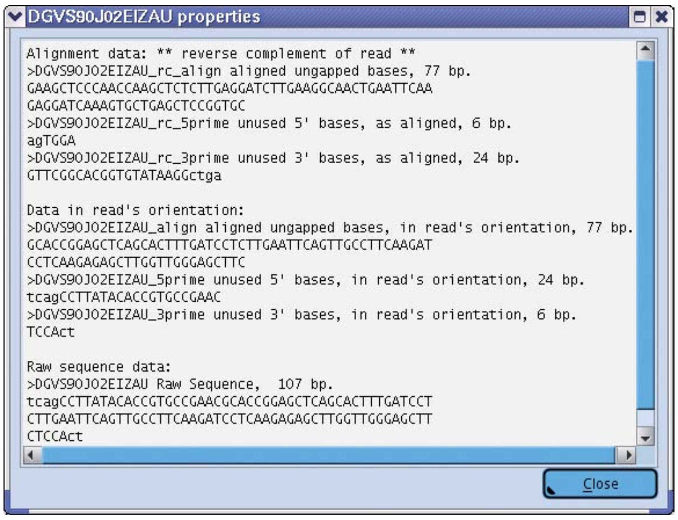


```
Alignment data:
>DGVS90J02DZWYS_align aligned ungapped bases, 78 bp.
GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGCAACTGAATTCAA
AAAAGATCAAAAGTGCTGGACTCCGTCG
>DGVS90J02DZWYS_5prime unused 5' bases, as aligned, 24 bp.
tcagAGCCTCTTACACCCAGTGGA
>DGVS90J02DZWYS_3prime unused 3' bases, as aligned, 15 bp.
Gcgtcggtcgacgat

Raw sequence data:
>DGVS90J02DZWYS Raw Sequence,  117 bp.
tcagAGCCTCTTACACCCAGTGGAGAAGCTCCCAACCAAGCTCTCTTGAG
GATCTTGAAGGCAACTGAATTCAAAAAAGATCAAAAGTGCTGGACTCCGT
CGGcgtcggtcgacgat
```

Figure 12–5: The forward read properties window with FASTA sequences showing the aligned portion of the read, the unused flanking sequences, and the full raw sequence from the Read Data file. Low quality stretches of bases and the sequencing key are denoted in lowercase letters in the sequences.

12.3.2.3 Properties Window for a Reverse Read

The properties window for a reverse read (Figure 12–6) has the same types of information and utility outlined for the forward read properties window, above (section 12.3.2.2) except that, as a convenience, the alignment data is presented in two blocks: in the first block, the sequences are presented as the reverse complements of the actual read data (thus the "_rc_" portion of the FASTA identifiers), so they can be easily related to the sequences you will see in the alignments, on the Global Align or Consensus Align tabs. The second alignment data block shows the same data (the aligned read and the flanking unused sequences) in the orientation of the read as it was sequenced (in the same orientation as the raw sequence from the Read Data file).



```
Alignment data: ** reverse complement of read **
>DGVS90J02EIZAU_rc_align aligned ungapped bases, 77 bp.
GAAGCTCCCAACCAAGCTCTCTTGAGGATCTTGAAGGCAACTGAATTCAA
GAGGATCAAAGTGCTGAGCTCCGGTGC
>DGVS90J02EIZAU_rc_5prime unused 5' bases, as aligned, 6 bp.
agTGGA
>DGVS90J02EIZAU_rc_3prime unused 3' bases, as aligned, 24 bp.
GTTCGGCACGGTGTATAAGGctga

Data in read's orientation:
>DGVS90J02EIZAU_align aligned ungapped bases, in read's orientation, 77 bp.
GCACCGGAGCTCAGCACTTTGATCCTCTTGAATTCAGTTGCCTTCAAGAT
CCTCAAGAGAGCTTGGTTGGGAGCTTC
>DGVS90J02EIZAU_5prime unused 5' bases, in read's orientation, 24 bp.
tcagCCTTATACACCGTGCCGAAC
>DGVS90J02EIZAU_3prime unused 3' bases, in read's orientation, 6 bp.
TCCAct

Raw sequence data:
>DGVS90J02EIZAU Raw Sequence,  107 bp.
tcagCCTTATACACCGTGCCGAACGCACCGGAGCTCAGCACTTTGATCCT
CTTGAATTCAGTTGCCTTCAAGATCCTCAAGAGAGCTTGGTTGGGAGCTT
CTCCAct
```

Figure 12–6: The reverse read properties window with FASTA sequences showing the aligned portion of the read, the unused flanking sequences, and the full raw sequence from the Read Data file. The alignment data is presented in two blocks, the first is reverse-complemented and the second is in the actual orientation of the read as it was sequenced. Low quality stretches of bases and the sequencing key are denoted in lowercase letters in the sequences.

12.4 Automatic Project Initialization in the GUI

When the "New" button in the AVA GUI is used to create a Project, an initialization script containing CLI-script commands is automatically carried out. This script is created as part of the software installation and is only automatically used when creating a new Project in the GUI; it is not used when opening pre-existing Projects and it is not automatically used when creating Projects via the CLI (section 12.5).

The default initialization script that is provided with the AVA software version 2.0.00 serves two main purposes: it automatically loads the 454Standard MID Group, and it provides the ability to automatically call through to optional customized initialization scripts in the users' home directories. The functionality of the initialization script is optional, and can be disabled by an administrator by commenting out actions in the script (by placing a pound character ('#') in front of commands). However, the script should not be deleted entirely, or a 'missing' file warning will be encountered each time a new Project is created via the GUI.

12.4.1 Default Initialization Script Location

The default initialization script is located relative to the main software installation. If the software was installed to the standard location ('/opt/454'), then the initialization script will be found as the following: /opt/454/apps/amplicons/config/lib/newProjectInit.ava. More generally, the file will be located at *installDir*/apps/amplicons/config/lib/newProjectInit.ava (where installDir is replaced by the main software installation path).

12.4.2 Default Initialization Script Contents

The contents of the default initialization script are provided below.

```
# GS Amplicon Variant Analyzer CLI script used to populate new projects
#
# This script adds the “Standard” 454 MIDs to the project and
# then runs the user-specific initialization script found in the user’s
# home directory, if any.
#
# Sites may want to edit this file to further customize the initialization
# of new AVA projects.  To completely disable the actions of this file,
# comment out, or delete, all its lines:  do not delete the file itself,
# however, or else a warning message about a missing file will be displayed
# each time a new project is created from the gsAmplicon application GUI.



# Step 1: Populate with the “Standard” 454 MIDs
#         (To prevent this, comment out or delete the following line:)
utility execute create454StandardMIDs.ava

# Step 2: Run whatever “new project” scripts are in the
#         user’s home directory.
#         (Note, due to the use of -onMissingScript with the value ‘ignore’,
#          it is not an error if the user has no such script)
utility execute -onMissingScript ignore %homeDir/.gsAmplicon_newProjectInit.ava
```

The script mainly contains comments (lines beginning with ‘#’) and blank lines and only contains 2 actual commands. Both of the commands are ‘utility execute’ commands which run other scripts.

12.4.2.1 Step 1: Loading the “Standard” 454 MIDs

Fourteen MIDs were carefully selected four use with Amplicon libraries, in the Genome Sequencer System (section 9.3.2.6). As part of the default initialization process, these MIDs are automatically loaded into the Project as a convenience and as a means of reducing data entry error. The command ‘`utility execute create454StandardMIDs.ava`’ runs a default script called ‘`create454StandardMIDs.ava`’ that is located in the same directory as the default initialization script (‘*installDir*/apps/amplicons/config/lib’). The script creates Amplicon MIDs Mid1-Mid14 in the Project.

12.4.2.2 Step 2: Running User-Customized Initialization Functions

Since the default initialization script is part of the main software installation, the average user probably will not have permissions to edit the script. To enable users to customize the automated project initialization, the default script calls through to a script in the user’s home directory called ‘.gsAmplicon_newProjectInit.ava’ by using the command ‘`utility execute  onMissingScript ignore %homeDir/.gsAmplicon_newProjectInit.ava`’. Thus, users can create a script called ‘.gsAmplicon_newProjectInit.ava’ in their home directory, and fill it with CLI-commands to add extra functions to the initialization process. (Note that the file name intentionally begins with a ‘dot’, which makes the file invisible to standard listings of the user’s home directory.) The customized user script is entirely optional, however; no errors or warnings will be issued if the user does not create one.

The name of the user-customized script is dictated by the command issued in the default initialization script. If the administrator changes the name of the called script by editing the default initialization script *e.g.* so that the Step 2 command is ‘`utility execute onMissingScriptignore%homeDir/.gsAmplicon_user_customization.ava`’, then the corresponding user-customized script would have to be named ‘`.gsAmplicon_user_customization.ava`’.

12.4.3 Initialization Script Restrictions

Although the default initialization script can be edited and custom scripts can be nested within it (as illustrated by the call to the user's home directory script), the default initialization script and any of the subordinate scripts are precluded from utilizing certain specific CLI commands. The excluded commands are:

- open
- close
- exit
- create project
- any of the computation commands, such as computation start and computation stop

The reason for these exclusions is that automatic initialization is intended to work in the context of creating a new Project and leaving it in a state where more CLI commands can be issued. The banned commands would not make any sense in this context because they cause a switch to other Projects, shut down the Project, or attempt to prematurely compute the Project.

12.4.4 Initialization Script Error Handling

The automatic initialization script and any other scripts called within it are controlled by an error handler that reports problems encountered when running the scripts. Those errors, however, do not prevent the successful creation of a Project via the 'New' button in the GUI. So, errors might cause portions of an initialization to fail, but the new Project will be accessible. This is intended to prevent mistakes in editing of the default initialization file from locking users out of creating new Projects via the GUI. As mentioned earlier, a missing default initialization script will cause a warning to be issued, but a missing user-customized initialization file will be ignored.

12.5 Project Initialization and the CLI

When using the CLI, the 'create project' command is used to set up new Projects. This process is entirely under user control, and there is no attempt to automatically initialize the Project as is done in the GUI when using the 'New' button (section 12.4). Avoiding automatic initialization in the CLI maintains backward compatibility with pre-existing CLI scripts created for use with prior software versions.

However, this doesn't prevent a user from taking advantage of the default initialization script if desired. The script can be incorporated into CLI Project setup by using the 'utility execute' command. To take advantage of the default initialization script, use the command:

```
utility execute %libDir/newProjectInit.ava
```

If the only functionality that the user wants to borrow from the initialization is loading the '454Standard' MIDs (perhaps to add MIDs to a pre-existing Project), the user can directly call the script that the initialization script uses. To load the fourteen '454Standard' Amplicon MIDs to a Project, as shown in the example CLI script in section. 11.6, use the command:

```
utility execute %libDir/create454StandardMIDs.ava
```

12.6 Multiplex Amplicon Libraries

Because Amplicon libraries are not supported with the GS FLX Titanium chemistry, the *GS FLX Amplicon DNA Library Preparation Method Manual* is not being updated as part of the launch of the new chemistry. In the absence of such an update, this brief section is provided to illustrate the specifics of the structure of an MID-based library for Amplicon sequencing.

Multiplex Amplicon libraries are prepared the same way as standard Amplicon libraries (see the *GS FLX Amplicon DNA Library Preparation Method Manual*), with the exception that short Multiplex Identifier sequences, the MIDs, are added to the design of the Adaptors. Since the AVA software expects these sequences at the very beginning of the reads, the MID sequences must be positioned at the end of the Primer A and/or Primer B segments of the Adaptors, just past the sequencing key, and before the template-specific primers (see Figure 12–7).

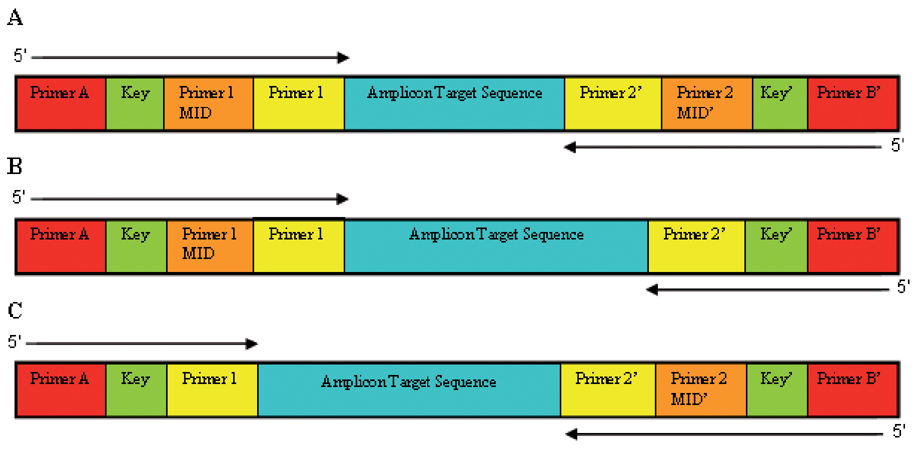


Figure 12–7: Diagrams of potential Amplicon structures using MIDs

(A) MIDs are inserted between the sequencing key and the sequence-specific primers (Primer 1 and Primer 2) on both ends of the Amplicon. A Multiplexer object describing Amplicons with this structure could have an encoding type of "both" (if unambiguous Sample assignment requires both MIDs be found), or "either" (if unambiguous Sample assignment can be made with the MID from either end independently).

(B) An MID is inserted between the sequencing key and the Primer 1 sequence-specific primer only, with no MID used with Primer 2. A Multiplexer object describing Amplicons with this structure must have an encoding type of "Primer 1 MID".

(C) An MID is inserted between the sequencing key and the Primer 2 sequence-specific primer only, with no MID used with Primer 1. A Multiplexer object describing Amplicons with this structure must have an encoding type of "Primer 2 MID".

An important difference between the way MIDs are used in Amplicon libraries, compared to Shotgun (sstDNA) libraries, is that Amplicon reads can carry an MID tag at each end, *i.e.* as part of both Adaptor A and Adaptor B. By contrast, "MID Adaptors" used to prepare Shotgun (sstDNA) libraries, such as the ones provided in the MID Adaptors Kits, carry an MID sequence only on Adaptor A, where the Sequencing Primer of the emPCR Kit II binds. (Note that the MID kits are not used to prepare Amplicon libraries, since the Adaptors for Amplicon libraries contain template-specific information and must be designed and obtained separately by the user.) The presence of MIDs at both ends of the reads in Amplicon libraries allows their use in a manner analogous to the use of the defined Primer 1 and Primer 2 in the standard (non-MID) AVA demultiplexing scheme; since Amplicons have fully defined sequences (unlike Shotgun library reads), the software knows from the experimental design exactly where the distal MID tag should be, and can thus look for it.

In addition, the possibility to use MIDs at both ends on Amplicons allows for combinatorial demultiplexing, which greatly increases the number of libraries that can be multiplexed with a given set of MIDs. For example, the 454Standard MID Group, which comprises 14 MIDs (see section 9.3.2.6), allows the multiplexing of up to 196 (14×14) separate Samples in a single PTP region (Read Data Set), when MIDs are placed at both ends.

However, as different experiments may require different amounts of multiplexing and read length considerations may make it impossible to exploit a demultiplexing scheme in which MIDs are at both ends of the read, the software allows for a number of flexible "encoding" schemes: the user can choose to tag Amplicon libraries at only one end of the reads like in Shotgun libraries (which is simpler), at both ends (to take advantage of combinatorial demultiplexing or to guarantee the ability to demultiplex forward and reverse reads from amplicons that are too large to read all the way through), or at neither end, (*i.e.* not use MIDs at all and rely on the template-specific Primer sequences to carry out the demultiplexing); encoding schemes are described in more detail in section 9.3.2.7.1.

Appendix E

Part E: Appendix

13. Data Files and Formats 508

13.1 Directory Naming Conventions 509
13.2 Format Requirements for Input FASTA Files 509
13.3 Standard File Formats 510
13.3.1 Composite wells file format 510
13.3.1.1 mimetype 512
13.3.1.2 meta.xml 512
13.3.1.3 history.xml 513
13.3.1.4 location.idx 515
13.3.1.5 metrics.xml 516
13.3.1.6 sequences.xml 518
13.3.1.7 filters.xml 519
13.3.1.8 filterResults.uint8.dat 520
13.3.1.9 Base Called Data 522
13.3.1.10 Other Public Streams 522
13.3.1.11 Other Private Streams 523
13.3.1.12 Other Files 523
13.3.2 Parameters and Viewable Metrics Files: 454 Parser Format (.parse, .txt) 524
13.3.3 Image Files (.pif) 524
13.3.4 Well-Level Signal Data Files (.wells) 525
13.3.5 Exportable Metrics Files (.csv) 525
13.3.6 DNA Sequence (FASTA; .fna) and Base Quality Score (.qual) Files 526
13.3.7 454 “Universal” Accession Numbers 527
13.3.8 Standard Flowgram Files (.sff) 528
13.3.8.1 Common Header Section 529
13.3.8.2 Read Header Section 530
13.3.8.3 Read Data Section 531
13.3.8.4 Computing Lengths and Scanning the File 531
13.4 Example Files 532
13.4.1 454QualityFilterMetrics.txt 532
13.4.2 454BaseCallerMetrics.txt 534
13.4.3 454RuntimeMetricsAll.txt 545
13.4.4 *region.key.*454Reads.fna and *region.key.*454Reads.qual 552
13.4.5 Consensus Basecalled Contig Files 554
13.4.6 454NewblerMetrics.txt File for the GS *De Novo* Assembler Application 557
13.4.7 454ReadStatus.txt 562
13.4.8 454AllDiffs.txt 564
13.4.9 454NewblerMetrics.txt File for the GS Reference Mapper Application 567
13.4.10 454NewblerProgress.txt 571
13.4.11 454AlignmentInfo.tsv 572
13.4.12 454ReadStatus.txt 572
13.4.13 454RefStatus.txt 573
13.4.14 sffinfo summary 573
13.4.15 MID Configuration File 575

13. Data Files and Formats

Section 1.3 lists all the files and folders that constitute the deliverable output of the Genome Sequencer FLX System Data Processing software for a generic sequencing Run, including the results of the GS *De Novo* Assembler and GS Reference Mapper applications. The actual directories generated may contain a number of additional files, but those are intermediate or log files generated for use only by your GS FLX support personnel, in the event that a Run might require additional investigation.

The Genome Sequencer FLX System software uses fixed names for the files it generates, and the structure and names of the directories allows to differentiate individual sequencing Runs or post-Run analyses. This section describes the nomenclature conventions and file formats used by the software, and shows samples of several file types. You can also view the output files from the sample datasets provided in the accompanying DVD, some using the GS Run Browser application. The section is divided as follows:

- Section 13.1: Directory Naming Conventions
- Section 13.2: Format Requirements for Input FASTA Files
- Section 13.3: Standard File Formats
- Section 13.4: Example Files

Note that the content of this Appendix does not apply to the GS Amplicon Variant Analyzer software, whose output structure is completely distinct (beyond the basic data processing files and folders).

13.1 Directory Naming Conventions

When a sequencing Run is performed on a Genome Sequencer FLX Instrument, its results are placed in a Run folder, where the format of the Run name is generated by the software and includes the following components:

```
R_year_month_day_hour_minute_second_instrument_user_runname
```

A similar naming convention is used for the Data Analysis folder(s) which are deposited inside the corresponding Run folder by the Run-time data processing applications (either on the Genome Sequencer FLX Instrument or on a DataRig). Data Analysis folders contain all the flow signal and signal processing files. Their names include the following components (except that "instrument" and "user" are not included when using the command line software):

```
D_year_month_day_hour_minute_second_instrument_user_analysisname
```

Finally, the GS *De Novo* Assembler and GS Reference Mapper applications create a "Post-Run Analysis" sub-directory to include all the files they generate, using the following naming convention:

```
P_year_month_day_hour_minute_second_runCommandName
```

The rest of the files within these directories (or within the current working directory) have either fixed names or simple, standard naming conventions (*e.g.* files specific to a region or key are named with the region or key at the beginning of the name). Exact nomenclature for all the files and other sub-folders produced by the data processing applications are provided in the "output" subsection of the description of each application, within sections 0 through 6.

13.2 Format Requirements for Input FASTA Files

The GS *De Novo* Assembler, the GS Reference Mapper and the fnafile command (from the SFF Tools) can all take FASTA files as input. For the reference or input read FASTA file(s) to be readable by the Genome Sequencer System software, they must follow the industry standards for a FASTA file. In particular:

- The first line (descriptor line) of each sequence entry in the file should begin with a '>'.
- There may be one or more additional header lines for a sequence entry, each beginning with a '>' or ';' character. The first line not beginning with a '>' or ';' starts the sequence region of the entry.
- The sequence region may contain any characters, but only the alphabetic characters will be used to form the sequence. All alphabetic characters are converted to uppercase, and any alphabetic character that is not A, C, G or T will be treated as an N.
- Multiple sequences may be included in the file (each starting with a '>').
- Only the characters between the '>' and the first whitespace character are used to identify the sequence (*i.e.*, the "accno" for the sequence). For clarity, each sequence in a project should be identified uniquely within the characters prior to the first whitespace in their respective descriptor lines.

For example:

```
>Ecolik12     4300K bp
CCTTGTGCAGTAGCACTTAATCATCATGTTTTAGCATTTTGATCTTCTGCTCAATTTCTT
AAGCTAGACGCTCAATCTTCTTATGATGAACGATTTCTTCTTCATGGTGTTTTTTCATAT
......
```

13.3 Standard File Formats

Most of the file formats are specific to the type of data being stored, such as the image files or the wells data files. Other files adhere to standard formats used throughout the generation and processing of sequencing Run data, assembly data and mapping data. Sample files in many of these formats are provided in section 13.4. You can also view the output files from the sample datasets provided in the accompanying DVD, some using the GS Run Browser application.

13.3.1 Composite wells file format

The CWF file is a container format which stores multiple "streams" of information. The container itself is a "ZIP" file (http://www.pkware.com/documents/casestudies/APPNOTE.TXT) with a single level hierarchy. Each stream is named and compressed separately, allowing for rapid access to any information in the file. The CWF file format is inspired from the "OpenDocument format" described in ISO/IEC 26300:2006 (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43485). The OpenDocument format benefits from the segregation of concerns by separating the content, styles, metadata and application settings into four separate XML streams; the CWF format maintains a similar separation: flowgrams, called bases, meta data, processing history and run metrics are stored individually. The user should not normally unpack the CWF file, as each file is read as needed. Nonetheless, a C library (libcwf) is available that can read the CWF file format, for convenience.

The data for each region of the PicoTiterPlate device is stored in a separate CWF file. Because this file is fully self-contained, it is the only file that needs to be moved between the instrument and the data processing system for continued analysis (after the image processing step of the GS Run Processor. As of October 2008, the CWF file format contains all the necessary information to generate any pipeline output artifact on-demand except for Standard Flowgram Format (SFF) files. Therefore, the user only needs to store one CWF file and one SFF file per region to fully archive the experiment's processed data.

Textual data in the CWF container will generally be stored as XML. Specifically, there are four main required XML streams: meta.xml, metrics.xml, history.xml, sequences.xml and one optional one, filters.xml. Each of these files references a single XML Schema (which is available on request).

Table 13–1 shows an example listing of a CWF file's streams that might exist at the end of signal processing. This file represents the data from one region of a high-quality 2-region sequencing Run, and the Table shows the size savings provided by the CWF compressed format. Each stream is described separately below.

Stream Name	Uncompressed Size	Compressed Size	Compression Ratio
mimetype	24	24	0%
rawWellDensity.pgm	8376341	1519187	82%
cfValues.double.dat	3749552	3328881	11%
ieValues.double.dat	3749552	3021477	19%
keyPassWellDensity.pgm	8376341	1103849	87%
histogram.unfiltered.ATGC.pgm	204695	24439	88%
histogram.unfiltered.TCAG.pgm	684743	40138	94%
filterResults.uint8.dat	468694	130450	72%
trimInfo.uint16.dat	937388	444081	53%
histogram.filteredCounts.ATGC.pgm	204695	22734	89%
histogram.nmers.ATGC.pgm	263879	20221	92%
histogram.filteredCounts.TCAG.pgm	684743	48430	93%
histogram.nmers.TCAG.pgm	1064508	60786	94%
0-149421.char.dna	33554213	10135865	70%
0-149421.uint_8.flow	33554213	9872137	71%
0-149421.uint_8.score	33554213	21064266	37%
149422-288120.char.dna	33554192	10070875	70%
149422-288120.uint_8.flow	33554192	9787599	71%
149422-288120.uint_8.score	33554192	20866664	38%
288121-424287.char.dna	33554389	10066101	70%
288121-424287.uint_8.flow	33554389	9787253	71%
288121-424287.uint_8.score	33554389	20863713	38%
424288-468693.char.dna	10323379	3139442	70%
424288-468693.uint_8.flow	10323379	3056298	70%
424288-468693.uint_8.score	10323379	6530831	37%
h0-41220.wel	33553894	29209536	13%
h41221-82441.wel	33553894	29206671	13%
h82442-123662.wel	33553894	29126347	13%
h123663-164883.wel	33553894	29069883	13%
h164884-206104.wel	33553894	29068170	13%
h206105-247325.wel	33553894	29082037	13%
h247326-288546.wel	33553894	29120018	13%
h288547-329767.wel	33553894	29184495	13%
h329768-370988.wel	33553894	29262051	13%
h370989-412209.wel	33553894	29361393	13%
h412210-453430.wel	33553894	29461534	12%
signalPerBase.float.dat	1874776	1539471	18%
h453431-468693.wel	12424082	10939973	12%
baseCalledSeq.dat	2812164	1126775	60%
sequences.xml	2334	612	74%
location.idx	4686940	2962553	37%
meta.xml	2162	701	68%
filters.xml	839	286	66%
metrics.xml	130111	18528	86%
history.xml	3286	1284	61%
TOTAL	**752 MB**	**482MB**	**36%**

Table 13–1:List of the streams in a CWF file following image and signal processing of a sequencing Run

13.3.1.1 mimetype

This is a single line file containing the words:

application/vnd.454.cwf

It must be the first stream in the file and must be stored uncompressed.

13.3.1.2 meta.xml

Like the OpenDocument format, meta.xml stores information about the run itself. As a convenience, the schema defining the XML data stored in the CWF file references the Dublin Core ("DC") metadata elements. The DC elements used and their interpreted meaning are summarized in Table 13–2. An example meta.xml file is shown in Figure 13–1.

DC Element	454 Usage
Title	Run name
Description	User defined
Type	"flowgrams"
Source	Serial number of instrument performing the run
Relation	Original run name. *e.g.* "R_2007_06_27_15_44_21_rig3_ccelone_1007075seqkit93555420PELTxxEX2xxVERIIF2"
Creator	Instrument operator's name
Date	"dcterms:created": Date analysis was performed
Identifier	UUID version of job

Table 13–2: Dublin Core metadata elements used in CWF files

```
<?xml version="1.0" encoding="utf-8"?>
<Metadata xmlns:tns="http://purl.org/dc/terms/"
xmlns:tnsa="http://purl.org/dc/elements/1.1/"
xmlns:tnsb="http://purl.org/dc/dcmitype/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="GSDataProcessing-1.0.xsd">
<tnsa:title>R_2007_06_27_15_44_21_rig3_ccelone_1007075seqkit93555420PELTxxEX2xxV
ERIIF2</tnsa:title>
  <tnsa:type>flowgrams</tnsa:type>
  <tnsa:creator>Chris Celone &lt;ccelone@454.com&gt;</tnsa:creator>
  <tns:created>2007-06-27T15:44:21Z</tns:created>
  <Run>
    <Name>0</Name>
    <Project>Applications I</Project>
    <Kit>LR70KIT</Kit>
    <Script>100x_TACG_70x75_LR70KIT.icl</Script>
    <RegionCount>2</RegionCount>
    <RegionLayoutName>2 Regions</RegionLayoutName>
    <PTP>
      <ID>630751</ID>
      <WellSize unit="um">50</WellSize>
      <Size unit="mm">
        <Width>70</Width>
        <Height>75</Height>
      </Size>
    </PTP>
    <Flow>
      <ActualOrder>SSSSPSSTACGT...CGPSS</ActualOrder>
      <FlowCount>403</FlowCount>
      <CycleCount>100</CycleCount>
      <FlowOrder>PTACGTACGTACGT...CGP</FlowOrder>
    </Flow>
  </Run>
  <Region>
    <Name>Region0</Name>
    <Number>1</Number>
    <TemplateBounds unit="pixel">
      <Center>
        <X>1024</X>
        <Y>2048</Y>
      </Center>
      <Dimension>
        <Width>2046</Width>
        <Height>4094</Height>
      </Dimension>
    </TemplateBounds>
    <RevisedBounds unit="pixel">
      <Center>
        <X>1024</X>
        <Y>2048</Y>
      </Center>
      <Dimension>
        <Width>2046</Width>
        <Height>4094</Height>
      </Dimension>
    </RevisedBounds>
  </Region>
  <WellCount>468698</WellCount>
</Metadata>
```

Figure 13–1: Example meta.xml stream

13.3.1.3 history.xml

This is a single XML file showing what processing has been done to these wells. It contains reference copies of the pipeline parameters that were used to create the final result, as well as processing times, dates, software revision numbers and a Universally Unique Identifier ("UUID") for each processing step. Each analysis performed on the data set adds a new "Job" element to the stream. An example history.xml file is shown in Figure 13–2.

```
<?xml version="1.0" encoding="utf-8"?>
<History xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="GSDataProcessing-1.0.xsd">
  <Job>
    <ID>edf2c3fe-24fe-11dd-a9ef-001d60a92f87</ID>
    <Name></Name>
    <ProcessingDirectoryName>
    D_2008_05_18_13_22_00_zappa_allButMetrics</ProcessingDirecto-
ryName>
    <OS>Linux</OS>
    <StartTime>2008-05-18T17:22:17Z</StartTime>
    <TotalJobSeconds>3884</TotalJobSeconds>
    <PartialJobSeconds>3884</PartialJobSeconds>
    <GsRunProcessorVersion>20080308172036</GsRunProcessorVersion>
    <Host>zappa.bw01.labrat.com</Host>
    <NumDataSetsInJob>2</NumDataSetsInJob>
    <NumProcessors>10</NumProcessors>
    <Type>allButMetrics</Type>
    <Pipeline>allButMetrics.xml</Pipeline>
    <ParamsUsed>
      <WellFinder>
        <kernelSize>21</kernelSize>
        <upsampleHighDensityPtps>true</upsampleHighDensityPtps>
        <upsampleFactor>1</upsampleFactor>
        <minPPISignal>80</minPPISignal>
        <minConsensusSignal>70</minConsensusSignal>
        <minWellSpacing>2</minWellSpacing>
        <secondSearchPass>true</secondSearchPass>
        <maskConstant>0.1054</maskConstant>
        <maskAlpha>0.6</maskAlpha>
        <maskHoleSize>2</maskHoleSize>
        <numPixelsPerWell>4</numPixelsPerWell>
        <morphologyThresholdMultiplier>
        0</morphologyThresholdMultiplier>
        <morphologyNumInARow>5</morphologyNumInARow>
      </WellFinder>
      <WellBuilder>
        <kernelSize>21</kernelSize>
        <minPPISignal>80</minPPISignal>
        <scaleFactor>1</scaleFactor>
      </WellBuilder>
      <NukeSignalStrengthBalancer>
        <computeMedianRange>41</computeMedianRange>
      </NukeSignalStrengthBalancer>
      <CafieCorrector>
        <maxAcceptableDroop>0.25</maxAcceptableDroop>
      </CafieCorrector>
      <IndividualWellScaler>
        <startMinSinglet>0.75</startMinSinglet>
        <startMaxSinglet>1.25</startMaxSinglet>
        <windowEachSide>13</windowEachSide>
        <stepSize>1</stepSize>
        <minSingletsPerWindow>3</minSingletsPerWindow>
        <useAverage>true</useAverage>
        <interpolateVacantWindows>true</interpolateVacantWindows>
        <reSeedThresholds>true</reSeedThresholds>
        <padEnd>false</padEnd>
      </IndividualWellScaler>
      <WellScreener>
        <enable>false</enable>
        <useStdDevThresholding>true</useStdDevThresholding>
      </WellScreener>
    </ParamsUsed>
  </Job>
</History>
```

Figure 13–2: Example history.xml stream

13.3.1.4 location.idx

This is a binary index to the wells files. It contains common data about each well that can be used to support a well browser-style application. Items in this stream are stored in Intel Little-Endian format (*i.e.* the rank is stored as Byte3 Byte2 Byte1 Byte0). The wells file contains one field per well, and each field is made up of the packed structure shown in Figure 13–3:

	15												*0*
1	Rank (unsigned integer)												
2													
3													
4													
5	X Coordinate (integer part)												
6												fract.	
7	Y Cordinate (integer part)												
8												fract.	
9	P	Sequence ID											
10	Reserved												

Figure 13–3: Packed structure of the location.idx file

- P is a bit showing if the well has passed filtering (1) or has been discarded (0).
- Sequence ID is an index to the table contained in sequences.xml (see below) showing which sequence (if any) is matched.
- The reserved field is for future use and should be set to all 0’s.
- The X and Y coordinates are stored with two bits of fractional information. This allows the storage of well coordinates with sub-pixel resolution in the CWF file format. Applications accessing the coordinates may simply choose to map bytes 5/6 and 7/8 to 16-bit integers and divide by 4, preserving or discarding the subpixel data as needed. Also note that all coordinates are relative to a common “0,0” location representing the upper left hand corner of the PicoTiterPlate device (the corner opposite the DataMatrix code), no matter what the region. In other words, the coordinate reflects the actual location of the well on the original PicoTiterPlate device and not the offset into the region itself.

13.3.1.5 metrics.xml

This file contains all the derived statistics created during data processing. This file can be used to create all the ancillary output files, including all the reports that were produced by the Genome Sequencer FLX System software versions anterior to 2.0.00. The data is divided into various sections in four main types.

- The first is the header section, which contains metrics that are valid across all keys and sequences.
- The next sections are the "MetricsPerKey," containing metrics that cover one key, either library or control. There will be one MetricsPerKey block per key used in the experiment.
- The next sections are the "MetricsPerSequence" blocks. The metrics for each Control DNA sequence are contained in separate blocks.
- The "Other" block is a free-form container for metrics that may be used by Roche and 454 troubleshooters to diagnose problematic sequencing Runs. These metrics can and will change between releases of software, and therefore, should not be depended upon for the assessments of Runs by users.

An additional block, the "Streams" block, acts as a manifest for data stored in auxiliary streams, inside the CWF file. Each data stream is individually tagged with a "type" identifier. The stream block contains the exact stream name, and the type of the binary data contained in the data. While the exact stream name and data type may change with future releases of the software, the "type" name will remain constant. Users implementing CWF file readers should use the streams information contained in metrics.xml file to find their data and not depend on a particular file naming convention. See the sections on "Other Streams" (13.3.1.10 and 13.3.1.11) for information on the data types and stream names.

An example metrics.xml file is shown in Figure 13–4.

```
<?xml version="1.0" encoding="utf-8"?>
<Metrics xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="GSDataProcessing-1.0.xsd">
  <RunMetrics>
    <MaxWellCount>529820</MaxWellCount>
    <RawWellCount>468698</RawWellCount>
    <SampleKeyPassWellCount>451747</SampleKeyPassWellCount>
    <ControlKeyPassWellCount>7818</ControlKeyPassWellCount>
    <ControlKeys>
      <Key>ATGC</Key>
    </ControlKeys>
    <SampleKeys>
      <Key>TCAG</Key>
    </SampleKeys>
    <Streams>
      <Stream type="rawWellDensity">
        <StreamName>rawWellDensity.pgm</StreamName>
        <DataType>image</DataType>
      </Stream>
      <Stream type="carryForwardCorrections">
        <StreamName>cfValues.double.dat</StreamName>
        <DataType>double</DataType>
      </Stream>
      <Stream type="incompleteExtensionCorrections">
        <StreamName>ieValues.double.dat</StreamName>
        <DataType>double</DataType>
      </Stream>
       <Stream type="filterResults">
        <StreamName>filterResults.uint8.dat</StreamName>
        <DataType>byte</DataType>
      </Stream>
```

▶▶▶

```
        <Stream type="signalPerBase">
          <StreamName>signalPerBase.float.dat</StreamName>
          <DataType>float</DataType>
        </Stream>
      </Streams>
      <Other>
        <NukeSignalStrengthBalancer>
          <medianOneMerA>1.09085</medianOneMerA>
          <medianOneMerT>0.909846</medianOneMerT>
          <medianOneMerG>0.898174</medianOneMerG>
          <medianOneMerC>0.894138</medianOneMerC>
        </NukeSignalStrengthBalancer>
        <BlowByCorrector>
          <droopLambda>-0.00171434</droopLambda>
          <MedianSignal>1375.71</MedianSignal>
          <MaximumSignal>5086.11</MaximumSignal>
          <MedianDensity>12</MedianDensity>
          <MinimumDensity>1</MinimumDensity>
          <MaximumDensity>19</MaximumDensity>
          <num_low_density_low_signal_wells>
          14742</num_low_density_low_signal_wells>
          <num_high_density_low_signal_wells>
          10481</num_high_density_low_signal_wells>
          <num_low_density_high_signal_wells>
          13645</num_low_density_high_signal_wells>
          <num_high_density_high_signal_wells>
          14899</num_high_density_high_signal_wells>
          <mask_averaging_used>true</mask_averaging_used>
          <FinalMask>
            <class density="high" signal="high" class="0">
              <epsilon>0.174537</epsilon>
              <beta>0.964658</beta>
            </class>
            <class density="low" signal="high" class="1">
              <epsilon>0.188713</epsilon>
              <beta>0.988837</beta>
            </class>
            <class density="high" signal="low" class="2">
              <epsilon>0.171184</epsilon>
              <beta>0.950913</beta>
            </class>
            <class density="low" signal="low" class="3">
              <epsilon>0.184087</epsilon>
              <beta>0.900547</beta>
            </class>
          </FinalMask>
        </BlowByCorrector>
        <CafieCorrector>
          <droopLambda>-0.00158241</droopLambda>
        </CafieCorrector>
        <NukeSignalStrengthBalancer>
          <medianOneMerA>1.01745</medianOneMerA>
          <medianOneMerT>0.986296</medianOneMerT>
          <medianOneMerG>0.987408</medianOneMerG>
          <medianOneMerC>0.980024</medianOneMerC>
        </NukeSignalStrengthBalancer>
      </Other>
    </RunMetrics>
  </Metrics>
```

Figure 13– 4: Example metrics.xml stream

13.3.1.6 sequences.xml

A list of sequences referred to by the Sequence ID field in the locations.idx node. This stream can also be used to identify which sequences denote Control DNA reads and which are library fragments. By convention, library keys are named "ATCG-library" where "ATGC" is the four letter library key. An example sequences.xml file is shown in Figure 13–5. (Note that the sequences were truncated (ellipses) on the Figure, for brevity; the full sequence of each Control DNA is included in the actual sequences.xml file).

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Sequences>
  <Sequence Type="None">
    <ID>0</ID>
    <Name>unknown</Name>
    <Key></Key>
    <Seq></Seq>
  </Sequence>
  <Sequence Type="Control">
    <ID>1</ID>
    <Name>ATGC-control</Name>
    <Key>ATGC</Key>
    <Seq>ATGC</Seq>
  </Sequence>
  <Sequence Type="Library">
    <ID>2</ID>
    <Name>TCAG-key</Name>
    <Key>TCAG</Key>
    <Seq>TCAG</Seq>
  </Sequence>
  <Sequence Type="Control">
    <ID>3</ID>
    <Name>TF2LonG</Name>
    <Key>ATGC</Key>
    <Seq>ATGCCA...TGTGTG</Seq>
  </Sequence>
  <Sequence Type="Control">
    <ID>4</ID>
    <Name>TF7LonG</Name>
    <Key>ATGC</Key>
    <Seq>ATGC...TTCCTGTGTG</Seq>
  </Sequence>
  <Sequence Type="Control">
    <ID>5</ID>
    <Name>TF90LonG</Name>
    <Key>ATGC</Key>
    <Seq>ATGCCGCA...GTGTG</Seq>
  </Sequence>
  <Sequence Type="Control">
    <ID>6</ID>
    <Name>TF100LonG</Name>
    <Key>ATGC</Key>
    <Seq>ATGCAT...GTGTG</Seq>
  </Sequence>
  <Sequence Type="Control">
    <ID>7</ID>
    <Name>TF120LonG</Name>
    <Key>ATGC</Key>
    <Seq>ATGCA...CCTGTGTG</Seq>
  </Sequence>
  <Sequence Type="Control">
    <ID>8</ID>
    <Name>TF150MMP7A</Name>
    <Key>ATGC</Key>
    <Seq>ATGCGC...ATGG</Seq>
  </Sequence>
</Sequences>
```

Figure 13– 5: Example sequences.xml stream

13.3.1.7 filters.xml

A list of filters referred to by the values in the "filterResults.uint8.dat" stream (see section 13.3.1.8, below). Note that the order of filters in this file is not guaranteed. It is also likely that the filters will be reorganized in a future release of the software to provide more detail. An example filters.xml file is shown in Figure 13–6.

```xml
<?xml version="1.0" encoding="utf-8"?>
<Filters xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="GSDataProcessing-1.0.xsd">
  <Filter basic="true">
    <ID>0</ID>
    <Name>Pass</Name>
  </Filter>
  <Filter basic="true">
    <ID>1</ID>
    <Name>No Key</Name>
  </Filter>
  <Filter basic="true">
    <ID>2</ID>
    <Name>Bad Band</Name>
  </Filter>
  <Filter basic="true">
    <ID>3</ID>
    <Name>Trimmed Too Short Quality</Name>
  </Filter>
  <Filter basic="true">
    <ID>4</ID>
    <Name>Low Pass Filter</Name>
  </Filter>
  <Filter basic="true">
    <ID>5</ID>
    <Name>Classifier Filter</Name>
  </Filter>
  <Filter basic="true">
    <ID>6</ID>
    <Name>Dot Filter</Name>
  </Filter>
  <Filter basic="true">
    <ID>7</ID>
    <Name>Mixed Filter</Name>
  </Filter>
  <Filter basic="true">
    <ID>8</ID>
    <Name>Trimmed Too Short Primer</Name>
  </Filter>
  <Filter basic="true">
    <ID>9</ID>
    <Name>Low Quality</Name>
  </Filter>
</Filters>
```

Figure 13–6: Example filters.xml stream

13.3.1.8 filterResults.uint8.dat

This file contains a binary list of wells that failed filtering and the specific filter that the well did not pass. The filters are defined in file filters.xml. This stream consists of one byte per well, sorted by rank. See the discussion of “Other Streams” (sections 13.3.1.10 and 13.3.1.11) for a description on the layout of this stream.

The majority of the CWF payload consists of the well flow values themselves. In order to support rapid, random flow extraction from a CWF file, wells are stored in “blocks”. The data format for the blocks can vary, depending on the application. The size of each block is equal to the largest multiple of the data size times the number of flows, that is smaller than 32 Megabytes. In other words, no well’s flow values will be broken between blocks and no block will be larger than 32 MB.

The naming convention for each block of flowgrams is:
TY-Z.wel

… where

- T is a letter indicating the type of the block (see below),
- Y is the starting well index, and
- Z is the final well index.

Y and Z are integer values and should not be padded with zeros. The number of wells in each block is arbitrary and should not be hard coded. There is no assumption that all blocks in the file are stored in the same format nor that all block have the same number of wells. For example, Control DNA reads and failed wells may be archived in a lower fidelity format, while the library fragments are stored as full floating point numbers. However, the format cannot currently store discontinuous ranges of elements in a block, which constitutes a limitation of this feature. Users of the CWF format are encouraged to use libcwf to insulate them from the complexity of extracting with the well information from the CWF file.

The block types are as follows:

- r: This is a raw well block type, identical to the *.wells format of software versions 1.1.03 and earlier. Values are stored “Little Endian” byte order as generated by Intel brand ×86 processors.
- Header:
  - 32 bits: numWells (as unsigned integer)
  - 16 bits: numFlows (as unsigned integer)
  - numFlows bytes: flowLabels (one of “A”,”T”,”G”,”C”,”P”)
- Body:
  - 32 bits: rank (as unsigned integer)
  - 16 bits: xCoord (as unsigned integer)
  - 16 bits: yCoord (as unsigned integer)
  - 32 bits * numFlows: flowValues (as IEEE Single Precision Float)

- h: Half-precision floating point: Each block is made up of four arrays stored back-to-back without padding. The size of the first three arrays is equal to the data type size times the number of wells in the block. The last array consumes the rest of the block, and is equal to 2 bytes times the number of wells times the number of flows. The number of wells is derived from the name of the stream, and the number of flows can be retrieved from the meta.xml stream. Values are stored "Little Endian" byte order as generated by Intel brand ×86 processors.
  - X-coordinates as unsigned 16-bit numbers
  - Y-coordinates as unsigned 16-bit numbers
  - flow ranks as unsigned 32-bit numbers
  - flow values as "half-precision binary floating point numbers"

The half-precision floating point is a relatively new binary floating point format that uses 2 bytes and which is not covered by the IEEE 754 standard for encoding floating point numbers (but is included in the IEEE 754r proposed revision; http://www.validlab.com/754R/). The format uses 1 sign bit, a 5-bit excess-15 exponent, 10 mantissa bits (with an implied 1 bit) and all the standard IEEE rules. The minimum and maximum representable values are 2.98×10^{-8} and 65504, respectively. Libcwf includes a half to full precision floating point conversion routine.

- f: Full-precision floating point: Each block is made up of four arrays stored back-to-back without padding. The size of the first three arrays is equal to the data type size times the number of wells in the block. The last array consumes the rest of the block, and is equal to 4 bytes times the number of wells times the number of flows. The number of wells is derived from the name of the stream, and the number of flows can be retrieved from the meta.xml stream. Values are stored "Little Endian" byte order as generated by Intel brand ×86 processors.
  - X-coordinates as unsigned 16-bit numbers
  - Y-coordinates as unsigned 16-bit numbers
  - flow ranks as unsigned 32-bit numbers
  - flow values as IEE 754 binary full precision floating point number (http://grouper.ieee.org/groups/754/)
- i: Integer: Each block is made up of four arrays stored back-to-back without padding. The size of the first three arrays is equal to the data type size times the number of wells in the block. The last array consumes the rest of the block, and is equal to 2 bytes times the number of wells times the number of flows. The number of wells is derived from the name of the stream, and the number of flows can be retrieved from the meta.xml stream. Values are stored "Little Endian" byte order as generated by Intel brand ×86 processors. (Note that this block is rarely used because it cannot store values less than zero, which can occur during signal processing routines, and lacks precision for values near one.)
  - X-coordinates as unsigned 16-bit numbers
  - Y-coordinates as unsigned 16-bit numbers
  - flow ranks as unsigned 32-bit numbers
  - flow values as IEE 754 binary full-precision floating point number

13.3.1.9 Base Called Data

If a data set has processed through the baseCaller section of the GS Run Processor, the actual bases are written into the CWF file. This allows for the generation of FASTA FNA and QUAL files on demand. Like the flowgrams, the reads are stored in blocks. Unlike the flowgrams, however, each block consists of a variable number of reads. A special stream called “baseCalledSeq.dat” is used to index the base called blocks. This stream contains 6 bytes per well:

- Byte 0/1: Total stored read length
- Byte 2/3: Number of reads to trim from the distal end (3’)
- Byte 4/5: Number of reads to trim from the local end (5’)

The complete data for any read is stored in three separate streams, identified by the “dna”, “qual” and “flow” extensions, containing the reads, the PHRED based quality values and the offset flow indices, respectively. Each base of each well uses one byte in each file.

The total number of bytes consumed by a read is reflected in the first field of the baseCalledSeq.dat. Therefore these two bytes can be used as an index of sorts. For example, the byte offset in the “dna” file for read 100 can be found by summing the first two bytes of the first 99 entries in baseCalledSeq.dat. The 100th entry can then be used to tell how many bytes are available for read 100. It is important to note that since the base calls are stored in blocks, one must first find the appropriate block, then compute the offset from there. Again, users of the CWF format are encouraged to use the libcwf to insulate them from errors in extracting the base information.

13.3.1.10 Other Public Streams

The cwf file can also contain other binary streams. These are usually identified by the .dat suffix (except in the case of the “image” type, see note below). Many are used for storing intermediate results of various processing stages, but there are three notable streams that contain metrics data that may be of interest to end-users.

- The first, “filterResults,” can be used to find which wells have passed which filter.
- The second, “rawWellDensity” is a grayscale image containing the bead loading density plot.
- The last is the “keyPassDensity” stream, a grayscale image showing the keypass density.

With the exception of the “image” format stream, each stream contains one entry per read, sorted in rank order. The location.idx can be used to find which offset corresponds to which read.

The possible data types are listed in Table 13–3: Possible data types, and the current stream types, in Table 13–4: Current stream types.

Name	Size in Bytes/Well	Notes
byte	1	
short	2	Signed short, Intel byte order
unsignedShort	2	Unsigned short, Intel byte order
int	4	Standard integer, intel byte order
unsignedInt	4	Unsigned integer, Intel byte order
float	4	IEEE 754 Single Precision floating point number
double	4	IEEE 754 Double Precision floating point number
image	N/A	PGM format graphics. See note below on resolution / registration.

Table 13–3: Possible data types

Stream Type	Contents
filterResults	The information about which well failed which filter in the qualityFiltering section of code.
trimInfo	The trim points from the end of the reads. This number is in flowgram-space instead of base-space like the information in the baseCallerSeq.dat stream.
cfValues	The carry forward corrections for each well.
ieValues	The incomplete extension correction factors for each well.
signalPerBase	The average value of the keypass flows for this well. Can be used to do a simple base calling. Also can be used to judge the strength of the well.
keyPassDensity	An image referenced to the region showing the density wells that pass key. This is normalized for each PTP pitch, so 0=0% loading and 255=100% loading.
rawWellDensity	An image referenced to the region showing the density of the bead loading. This is normalized for each PTP pitch, so 0=0% loading and 255=100% loading.

Table 13–4: Current stream types

Note on Image Formats: Currently, the only image format stored in CWF is the lossless Portable Anymap format, specifically the "P5" PGM (Portable Graymap) variant (http://netpbm.sourceforge.net/doc/pgm.html). To save space, only the area encompassed by the region is included in the image. To properly register the image against the original PTP device, you must offset the image slice by the region boundary. This region boundary can be read in the "RevisedBounds" element of the "Region" block in meta.xml stream. 454 Life Sciences Corporation may introduce other image formats in future variants of the CWF file, so it is important to read the "magic number" and/or file extension of the image to determine the correct image decoder to use.

13.3.1.11 Other Private Streams

There are other streams of data that can be stored in the CWF file, that contain intermediate results from various pipeline and post processing functions. These are left unspecified to avoid restricting the development of new algorithms in the data processing applications. CWF file users should neither depend on their existence or attempt to parse them as their contents may change between revisions of the software or invocations of different processing streams.

13.3.1.12 Other Files

There may be other files in the CWF container, the contents of which may include log files and other binary data. A proper CWF writer (like the GS Run Processor) will copy any other unknown stream verbatim to the destination. CWF file readers should not balk at these extra streams, but should not depend on their existence either. This allows advanced users to add additional payloads to the CWF container before moving them to the data analysis system.

13.3.2 Parameters and Viewable Metrics Files: 454 Parser Format (.parse, .txt)

The "parser" format, developed at 454 Life Sciences Corporation, is a standard format used for all the software parameter files and for most of the metrics files. This is a text-based format that organizes the text in titled "groups" that contain either sub-groups or name/value pairs of strings. A parser file consists of one or more of the following:

- A C-style comment, using /* and */ to delineate the comment text
- A group, whose syntax is '`groupname { … }`' and where one or more comments, sub-groups or name/value pairs can occur between the braces
- A name/value pair, whose syntax is '`name = "value";`' and where the quotes around the value are optional, but the equals sign and semi-colon are required.

The parser file format is free-form, *i.e.* the syntactic elements can appear with any style of white space or line division. However, the output files generated by the Genome Sequencer FLX System software use a standard indentation convention where the group names appear by themselves on a line with the braces below it; all the text between the braces is indented; and each name/value pair appears on a single line.

Several examples of parser files are shown in section 13.4.

13.3.3 Image Files (.pif)

The ".pif" file format was developed at 454 Life Sciences Corporation for storing image data from the Genome Sequencer FLX Instrument. The file consists of a header, followed by data. The byte order is little endian.

The header is 12 bytes long, comprised of three 4-byte integers: the first integer value is the number of bits per pixel of data; the second integer is the width of the image in pixels; and the third integer is the height of the image in pixels. The following data in the file are the pixel intensity values, presented in row major order starting in the upper left corner. Currently, all image data is stored in 16 bit unsigned integers, or 2 bytes per pixel. Valid image data is limited to the first 14 bits.

As these are binary-format files, an example cannot be provided in this text-based document.

13.3.4 Well-Level Signal Data Files (.wells)

The wells data file is a legacy binary file containing counts values for the light collected on the nucleotide and PPi images, at each "active" well location. The file consists of a header and a body, where the header contains the following fields:

```
unsigned int numWells;
unsigned short numFlows;
char flowOrder[numFlows];
```

…where numWells is the number of wells (or reads) in the file, numFlows is the number of flows in the sequencing Run script, and flowOrder are characters indicating the reagent for each flow: 'A', 'C', 'G' and 'T' specify each nucleotide flow, and 'P' signifies a PPi flow.

The body of the wells data file contains numWells records of the following fields:

```
unsigned int rank;
unsigned short x;
unsigned short y;
float flowValues[numFlows];
```

…where rank is the general ranking of the well (by signal intensity); x and y are the coordinates of the well center pixel; and flowValues are the signal values for all the flows in this well.

All the multi-byte values in the header and body are written using little endian byte ordering (consistent with the Linux operating system). As these are binary-format files, an example cannot be provided in this text-based document.

13.3.5 Exportable Metrics Files (.csv)

The comma-separated values text file (.csv) is an alternate format in which a number of the metrics files are output. This format is suitable for automated parsing by programs or for loading into a spreadsheet program like Microsoft Excel. The contents of these files, where generated, is identical to the corresponding file formatted in the 454 parser format described above (.txt).

13.3.6 DNA Sequence (FASTA; .fna) and Base Quality Score (.qual) Files

Three of the Genome Sequencer FLX System data processing applications output DNA sequences: Signal Processing, for the basecalls of individual reads; GS *De Novo* Assembler, for the *de novo*-assembled consensus sequence of the sample DNA library; and GS Reference Mapper, for the sample's consensus sequence mapped to a reference sequence. These use the FASTA standard file format (.fna), and are always accompanied by a corresponding base quality scores file, in the .qual format. Examples are shown in section 13.4.4 (*region.key*.454Reads) and 13.4.5 (454AllContigs).

Note that the description lines are slightly different depending on whether the FASTA file outputs contain reads or contigs, and for contigs, whether they were generated by the GS *De Novo* Assembler or by the GS Reference Mapper application.

1. For individual reads, the description lines are formatted as:

   >*rank_x_y* length=*XX*bp uaccno=*accession*

   …where "*rank_x_y*" is the identifier or accession number of the read (the rank, x and y values are as described in section 13.3.4), *XX*bp is the length in bases of the read, and *accession* is the full universal accession number for the read.

2. For contigs generated by the GS *De Novo* Assembler application, the description lines are formatted as follows:

   \>contig*XXXXX* length=*abc* numReads=*xyz*

   …where "contig*XXXXX*" is the identifier of the contig and "*XXXXX*" is a sequential numbering of the contigs in the assembly; and where the length and numReads values are the length in bases of the contig and the number of reads that were used in that contig's multiple alignment.

3. For contigs generated by the GS Reference Mapper application, the description lines are formatted as follows:

   \>contig*XXXXX* *refaccno*, *YYY..ZZZ* length=*abc* numReads=*xyz*

   …where "contig*XXXXX*" is the identifier of the contig and "*XXXXX*" is a sequential numbering of the contigs along the reference; "*refaccno*" is the accession of the reference sequence where this contig aligns; "*YYY..ZZZ*" is the start and end position of the contig on that reference sequence; and the length and numReads values are the length in bases of the contig and the number of reads that were used in that contig's multiple alignment.

13.3.7 454 "Universal" Accession Numbers

The standard 454 read identifiers, used in Genome Sequencer System data analysis software versions prior to 1.0.52 (early GS 20), have the format "rank_x_y" (as in 003048_1034_0651), where "rank" is a ranking of the well in a region by signal intensity, and "x" and "y" are the pixel location of the well's center on the sequencing Run images. This identifier is guaranteed to be unique only within the context of a single sequencing Run, and may or may not be unique across specific sets of Runs.

To allow for the combination of reads across larger data sets, a more unique accession number format has been developed. An accession in this format is a 14 character string, as in C3U5GWL01CBXT2, and consist of 4 components:

C3U5GW	- a six character encoding of the timestamp of the Run
L	- a randomizing "hash" character to enhance uniqueness
01	- the region the read came from, as a two-digit number
CBXT2	- a five character encoding of the X,Y location of the well

The timestamp, hash character and X,Y location use a base-36 encoding (where values 0-25 are the letters 'A'-'Z' and the values 26-35 are the digits '0'-'9'). An accession thus consists only of letters and digits, and is case-insensitive.

- The timestamp is encoded by computing a "total" value as shown below, then converting it into a base-36 string:

  ```
  total =
          (year - 2000) * 13 * 32 * 24 * 60 * 60 +
          month * 32 * 24 * 60 * 60 +
          day * 24 * 60 * 60 +
          hour * 60 * 60 +
          minute * 60 +
          second;
  ```

  As a result of this calculation, the first character of read accessions will always be a letter for Runs performed from now until 2038. The timestamp values are taken from the rigRunName found in the analysisParms.parse file in the specified analysis directory. This rigRunName is the R_... name that is generated by the instrument software, and is also used as the standard directory name for the Run. Thus, a Run whose name begins with R_2004_09_22_16_59_10_... generates C3U5GW as its encoded timestamp value.

- Since two Runs may be started at the same second, an additional base-36 character is generated by hashing the full rigRunName to a base-31 number (the highest prime below 36), as in:

  ```
  chval = 0;
  for (s=rigRunName; *s; s++) {
              chval += (int) *s;
              chval %= 31;
  }
  ch = (chval < 26 ? 'A' + chval : '0' + chval - 26);
  ```

- The X,Y location is encoded by computing a total value of "X * 4096 + Y" and encoding that as a five character, base-36 string.

13.3.8 Standard Flowgram Files (.sff)

The Standard Flowgram File is used to store the information on one or many GS FLX reads and their trace data. Sequencing reads obtained using the Genome Sequencer FLX System differ from reads obtained using more traditional methods ("Sanger sequencing") in that the GS FLX data does not provide individual base measurements from which basecalls can be derived. Instead, it provides measurements that estimate the length of each homopolymer stretch in the sequence (*e.g.* in "AAATGG", "AAA" is a 3-mer stretch of A's, "T" is a 1-mer stretch of T's and "GG" is a 2-mer stretch of G's). A basecalled sequence is then derived by converting each estimate into a homopolymer stretch of that length and concatenating the homopolymers.

The .sff file format consists of three sections: a common header section occurring once in the file; then for each read stored in the file, a read header section and a read data section. The data in each section consists of a combination of numeric and character data; the specific fields for each section are defined below. The sections adhere to the following rules:

- The standard Unix types uint8_t, uint16_t, uint32_t and uint64_t are used to define 1, 2, 4 and 8 byte numeric values.
- All multi-byte numeric values are stored using big endian byteorder (same as the SCF file format).
- All character fields use single-byte ASCII characters.
- Each section definition ends with an "eight_byte_padding" field, which consists of 0 to 7 bytes of padding, so that the byte length of each section is divisible by 8 (and hence the next section is aligned on an 8-byte boundary).

13.3.8.1 Common Header Section

The common header section consists of the following fields:

Field name	Format	Properties
magic_number	uint32_t	The magic_number field value is 0x2E736666, the uint32_t encoding of the string „.sff“.
version	char[4]	The version number is 0001, or the byte array „\0\0\0\1“.
index_offset	uint64_t	The index_offset and index_length fields are the offset and length of an optional index of the reads in the SFF file. If no index is included in the file, both fields must be 0.
index_length	uint32_t	
number_of_reads	uint32_t	The number_of_reads field should be set to the number of reads stored in the file.
header_length	uint16_t	The header_length field should be the total number of bytes required by this set of header fields, and should be equal to „31 + number_of_flows_per_read + key_length“, rounded up to the next value divisible by 8.
key_length	uint16_t	The key_length field should be set to the length of the key sequence used for these reads.
number_of_flows _per_read	uint16_t	The number_of_flows_per_read should be set to the number of flows for each of the reads in the file.
flowgram_format_ code	uint8_t	The flowgram_format_code should be set to the format used to encode each of the flowgram values for each read. Currently, only one flowgram format has been adopted, so this value should be set to 1. The flowgram format code 1 stores each value as a uint16_t, where the floating point flowgram value is encoded as „(int) round(value * 100.0)“, and decoded as „(storedvalue / 100.0)“. In other words, the values are stored as an integer encoding of a limited precision floating point value, keeping 2 places to the right of the decimal point, and capping the values at 655.35.
flow_chars	char[number_of _flows_per_read]	The flow_chars should be set to the array of nucleotide bases (‚A‘, ‚C‘, ‚G‘ or ‚T‘) that correspond to the nucleotides used for each flow of each read. The length of the array should equal number_of_flows_per_read. Note that the flow_chars field is not null-terminated.
key_sequence	char[key_length]	The key_sequence field should be set to the nucleotide bases of the key sequence used for these reads. Note that the key_sequence field is not null-terminated.
eight_byte_ padding	uint8_t[*]	If any eight_byte_padding bytes exist in the section, they should have a byte value of 0.

If an index is included in the file, the index_offset and index_length values in the common header should point to the section of the file containing the index. To support different indexing methods, the index section should begin with the following two fields:

▶ index_magic_number	uint32_t
▶ index_version	char[4]

… and should end with an eight_byte_padding field, so that the length of the index section is divisible by 8. The format of the rest of the index section is specific to the indexing method used. The index_length given in the common header should include the bytes of these fields and the padding.

13.3.8.2 Read Header Section

The rest of the file contains the information about the reads, namely number_of_reads entries consisting of read header and read data sections. Each read header section consists of the following fields:

Field name	Format	Properties
read_header_ length	uint16_t	The read_header_length should be set to the length of the read header for this read, and should be equal to „16 + name_length“ rounded up to the next value divisible by 8.
name_length	uint16_t	The name_length field should be set to the length of the read‘s accession or name.
number_of_bases	uint32_t	The number_of_bases should be set to the number of bases called for this read.
clip_qual_left	uint16_t	▶ The clip_qual_left and clip_adapter_left fields should be set to the position of the first base after the clipping point, for quality and/or an adapter sequence, at the beginning of the read. If only a combined clipping position is computed, it should be stored in clip_qual_left. ▶ The clip_qual_right and clip_adapter_right fields should be set to the position of the last base before the clipping point, for quality and/or an adapter sequence, at the end of the read. If only a combined clipping position is computed, it should be stored in clip_qual_right. Note that the position values use 1-based indexing, so the first base is at position 1. If a clipping value is not computed, the field should be set to 0. Thus, the first base of the insert is: max(1, max(clip_qual_left, clip_adapter_left)) …and the last base of the insert is: min((clip_qual_right == 0 ? number_of_bases : clip_qual_right), (clip_adapter_right == 0 ? number_of_bases : clip_adapter_right))
clip_qual_right	uint16_t	
clip_adapter_left	uint16_t	
clip_adapter_right	uint16_t	
Name	char[name_ length]	The name field should be set to the string of the read‘s accession or name. Note that the name field is not null-terminated.
eight_byte_padding	uint8_t[*]	If any eight_byte_padding bytes exist in the section, they should have a byte value of 0.

13.3.8.3 Read Data Section

The read data section consists of the following fields:

Field name	Format	Properties
flowgram_values	uint*_t[number _of_flows]	The flowgram_values field contains the homopolymer stretch estimates for each flow of the read. The number of bytes used for each value depends on the common header flowgram_format_code value (where the current value uses a uint16_t for each value).
flow_index_per_base	uint8_t[number _of_bases]	The flow_index_per_base field contains the flow positions for each base in the called sequence (*i.e.*, for each base, the position in the flowgram whose estimate resulted in that base being called). Note that these values are "incremental" values, *i.e.* the stored position is the offset from the previous flow index in the field. All position values (prior to their incremental encoding) use 1-based indexing, so the first flow is flow 1.
Bases	char[number _of_ bases]	The bases field contains the basecalled nucleotide sequence.
quality_scores	uint8_t[number _of_bases]	The quality_scores field contains the quality scores for each of the bases in the sequence, where the values use the standard -log10 probability scale.
eight_byte_padding	uint8_t[*]	If any eight_byte_padding bytes exist in the section, they should have a byte value of 0.

13.3.8.4 Computing Lengths and Scanning the File

The length of the various read's section will vary because of different length accession numbers and different length nucleotide sequences. However, the various flow, name and bases lengths given in the common and read headers can be used to scan the file, accessing each read's information or skipping read sections in the file. The following pseudocode gives an example method to scan the file and access each read's data:

```
1  Open the file and/or reset the file pointer position to the first byte of the file.
2  Read the first 31 bytes of the file, confirm the magic_number value and version, then
   extract the number_of_reads, number_of_flows_per_read, flowgram_format_code,
   header_length, key_length, index_offset and index_length values.
   a. Convert the flowgram_format_code into a flowgram_bytes_per_flow value
      (currently with format_code 1, this value is 2 bytes).
3  If the flow_chars and key_sequence information is required, read the next
   "header_length - 31" bytes, then extract that information. Otherwise, set the file
   pointer position to byte header_length.
4  While the file contains more bytes, do the following:
   a. If the file pointer position equals index_offset, either read or skip index_length
      bytes in the file, processing the index if read.
   b. Otherwise,
      I.   Read 16 bytes and extract the read_header_length, name_length and number_
           of_bases values.
      II.  Read the next "read_header_length - 16" bytes to read the name.
      III  At this point, a test of the name field can be performed, to determine whether to
           read or skip this entry.
           01. Compute the read_data_length as
           "number_of_flows * flowgram_bytes_per_flow + 3 * number_of_bases"
           ...rounded up to the next value divisible by 8.
           02. Either read or skip read_data_length bytes in the file, processing the read
               data if the section is read.
```

13.4 Example Files

13.4.1 454QualityFilterMetrics.txt

Example file listing for 454QualityFilterMetrics.txt

```
/****************************************************************************
**
**      454 Life Sciences Corporation
**
**      Software Release: 2.0.00.17
**
**      Quality Filter Metrics Results
**
**      Run Name:      R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli
**      Analysis Name: R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcol
i/D_2008_08_19_18_48_14_dragonforce_fullProcessing
**
**      File Created:  2008/05/29 07:49:18
**
***************************************************************************/

/*
**      Run conditions
*/
runConditions
{
        dateOfFile          = "2008/05/29 07:49:18";

        rigRunName          = "R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70
xVFRxEcoli";
        analysisName  = "R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEc
oli/D_2008_08_19_18_48_14_dragonforce_fullProcessing";
        PTPBarCode          = "Unknown";
}

region
{
        name            = 1;
        totalRawWells   = 1021913;
        totalKeyPass    = 1014938;

        key
        {
                keySequence                 = CATG;
                numKeyPass                  = 8205;
                numDotFailed                = 230;
                numMixedFailed              = 107;
                numTrimmedTooShortQuality = 1848;
                numTrimmedTooShortPrimer  = 0;
                totalPassedFiltering      = 6020;
        }
        key
        {
                keySequence                 = TCAG;
                numKeyPass                  = 1006733;
                numDotFailed                = 25656;
                numMixedFailed              = 49265;
                numTrimmedTooShortQuality = 243222;
                numTrimmedTooShortPrimer  = 316;
                totalPassedFiltering      = 688274;
        }
}
```

```
region
{
       name              = 2;
       totalRawWells     = 1102293;
       totalKeyPass      = 1096067;

       key
       {
              keySequence                = CATG;
              numKeyPass                 = 8070;
              numDotFailed               = 245;
              numMixedFailed             = 165;
              numTrimmedTooShortQuality = 2187;
              numTrimmedTooShortPrimer  = 0;
              totalPassedFiltering       = 5473;
       }
       key
       {
              keySequence                = TCAG;
              numKeyPass                 = 1087997;
              numDotFailed               = 24931;
              numMixedFailed             = 57354;
              numTrimmedTooShortQuality = 296228;
              numTrimmedTooShortPrimer  = 330;
              totalPassedFiltering       = 709154;
       }
}
```

13.4.2 454BaseCallerMetrics.txt

Example file listing for 454BaseCallerMetrics.txt (truncated, as shown by bracketed ellipsis).

```
/***************************************************************************
**
**      454 Life Sciences Corporation
**
**      Software Release: 2.0.00.17
**
**      Base Call Metrics Results
**
**      File Created:  2008/05/29 07:49:18
**
***************************************************************************/

/*
**      Run Conditions
*/
runParameters
{
       rigResultDirectory = "R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xV
FRxEcoli/D_2008_08_19_18_48_14_dragonforce_fullProcessing";
       PTPBarCode         = "701659";
}
/*
**      Basecall Results
*/
basecallResults
{
       numReads  = 1408917;
       aveLength = 411.680;
       stdDev    = 125.150;
}
/*
**      Region-Key Results
*/
regionKey
{
       region = 1;
       key    = CAT;

       numberReads    = 6021;
       totalBases     = 2076626;
       averageLength  = 344.897,109.984;
       averageQuality = 24.8004,7.19542;

       lengthHistogram
       {
              lengthCount = 49,2;
              lengthCount = 50,4;
              lengthCount = 51,3;
              lengthCount = 52,1;
              lengthCount = 53,5;
              lengthCount = 54,3;
              lengthCount = 55,3;
              lengthCount = 56,1;
              lengthCount = 57,7;
              lengthCount = 58,2;
              lengthCount = 59,4;
              lengthCount = 60,7;
              lengthCount = 61,10;
              lengthCount = 62,8;
              lengthCount = 63,10;
              lengthCount = 64,4;
              lengthCount = 65,2;
              lengthCount = 66,3;
              lengthCount = 67,6;
              lengthCount = 68,8;
              lengthCount = 69,1;
              lengthCount = 70,4;
              lengthCount = 71,10;
              lengthCount = 72,12;
              lengthCount = 73,7;
              lengthCount = 74,6;
              lengthCount = 75,2;
              lengthCount = 76,6;
              lengthCount = 77,5;
              lengthCount = 78,8;
              lengthCount = 79,1;
              lengthCount = 80,3;
              lengthCount = 81,6;
              lengthCount = 82,3;
              lengthCount = 83,6;
              lengthCount = 84,8;
              lengthCount = 85,6;
              lengthCount = 86,1;
              lengthCount = 87,8;
              lengthCount = 88,7;
              lengthCount = 89,4;
              lengthCount = 90,2;
              lengthCount = 91,3;
              lengthCount = 92,2;
              lengthCount = 93,2;
              lengthCount = 94,10;
              lengthCount = 95,1;
              lengthCount = 96,2;
              lengthCount = 97,2;
              lengthCount = 98,7;
              lengthCount = 99,10;
              lengthCount = 100,8;
              lengthCount = 101,4;
              lengthCount = 102,5;
              lengthCount = 103,6;
              lengthCount = 104,6;
              lengthCount = 105,5;
              lengthCount = 106,4;
              lengthCount = 107,4;
              lengthCount = 108,2;
              lengthCount = 109,4;
              lengthCount = 110,2;
              lengthCount = 111,5;
              lengthCount = 112,5;
              lengthCount = 113,3;
              lengthCount = 114,2;
              lengthCount = 115,2;
              lengthCount = 116,3;
              lengthCount = 117,5;
              lengthCount = 118,4;
              lengthCount = 119,4;
              lengthCount = 120,4;
              lengthCount = 121,2;
              lengthCount = 122,4;
              lengthCount = 123,1;
              lengthCount = 124,3;
              lengthCount = 125,2;
              lengthCount = 126,3;
              lengthCount = 127,4;
              lengthCount = 128,1;
              lengthCount = 129,5;
```

```
                lengthCount = 130,1;
                lengthCount = 132,3;
                lengthCount = 134,6;
                lengthCount = 135,1;
                lengthCount = 136,3;
                lengthCount = 138,7;
                lengthCount = 139,2;
                lengthCount = 140,1;
                lengthCount = 141,2;
                lengthCount = 142,4;
                lengthCount = 143,7;
                lengthCount = 144,6;
                lengthCount = 145,6;
                lengthCount = 146,1;
                lengthCount = 147,1;
                lengthCount = 148,5;
                lengthCount = 149,3;
                lengthCount = 150,1;
                lengthCount = 151,4;
                lengthCount = 152,3;
                lengthCount = 153,5;
                lengthCount = 154,6;
                lengthCount = 155,7;
                lengthCount = 156,10;
                lengthCount = 157,2;
                lengthCount = 158,1;
                lengthCount = 159,3;
                lengthCount = 160,6;
                lengthCount = 162,5;
                lengthCount = 163,2;
                lengthCount = 164,5;
                lengthCount = 165,3;
                lengthCount = 166,3;
                lengthCount = 167,2;
                lengthCount = 168,1;
                lengthCount = 169,6;
                lengthCount = 170,1;
                lengthCount = 171,4;
                lengthCount = 172,1;
                lengthCount = 173,3;
                lengthCount = 175,5;
                lengthCount = 176,8;
                lengthCount = 177,3;
                lengthCount = 178,1;
                lengthCount = 179,2;
                lengthCount = 180,1;
                lengthCount = 181,2;
                lengthCount = 182,1;
                lengthCount = 183,4;
                lengthCount = 184,3;
                lengthCount = 185,2;
                lengthCount = 186,2;
                lengthCount = 187,6;
                lengthCount = 188,5;
                lengthCount = 189,1;
                lengthCount = 190,6;
                lengthCount = 191,3;
                lengthCount = 192,7;
                lengthCount = 193,6;
                lengthCount = 194,9;
                lengthCount = 195,8;
                lengthCount = 196,4;
                lengthCount = 197,4;
                lengthCount = 198,1;
                lengthCount = 199,3;
```

```
lengthCount = 200,4;
lengthCount = 201,8;
lengthCount = 202,4;
lengthCount = 203,6;
lengthCount = 204,5;
lengthCount = 205,2;
lengthCount = 206,11;
lengthCount = 207,4;
lengthCount = 208,4;
lengthCount = 209,4;
lengthCount = 210,9;
lengthCount = 211,6;
lengthCount = 212,6;
lengthCount = 213,7;
lengthCount = 214,9;
lengthCount = 215,3;
lengthCount = 216,11;
lengthCount = 217,9;
lengthCount = 218,6;
lengthCount = 219,7;
lengthCount = 220,9;
lengthCount = 221,16;
lengthCount = 222,8;
lengthCount = 223,7;
lengthCount = 224,7;
lengthCount = 225,8;
lengthCount = 226,6;
lengthCount = 227,9;
lengthCount = 228,4;
lengthCount = 229,10;
lengthCount = 230,6;
lengthCount = 231,7;
lengthCount = 232,7;
lengthCount = 233,7;
lengthCount = 234,5;
lengthCount = 235,4;
lengthCount = 236,5;
lengthCount = 237,7;
lengthCount = 238,9;
lengthCount = 239,8;
lengthCount = 240,4;
lengthCount = 241,8;
lengthCount = 242,7;
lengthCount = 243,11;
lengthCount = 244,8;
lengthCount = 245,11;
lengthCount = 246,7;
lengthCount = 247,9;
lengthCount = 248,7;
lengthCount = 249,15;
lengthCount = 250,13;
lengthCount = 251,12;
lengthCount = 252,17;
lengthCount = 253,5;
lengthCount = 254,13;
lengthCount = 255,8;
lengthCount = 256,4;
lengthCount = 257,4;
lengthCount = 258,4;
lengthCount = 259,6;
lengthCount = 260,3;
lengthCount = 261,7;
lengthCount = 262,3;
lengthCount = 263,6;
lengthCount = 264,4;
```

```
                lengthCount = 265,10;
                lengthCount = 266,5;
                lengthCount = 267,15;
                lengthCount = 268,9;
                lengthCount = 269,11;
                lengthCount = 270,6;
                lengthCount = 271,10;
                lengthCount = 272,10;
                lengthCount = 273,4;
                lengthCount = 274,15;
                lengthCount = 275,18;
                lengthCount = 276,18;
                lengthCount = 277,15;
                lengthCount = 278,24;
                lengthCount = 279,14;
                lengthCount = 280,21;
                lengthCount = 281,19;
                lengthCount = 282,20;
                lengthCount = 283,16;
                lengthCount = 284,22;
                lengthCount = 285,24;
                lengthCount = 286,20;
                lengthCount = 287,18;
                lengthCount = 288,18;
                lengthCount = 289,14;
                lengthCount = 290,18;
                lengthCount = 291,6;
                lengthCount = 292,18;
                lengthCount = 293,15;
                lengthCount = 294,15;
                lengthCount = 295,20;
                lengthCount = 296,41;
                lengthCount = 297,50;
                lengthCount = 298,35;
                lengthCount = 299,43;
                lengthCount = 300,40;
                lengthCount = 301,40;
                lengthCount = 302,58;
                lengthCount = 303,47;
                lengthCount = 304,61;
                lengthCount = 305,74;
                lengthCount = 306,75;
                lengthCount = 307,84;
                lengthCount = 308,78;
                lengthCount = 309,79;
                lengthCount = 310,60;
                lengthCount = 311,53;
                lengthCount = 312,35;
                lengthCount = 313,34;
                lengthCount = 314,22;
                lengthCount = 315,32;
                lengthCount = 316,25;
                lengthCount = 317,17;
                lengthCount = 318,11;
                lengthCount = 319,32;
                lengthCount = 320,25;
                lengthCount = 321,15;
                lengthCount = 322,14;
                lengthCount = 323,10;
                lengthCount = 324,11;
                lengthCount = 325,10;
                lengthCount = 326,11;
                lengthCount = 327,17;
                lengthCount = 328,10;
                lengthCount = 329,3;
```

```
lengthCount = 330,18;
lengthCount = 331,15;
lengthCount = 332,13;
lengthCount = 333,8;
lengthCount = 334,12;
lengthCount = 335,7;
lengthCount = 336,13;
lengthCount = 337,14;
lengthCount = 338,14;
lengthCount = 339,15;
lengthCount = 340,9;
lengthCount = 341,26;
lengthCount = 342,25;
lengthCount = 343,17;
lengthCount = 344,24;
lengthCount = 345,21;
lengthCount = 346,20;
lengthCount = 347,24;
lengthCount = 348,29;
lengthCount = 349,27;
lengthCount = 350,17;
lengthCount = 351,15;
lengthCount = 352,5;
lengthCount = 353,13;
lengthCount = 354,17;
lengthCount = 355,26;
lengthCount = 356,15;
lengthCount = 357,16;
lengthCount = 358,14;
lengthCount = 359,11;
lengthCount = 360,23;
lengthCount = 361,21;
lengthCount = 362,13;
lengthCount = 363,10;
lengthCount = 364,11;
lengthCount = 365,17;
lengthCount = 366,21;
lengthCount = 367,15;
lengthCount = 368,15;
lengthCount = 369,17;
lengthCount = 370,27;
lengthCount = 371,20;
lengthCount = 372,16;
lengthCount = 373,15;
lengthCount = 374,13;
lengthCount = 375,13;
lengthCount = 376,12;
lengthCount = 377,6;
lengthCount = 378,18;
lengthCount = 379,10;
lengthCount = 380,10;
lengthCount = 381,13;
lengthCount = 382,16;
lengthCount = 383,18;
lengthCount = 384,14;
lengthCount = 385,14;
lengthCount = 386,15;
lengthCount = 387,8;
lengthCount = 388,11;
lengthCount = 389,9;
lengthCount = 390,15;
lengthCount = 391,11;
lengthCount = 392,12;
lengthCount = 393,19;
lengthCount = 394,17;
lengthCount = 395,13;
lengthCount = 396,14;
lengthCount = 397,16;
lengthCount = 398,14;
lengthCount = 399,9;
lengthCount = 400,12;
lengthCount = 401,19;
lengthCount = 402,22;
lengthCount = 403,15;
lengthCount = 404,16;
lengthCount = 405,18;
lengthCount = 406,18;
lengthCount = 407,24;
lengthCount = 408,20;
lengthCount = 409,16;
lengthCount = 410,20;
lengthCount = 411,17;
lengthCount = 412,23;
lengthCount = 413,31;
lengthCount = 414,30;
lengthCount = 415,22;
lengthCount = 416,23;
lengthCount = 417,18;
lengthCount = 418,19;
lengthCount = 419,15;
lengthCount = 420,18;
lengthCount = 421,14;
lengthCount = 422,16;
lengthCount = 423,20;
lengthCount = 424,17;
lengthCount = 425,18;
lengthCount = 426,12;
lengthCount = 427,12;
lengthCount = 428,19;
lengthCount = 429,15;
lengthCount = 430,29;
lengthCount = 431,28;
lengthCount = 432,27;
lengthCount = 433,17;
lengthCount = 434,20;
lengthCount = 435,20;
lengthCount = 436,15;
lengthCount = 437,21;
lengthCount = 438,18;
lengthCount = 439,18;
lengthCount = 440,18;
lengthCount = 441,16;
lengthCount = 442,5;
lengthCount = 443,11;
lengthCount = 444,12;
lengthCount = 445,15;
lengthCount = 446,13;
lengthCount = 447,9;
lengthCount = 448,15;
lengthCount = 449,20;
lengthCount = 450,17;
lengthCount = 451,13;
lengthCount = 452,13;
lengthCount = 453,5;
lengthCount = 454,14;
lengthCount = 455,25;
lengthCount = 456,12;
lengthCount = 457,7;
lengthCount = 458,24;
lengthCount = 459,13;
lengthCount = 460,21;
lengthCount = 461,34;
lengthCount = 462,25;
lengthCount = 463,26;
lengthCount = 464,16;
lengthCount = 465,30;
lengthCount = 466,22;
lengthCount = 467,18;
lengthCount = 468,19;
lengthCount = 469,14;
lengthCount = 470,19;
lengthCount = 471,14;
lengthCount = 472,11;
lengthCount = 473,19;
lengthCount = 474,15;
lengthCount = 475,16;
lengthCount = 476,19;
lengthCount = 477,20;
lengthCount = 478,10;
lengthCount = 479,13;
lengthCount = 480,14;
lengthCount = 481,15;
lengthCount = 482,15;
lengthCount = 483,10;
lengthCount = 484,10;
lengthCount = 485,6;
lengthCount = 486,13;
lengthCount = 487,11;
lengthCount = 488,3;
lengthCount = 489,11;
lengthCount = 490,12;
lengthCount = 491,17;
lengthCount = 492,14;
lengthCount = 493,21;
lengthCount = 494,26;
lengthCount = 495,51;
lengthCount = 496,59;
lengthCount = 497,53;
lengthCount = 498,57;
lengthCount = 499,55;
lengthCount = 500,52;
lengthCount = 501,37;
lengthCount = 502,27;
lengthCount = 503,11;
lengthCount = 504,11;
lengthCount = 505,3;
lengthCount = 506,3;
lengthCount = 507,3;
lengthCount = 508,4;
lengthCount = 509,1;
lengthCount = 510,1;
lengthCount = 511,4;
lengthCount = 512,6;
lengthCount = 513,2;
lengthCount = 514,6;
lengthCount = 515,6;
lengthCount = 516,3;
lengthCount = 517,1;
lengthCount = 518,6;
lengthCount = 519,2;
lengthCount = 520,5;
lengthCount = 521,2;
lengthCount = 522,4;
lengthCount = 523,3;
lengthCount = 524,2;
```

```
        lengthCount = 525,5;
        lengthCount = 526,3;
        lengthCount = 527,3;
        lengthCount = 528,2;
        lengthCount = 529,8;
        lengthCount = 530,7;
        lengthCount = 531,7;
        lengthCount = 532,2;
        lengthCount = 533,3;
        lengthCount = 534,1;
        lengthCount = 536,1;
}
qualityHistogram
{
        qualityBinCount = 0,847;
        qualityBinCount = 6,17;
        qualityBinCount = 7,3080;
        qualityBinCount = 8,8362;
        qualityBinCount = 9,5477;
        qualityBinCount = 10,1276;
        qualityBinCount = 11,93696;
        qualityBinCount = 12,20247;
        qualityBinCount = 13,71436;
        qualityBinCount = 14,19426;
        qualityBinCount = 15,58305;
        qualityBinCount = 16,42293;
        qualityBinCount = 17,32671;
        qualityBinCount = 18,112311;
        qualityBinCount = 19,45738;
        qualityBinCount = 20,67565;
        qualityBinCount = 21,126896;
        qualityBinCount = 22,118779;
        qualityBinCount = 23,46687;
        qualityBinCount = 24,35607;
        qualityBinCount = 25,95486;
        qualityBinCount = 26,87452;
        qualityBinCount = 27,101505;
        qualityBinCount = 28,70254;
        qualityBinCount = 29,57202;
        qualityBinCount = 30,169097;
        qualityBinCount = 31,37174;
        qualityBinCount = 32,348894;
        qualityBinCount = 33,23706;
        qualityBinCount = 34,103531;
        qualityBinCount = 35,38321;
        qualityBinCount = 36,9086;
        qualityBinCount = 37,7207;
        qualityBinCount = 38,5449;
        qualityBinCount = 39,2286;
        qualityBinCount = 40,9260;
}

region = 1;
key    = TCA;

numberReads    = 688270;
totalBases     = 286727561;
averageLength  = 416.592,123.032;
averageQuality = 27.6127,7.27458;
```

```
       lengthHistogram
       {
              lengthCount = 10,1;
              lengthCount = 11,1;
              lengthCount = 29,1;
              lengthCount = 32,1;
[…]
              lengthCount = 603,1;
              lengthCount = 606,1;
              lengthCount = 607,1;
              lengthCount = 642,1;
       }
       qualityHistogram
       {
              qualityBinCount = 0,63892;
              qualityBinCount = 6,17868;
              qualityBinCount = 7,116822;
              qualityBinCount = 8,240702;
[…]
              qualityBinCount = 37,5451684;
              qualityBinCount = 38,1698094;
              qualityBinCount = 39,746023;
              qualityBinCount = 40,6276731;
       }

}

regionKey
{
       region = 2;
       key    = CAT;

       numberReads    = 5473;
       totalBases     = 1797367;
       averageLength  = 328.406,118.544;
       averageQuality = 24.7324,7.15332;

       lengthHistogram
       {
              lengthCount = 47,1;
              lengthCount = 48,2;
              lengthCount = 49,5;
              lengthCount = 50,6;
[…]
              lengthCount = 528,3;
              lengthCount = 530,10;
              lengthCount = 531,5;
              lengthCount = 536,1;
       }
       qualityHistogram
       {
              qualityBinCount = 0,678;
              qualityBinCount = 6,11;
              qualityBinCount = 7,2763;
              qualityBinCount = 8,7068;
[…]
              qualityBinCount = 37,7136;
              qualityBinCount = 38,3931;
              qualityBinCount = 39,1721;
              qualityBinCount = 40,5947;
       }

       region = 2;
       key    = TCA;
```

```
        numberReads    = 709153;
        totalBases     = 289420953;
        averageLength  = 408.122,126.837;
        averageQuality = 27.3364,7.29025;

        lengthHistogram
        {
                lengthCount = 36,1;
                lengthCount = 37,3;
                lengthCount = 38,9;
                lengthCount = 39,21;
[…]
                lengthCount = 595,1;
                lengthCount = 596,1;
                lengthCount = 597,2;
                lengthCount = 601,1;
        }
        qualityHistogram
        {
                qualityBinCount = 0,59062;
                qualityBinCount = 6,20889;
                qualityBinCount = 7,135008;
                qualityBinCount = 8,270877;
[…]
                qualityBinCount = 37,4297328;
                qualityBinCount = 38,1683005;
                qualityBinCount = 39,733672;
                qualityBinCount = 40,5473350;
        }

}
```

13.4.3 454RuntimeMetricsAll.txt

Example file listing for 454RuntimeMetricsAll.txt (truncated, as shown by bracketed ellipsis).

```
/****************************************************************************
**
**       454 Life Sciences Corporation
**
**       Software Release: 2.0.00.17
**
**       Runtime Metrics Results
**
**       Run Name:      R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli
**       Analysis Name: D_2008_08_19_18_48_14_dragonforce_fullProcessing
**       Region Name:   All
**       Key Sequence:  All
**
**       File Created:  2008/05/29 07:49:18
**
****************************************************************************/

softwareVersion
{
       softwareVersionTag = “2.0.00.17”;
}

/*
**       Run Conditions
*/
runConditions
{
       dateOfFile = “2008/05/29 07:49:18”;

       runName      = “R_2008_05_29_07_49_18_FLX12070284_lmcdade_200xRev10G70xVFRxEcoli”;
       analysisName = “D_2008_08_19_18_48_14_dragonforce_fullProcessing”;

       PTPType      = “unknown”;
       PTPBarCode   = “701659”;
       analysisType = “<analysisType goes here>”;

       numberOfRegions = 2;
       numberOfCycles  = -1;

       flowOrder = “<flowOrder goes here>”;
}

/*
**       Region ‘1’
*/
region
{
       name = “1”;

       rawWells     = 1021913;
       keyPassWells = 1021913;

       /*
       **       Key ‘CATG’
       */
       key
       {
              keySequence  = “CATG”;
              keyPassWells = 8205;
              /*
```

```
**     Values are:
**
**     Average, Standard Deviation
*/
keySignalPerBase = 356.639000, 122.004000;

ppi1 = 106.825000, 31.010900;
ppi2 = 71.587900, 14.046000;
ppi3 = 55.368700, 11.994000;

singletA = 0.967696, 0.227894;
singletT = 0.996231, 0.211289;
singletG = 1.023410, 0.230903;
singletC = 1.019620, 0.220346;

singletN = 1.004150, 0.223871;

/*
**     Sequence Metrics for Templates
*/
matchSequenceMetrics
{
       /*
       **  Sequence 'all'
       */
       sequence
       {
              name = "all";

              /*
              **  Values are:
              **
              **  Accuracy, Length, Percent of keypass, Number of wells
              */
              wellsAtQuality = 100.000000, 100, 56.514321, 4637;
              wellsAtQuality = 99.000000, 100, 69.762340, 5724;
              wellsAtQuality = 98.000000, 100, 74.747105, 6133;
              wellsAtQuality = 95.000000, 100, 80.938452, 6641;
              wellsAtQuality = 80.000000, 100, 91.346740, 7495;

              wellsAtQuality = 100.000000, 150, 50.408288, 4136;
              wellsAtQuality = 99.000000, 150, 64.290067, 5275;
              wellsAtQuality = 98.000000, 150, 72.845826, 5977;
              wellsAtQuality = 95.000000, 150, 78.598416, 6449;
              wellsAtQuality = 80.000000, 150, 89.750152, 7364;

              wellsAtQuality = 100.000000, 200, 44.156002, 3623;
              wellsAtQuality = 99.000000, 200, 65.386959, 5365;
              wellsAtQuality = 98.000000, 200, 71.042048, 5829;
              wellsAtQuality = 95.000000, 200, 77.659963, 6372;
              wellsAtQuality = 80.000000, 200, 87.714808, 7197;

              wellsAtQuality = 100.000000, 250, 24.911639, 2044;
              wellsAtQuality = 99.000000, 250, 54.076782, 4437;
              wellsAtQuality = 98.000000, 250, 66.581353, 5463;
              wellsAtQuality = 95.000000, 250, 74.832419, 6140;
              wellsAtQuality = 80.000000, 250, 86.179159, 7071;

              wellsAtQuality = 100.000000, 300, 9.823278, 806;
              wellsAtQuality = 99.000000, 300, 42.815356, 3513;
              wellsAtQuality = 98.000000, 300, 59.024985, 4843;
              wellsAtQuality = 95.000000, 300, 72.650823, 5961;
              wellsAtQuality = 80.000000, 300, 84.765387, 6955;

              wellsAtQuality = 100.000000, 350, 5.179768, 425;
```

```
        wellsAtQuality = 99.000000, 350, 24.936015, 2046;
        wellsAtQuality = 98.000000, 350, 43.705058, 3586;
        wellsAtQuality = 95.000000, 350, 58.354662, 4788;
        wellsAtQuality = 80.000000, 350, 82.949421, 6806;

        wellsAtQuality = 100.000000, 400, 3.510055, 288;
        wellsAtQuality = 99.000000, 400, 19.865935, 1630;
        wellsAtQuality = 98.000000, 400, 35.758684, 2934;
        wellsAtQuality = 95.000000, 400, 54.357099, 4460;
        wellsAtQuality = 80.000000, 400, 80.134065, 6575;
}
/*
**      Sequence ‘AVTF100’
*/
sequence
{
        name = “AVTF100”;

        /*
        **      Values are:
        **
        **      Accuracy, Length, Percent of mapped, Number of wells
        */
        wellsAtQuality = 100.000000, 100, 52.614897, 664;
        wellsAtQuality = 99.000000, 100, 72.028526, 909;
        wellsAtQuality = 98.000000, 100, 78.209192, 987;
        wellsAtQuality = 95.000000, 100, 86.687797, 1094;
        wellsAtQuality = 80.000000, 100, 100.000000, 1262;

        wellsAtQuality = 100.000000, 150, 45.085297, 555;
        wellsAtQuality = 99.000000, 150, 66.043867, 813;
        wellsAtQuality = 98.000000, 150, 79.041430, 973;
        wellsAtQuality = 95.000000, 150, 86.433794, 1064;
        wellsAtQuality = 80.000000, 150, 100.000000, 1231;

        wellsAtQuality = 100.000000, 200, 40.329218, 490;
        wellsAtQuality = 99.000000, 200, 67.736626, 823;
        wellsAtQuality = 98.000000, 200, 75.802469, 921;
        wellsAtQuality = 95.000000, 200, 85.432099, 1038;
        wellsAtQuality = 80.000000, 200, 100.000000, 1215;

        wellsAtQuality = 100.000000, 250, 8.710218, 104;
        wellsAtQuality = 99.000000, 250, 45.142379, 539;
        wellsAtQuality = 98.000000, 250, 69.430486, 829;
        wellsAtQuality = 95.000000, 250, 81.490787, 973;
        wellsAtQuality = 80.000000, 250, 100.000000, 1194;

        wellsAtQuality = 100.000000, 300, 0.000000, 0;
        wellsAtQuality = 99.000000, 300, 17.918089, 210;
        wellsAtQuality = 98.000000, 300, 46.245734, 542;
        wellsAtQuality = 95.000000, 300, 77.815700, 912;
        wellsAtQuality = 80.000000, 300, 100.000000, 1172;

        wellsAtQuality = 100.000000, 350, 0.000000, 0;
        wellsAtQuality = 99.000000, 350, 12.065972, 139;
        wellsAtQuality = 98.000000, 350, 43.489583, 501;
        wellsAtQuality = 95.000000, 350, 76.041667, 876;
        wellsAtQuality = 80.000000, 350, 100.000000, 1152;

        wellsAtQuality = 100.000000, 400, 0.000000, 0;
        wellsAtQuality = 99.000000, 400, 13.127753, 149;
        wellsAtQuality = 98.000000, 400, 40.440529, 459;
        wellsAtQuality = 95.000000, 400, 74.096916, 841;
        wellsAtQuality = 80.000000, 400, 100.000000, 1135;
}
```

```
/*
**      Sequence ‘AVTF120’
*/
sequence
{
        name = “AVTF120”;

        /*
        **      Values are:
        **
        **      Accuracy, Length, Percent of mapped, Number of wells
        */
        wellsAtQuality = 100.000000, 100, 68.500000, 959;
        wellsAtQuality = 99.000000, 100, 77.571429, 1086;
        wellsAtQuality = 98.000000, 100, 82.428571, 1154;
        wellsAtQuality = 95.000000, 100, 88.857143, 1244;
        wellsAtQuality = 80.000000, 100, 100.000000, 1400;

        wellsAtQuality = 100.000000, 150, 54.591468, 755;
        wellsAtQuality = 99.000000, 150, 72.451193, 1002;
        wellsAtQuality = 98.000000, 150, 81.778742, 1131;
        wellsAtQuality = 95.000000, 150, 88.503254, 1224;
        wellsAtQuality = 80.000000, 150, 100.000000, 1383;

        wellsAtQuality = 100.000000, 200, 47.941176, 652;
        wellsAtQuality = 99.000000, 200, 74.485294, 1013;
        wellsAtQuality = 98.000000, 200, 81.617647, 1110;
        wellsAtQuality = 95.000000, 200, 90.294118, 1228;
        wellsAtQuality = 80.000000, 200, 100.000000, 1360;

        wellsAtQuality = 100.000000, 250, 16.317365, 218;
        wellsAtQuality = 99.000000, 250, 59.356287, 793;
        wellsAtQuality = 98.000000, 250, 77.544910, 1036;
        wellsAtQuality = 95.000000, 250, 88.922156, 1188;
        wellsAtQuality = 80.000000, 250, 100.000000, 1336;

        wellsAtQuality = 100.000000, 300, 1.599391, 21;
        wellsAtQuality = 99.000000, 300, 28.027418, 368;
        wellsAtQuality = 98.000000, 300, 65.270373, 857;
        wellsAtQuality = 95.000000, 300, 86.519421, 1136;
        wellsAtQuality = 80.000000, 300, 100.000000, 1313;

        wellsAtQuality = 100.000000, 350, 1.078582, 14;
        wellsAtQuality = 99.000000, 350, 18.412943, 239;
        wellsAtQuality = 98.000000, 350, 59.399076, 771;
        wellsAtQuality = 95.000000, 350, 83.667180, 1086;
        wellsAtQuality = 80.000000, 350, 100.000000, 1298;

        wellsAtQuality = 100.000000, 400, 0.316456, 4;
        wellsAtQuality = 99.000000, 400, 15.664557, 198;
        wellsAtQuality = 98.000000, 400, 50.395570, 637;
        wellsAtQuality = 95.000000, 400, 81.329114, 1028;
        wellsAtQuality = 80.000000, 400, 100.000000, 1264;
}
/*
**      Sequence ‘AVTF150’
*/
sequence
{
        name = “AVTF150”;
```

[…]

```
}
/*
**      Sequence ‘AVTF2’
```

```
                    */
                    sequence
                    {
                           name = “AVTF2”;

[…]
                    }
                    /*
                    **     Sequence ‘AVTF7’
                    */
                    sequence
                    {
                           name = “AVTF7”;

[…]
                    }
                    /*
                    **     Sequence ‘AVTF90’
                    */
                    sequence
                    {
                           name = “AVTF90”;

[…]
                    }
             }
      }
      /*
      **     Key ‘TCAG’
      */
      key
      {
             keySequence  = “TCAG”;
             keyPassWells = 1006733;

             /*
             **     Values are:
             **
             **     Average, Standard Deviation
             */
             keySignalPerBase = 456.229000, 234.255000;

             ppi1 = 112.406000, 32.735500;
             ppi2 = 76.168400, 13.682100;
             ppi3 = 58.838100, 11.109200;

      }
}
/*
**     Region ‘2’
*/
region
{
      name = “2”;

      rawWells     = 1102293;
      keyPassWells = 1102293;

      /*
      **     Key ‘CATG’
      */
      key
      {
             keySequence  = “CATG”;
             keyPassWells = 8070;
```

```
/*
**     Values are:
**
**     Average, Standard Deviation
*/
keySignalPerBase = 358.014000, 120.808000;

ppi1 = 100.810000, 30.335700;
ppi2 = 69.999100, 14.188000;
ppi3 = 53.900200, 12.131000;

singletA = 0.970121, 0.261892;
singletT = 1.002920, 0.245386;
singletG = 1.032410, 0.272516;
singletC = 1.025890, 0.256914;

singletN = 1.010380, 0.260756;

/*
**     Sequence Metrics for Templates
*/
matchSequenceMetrics
{
       /*
       **  Sequence ‘all’
       */
       sequence
       {
              name = “all”;

              /*
              **  Values are:
              **
              **  Accuracy, Length, Percent of keypass, Number of wells
              */
              wellsAtQuality = 100.000000, 100, 50.322181, 4061;
              wellsAtQuality = 99.000000, 100, 63.816605, 5150;
              wellsAtQuality = 98.000000, 100, 69.615861, 5618;
              wellsAtQuality = 95.000000, 100, 77.038414, 6217;
              wellsAtQuality = 80.000000, 100, 89.863693, 7252;

              wellsAtQuality = 100.000000, 150, 43.705081, 3527;
              wellsAtQuality = 99.000000, 150, 57.608426, 4649;
              wellsAtQuality = 98.000000, 150, 66.741016, 5386;
              wellsAtQuality = 95.000000, 150, 73.866171, 5961;
              wellsAtQuality = 80.000000, 150, 87.905824, 7094;

              wellsAtQuality = 100.000000, 200, 37.856258, 3055;
              wellsAtQuality = 99.000000, 200, 58.389095, 4712;
              wellsAtQuality = 98.000000, 200, 64.820322, 5231;
              wellsAtQuality = 95.000000, 200, 72.701363, 5867;
              wellsAtQuality = 80.000000, 200, 85.291202, 6883;

              wellsAtQuality = 100.000000, 250, 19.888476, 1605;
              wellsAtQuality = 99.000000, 250, 45.749690, 3692;
              wellsAtQuality = 98.000000, 250, 59.318463, 4787;
              wellsAtQuality = 95.000000, 250, 69.256506, 5589;
              wellsAtQuality = 80.000000, 250, 83.382900, 6729;

              wellsAtQuality = 100.000000, 300, 7.967782, 643;
              wellsAtQuality = 99.000000, 300, 35.724907, 2883;
              wellsAtQuality = 98.000000, 300, 51.189591, 4131;
              wellsAtQuality = 95.000000, 300, 66.629492, 5377;
              wellsAtQuality = 80.000000, 300, 81.648079, 6589;
```

```
        wellsAtQuality = 100.000000, 350, 3.791822, 306;
        wellsAtQuality = 99.000000, 350, 20.359356, 1643;
        wellsAtQuality = 98.000000, 350, 36.641884, 2957;
        wellsAtQuality = 95.000000, 350, 51.672862, 4170;
        wellsAtQuality = 80.000000, 350, 79.962825, 6453;

        wellsAtQuality = 100.000000, 400, 2.688971, 217;
        wellsAtQuality = 99.000000, 400, 15.947955, 1287;
        wellsAtQuality = 98.000000, 400, 29.566295, 2386;
        wellsAtQuality = 95.000000, 400, 47.459727, 3830;
        wellsAtQuality = 80.000000, 400, 76.418835, 6167;
}
/*
**      Sequence 'AVTF100'
*/
sequence
{
        name = "AVTF100";
```

[…]

```
}
/*
**      Sequence 'AVTF120'
*/
sequence
{
        name = "AVTF120";
```

[…]

```
}
/*
**      Sequence 'AVTF150'
*/
sequence
{
        name = "AVTF150";
```

[…]

```
}
/*
**      Sequence 'AVTF2'
*/
sequence
{
        name = "AVTF2";
```

[…]

```
}
/*
**      Sequence 'AVTF7'
*/
sequence
{
        name = "AVTF7";
```

[…]

```
}
/*
**      Sequence 'AVTF90'
*/
sequence
{
        name = "AVTF90";
```

```
[…]
                        }
                }
        }
        /*
        **      Key ‘TCAG’
        */
        key
        {
                keySequence  = “TCAG”;
                keyPassWells = 1087997;

                /*
                **      Values are:
                **
                **      Average, Standard Deviation
                */
                keySignalPerBase = 430.280000, 215.157000;

                ppi1 = 106.588000, 32.476400;
                ppi2 = 75.529900, 14.056700;
                ppi3 = 58.380900, 11.669200;

        }
}
```

13.4.4 *region.key*.454Reads.fna and *region.key*.454Reads.qual

A small portion of an example ‘1.TCA.454Reads.fna’ file is reproduced below:

```
[…]
>FA7UOGX01DU9RS rank=0000718 x=1469.5 y=934.0 length=498
GGTTTCTGAACGGCGATAGCGTCTTCAGGCGGACGCCGTTTAAAGGGATCAATAATGATG
TCATCAAGCGGATAATCGCTGGTGCCGTGGCGATACACTGAACGTTCCACCACGTTTTCA
ATTCACGAATATTTCCCGGCCAACGATAATTCAGCAATGTTTCTCTGGCGCGCTCCGTAA
ACCCGGGAACAGAGGCAGCTTGATTTCCCGACACATCTGGATGGCAAAGTATTCTGCCAT
CAACATTATGTCGCTTTCGCGCTCGCGCAGTGGTGGCAGTTGTACAACATCAAAAGCCAG
TCGGTCGAGCAGGTCAGCGCGAAAAGTGCCTTCATTGACCATCGCCGGGAGATCGGCATT
CGTCGCGCATACCAACCGCACATTCACCTGCAATGGTTGGCTGCCGCCAACGCGCTCCAG
TTCACCGTACTCAATCACGCGCAATAATTTCTCCTGCACCATCATCGGTGCCGTAGCGAG
TTCATCAAGAAATAGCCG
>FA7UOGX01BPK2A rank=0000719 x=584.0 y=1360.0 length=104
TTGCTAACACCTCTCGTCACGTCGCGAAGTCACCGTCAACTGGGAGCGAAAGATGGTCGC
CTGACGGCAGTGAAGATGGATTTCCGCGCCAACACTGGCCTTAC
>FA7UOGX01DFMYB rank=0000720 x=1291.0 y=609.0 length=398
TTTCAGTTATGATTGTGGGACTTATCAAAGGAGAGGCCATGCGTTCGATTGCCAGACGTA
CCGCAGTGGGAGCTGCACTATTGCTTGTCATGCCAGTAGCCGTATGGATTTCTGGCTGGC
GTTGGCAACCTGGAGAACAAAGTTGGCTACTAAAGCGGCTTTTGGGTTACTGAAACTGTC
ACCCAGCCCTGGGGCGTCATTACACATTTGATTTATTCGGCTGGTTCTCTGGTGTCTGCG
TTTTCGCATTAAGGCTGCCTTTGTATTATTTGCCATTCTGGCGGCCGCAATCCTTGTGGG
ACAAGGCGTTAAATCCTGGATCAAAGACAAAGTCCAGGAACCACGACCTTTTGTTATCTG
GCTGGAAAAAACACATCATATTCCGGTTGATGAGTTCT
>FA7UOGX01C0C6T rank=0000726 x=1117.0 y=819.0 length=408
TTCAGGGCGAGCATCTGATTATTGTGGCGACCGAAGACAACTTACTGAAGGCGCGGCGGC
ACAAGCGGTACAGTGCGCCAATATTCGTTTCGGCTATGCGGAAACGCAGTCTCTTATTTA
AGGGTGCAATGATGAATCCATTAATTATCAAACTGGGCGGCGTACTGCTGGATAGTGAAG
AGGCGCTGGAACGTCTGTTTAGCGCACTGGTGAATTATCGTGAGTCACATCAGCGTCCGC
TGGTGATTGTGCACGGCGGCGGTTGCGTGGTGGATGAGCTGATGAAAGGGCTGAATCTGC
CGGTGAAAAGAAACGGCCTGCGGGTGACGCCTGCTGATCAGATAGACCATTATCACCGGA
GCACTGGCGGAACGGCAAATAAACCCTGTTGGCATGGGCGAAGAACAT […]
```

The corresponding portion of the example file '1.TCA.454Reads.qual' is reproduced below:

```
[…]
>FA7UOGX01DU9RS rank=0000718 x=1469.5 y=934.0 length=498
30 30 24 24 24 35 35 33 24 24 24 33 35 35 35 35 35 35 40 40 40 40 40 40 40 40 40 40
40 40 40 40 40 37 37 37 37 37 36 31 31 24 24 24 29 29 29 33 35 35 35 35 35 35 35 35
35 35 35 35
35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35
35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 28 20 20 20
20 20 28 33
35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 33 33 33 33 33 33 33 35 35 35 35
35 35 35 35 35 35 35 35 35 35 29 24 24 24 29 33 35 35 35 35 35 35 35 35 35 33 33 33
35 35 33 16
16 15 15 15 16 16 16 15 15 14 25 33 33 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35
35 35 35 35 35 35 35 35 33 33 33 34 34 34 34 32 28 18 18 18 32 24 14 20 20 24 24 29
31 34 34 34
34 34 34 34 34 28 28 25 25 18 18 18 25 27 18 18 18 21 27 28 28 28 28 28 28 28 29 29
29 27 25 25 25 29 27 27 27 29 29 27 27 27 29 29 29 29 29 27 27 27 29 29 29 29 27 25
25 25 29 29
29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 28 29 29 25 25 25 25 25 27 25
25 25 29 29 29 29 21 21 21 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29
29 27 27 27
29 29 29 29 29 29 27 25 21 21 21 25 28 28 29 29 25 25 25 25 25 25 27 29 29 29 28 28
28 28 28 28 27 25 18 18 18 18 21 21 21 27 28 28 27 25 25 25 27 27 28 28 28 21 21 18
18 18 25 25
25 25 25 27 28 28 28 29 29 29 27 27 27 27 29 29 29 29 28 28 28 27 26 27 26 25 25 16
16 16 21 18 21 21 21 21 21 21 27 28 28 28 27 25 25 21 21 21 25 21 21 21 21 21 27 27
27 27 28 27
16 16 16 21 25 27 25 20 20 13 13 13 20 14 14 13 13 19
>FA7UOGX01BPK2A rank=0000719 x=584.0 y=1360.0 length=104
19 19 23 24 25 25 31 30 32 32 32 32 32 32 32 32 32 32 37 32 37 36 32 32 32 32 25 25
24 25 25 32 28 28 28 29 28 28 28 27 28 19 19 18 24 21 30 30 29 24 24 18 14 19 25 31
30 32 30 30
30 30 31 32 32 32 30 30 31 31 27 23 24 24 30 30 27 27 27 29 27 26 26 28 28 28 28 25
17 17 17 15 15 19 24 21 21 21 15 15 15 15 15 15
>FA7UOGX01DFMYB rank=0000720 x=1291.0 y=609.0 length=398
28 28 28 36 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 39 39 37 37 37 36 36 28 28
28 28 28 37 37 37 37 37 38 38 40 40 40 40 40 40 40 37 37 37 37 37 37 37 37 37 37 37
37 37 37 37
37 37 37 37 37 37 37 36 36 36 37 37 37 37 37 37 37 37 37 36 36 36 36 37 37 37 37 37
37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37
37 37 37 37
37 37 37 37 37 37 37 37 36 36 36 37 37 37 36 32 26 26 26 26 21 23 30 37 37 37 37 37
37 35 23 23 23 23 33 32 37 37 32 17 17 17 17 30 19 19 15 15 21 32 32 30 23 23 23 35
37 37 37 37
37 37 37 37 36 33 23 23 23 31 22 22 22 22 31 35 37 37 37 37 37 37 37 37 37 37 37 36
36 35 35 28 28 26 22 36 36 36 36 36 36 36 19 22 24 24 27 27 36 36 36 36 36 36 36 36
36 36 36 26
22 22 22 24 27 28 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
20 20 20 28 28 30 31 31 31 31 31 31 31 31 31 31 31 30 30 30 31 31 31 30 30 25 25 22
22 18 18 18
28 28 22 22 22 22 30 31 31 31 31 31 31 30 30 30 31 31 31 31 31 31 31 31 31 31 30 30
30 30 30 31 31 31 31 31 31 31 31 31 25 25 25 25 25 25 25 28 30 30 30 30 31 26 26 28
28 25 22 22
22 25 28 28 28 26 28 28 28 28 26 25 21 28 28 31 31 31 31 28 28 28 31 31 28 28 28 28
31 30 30 30 31 31 31 31 31 31
>FA7UOGX01C0C6T rank=0000726 x=1117.0 y=819.0 length=408
32 32 32 28 21 21 21 31 29 32 32 32 32 33 33 32 32 32 32 32 30 32 32 37 32 32 32 33
30 33 30 30 27 23 23 23 28 30 27 27 27 19 19 17 27 30 30 30 31 27 28 28 30 32 32 32
32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 30 31 31 31 30 30 30 27 23 18
18 18 27 28 28 28 28 28 28 30 30 23 23 18 18 18 23 27 30 29 28 28 28 28 25 28 25 26
16 16 16 16
16 16 16 16 17 21 28 28 29 29 29 30 30 31 31 31 31 31 31 31 17 15 15 15 20 20 28 28
22 13 13 13 15 25 15 15 15 25 21 22 26 30 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32
```

```
32 32 32 32 32 32 32 32 32 32 32 32 32 32 30 26 21 15 15 15 21 26 32 30 30 30 30 32
29 32 32 32 32 32 32 32 26 26 26 29 29 29 29 29 29 29 27 26 26 26 29 29 29 29 29 26
26 26 29 31
31 31 31 31 31 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22
22 21 21 21 21 22 22 22 22 21 21 21 21 21 21 18 13 13 13 13 13 20 20 21 21 18 18 18
20 21 21 22
22 22 21 21 21 14 14 14 14 22 15 15 15 22 18 18 18 18 22 22 22 22 22 22 22 18 22 22
22 22 22 22 22 22 22 22 22 22 22 21 20 20 18 18 13 13 13 13 18 18 20 20 18 18 18 20
21 22 22 21
19 20 18 18 18 18 18 11 11 11 16 11 11 11 11 15 11 11 11 11 8  8  8  14 18 18 18 18
18 18 21 22 22 22 22 22 22 22 18 14 18 18 21 19 16 18 11 11 […]
```

13.4.5 Consensus Basecalled Contig Files

A small portion of an example '454AllContigs.fna' file is reproduced below:

```
>contig00001  length=52472   numreads=7978
CGACTTTAAAGAATTGAACTAAAAATTCAAAAAGCAGTATTTCGGCGAGTAGCGCAGCTT
GGTAGCGCAACTGGTTTGGGACCAGTGGGTCGGAGGTTCGAATCCTCTCTCGCCGACCAA
TTTTGAACCCCGCTTCGGCGGGGTTTTTtGTTTTCTGTGCATTTCGTCACCCTCCCTTCG
CAATAAACGCCCGTAATAACTCATTGCCCCACGGTATGATTTCGCCCTTAACGTATTGAA
[…]
CCCGATCGCTGCGCGGTCATTGCCGGGGCGCTGCGTTACCCACGTTATCGCTGGTACGAC
CGTTTTATGATCAAGCTGATTATGAAGATGTCAGGCGGTGAAACGGATACGCGCAAAGAA
GTTGTCTATACCGATTGGGAGCAGGTGGCGAATTTCGCCCGAGAAATCGCCCATTTAACC
GACAAACCGACGCTGAAATAAGCATAAAGAAT
>contig00002  length=274   numreads=319
TGAGCAGTAAAACCTCTACAGGCTTGTAGCTCAGGTGGTTAGAGCGCACCCCTGATAAGG
GTGAGGTCGGTGGTTCAAGTCCACTCAGGCCTACCAAATTTGCACGGCAAATTTGAAGAG
GTTTTAACTACATGTTATGGGGCTATAGCTCAGCTGGGAGAGCGCCTGCTTTGCACGCAG
GAGGTCTGCGGTTCGATCCCGCATAGCTCCACCATCTCTGTAGTGaTTAAATAAAAAATA
CTTCAGAGTGTACCTGCAAAGGTTCACTGCGAAG
>contig00003  length=175796   numreads=25215
CCcTTATCAGCAAGGAAGGGACAACAGAATGGTGCGCAGAATACGATGAATGGGGCAACC
TGCTGAATGAAGAGAACCCGCATCAGCTGCAGCAGCTTATCCGCCTGCCGGGGCAGCAGT
ATGATGAGGAGTCCGGCCTGTATTACAACCGCCACCGCTATTATGACCCGCTGCAGGGGC
GGTATATCACTCAGGATCCGATTGGGCTGAAGGGGGgATGGAATTTTTATCAGTATCCGT
[…]
ATCTCACCGGGCAAAATGGTGCCGGGTTCATATTCACCTTTTAAGATCCGTTGCGCCAGC
TTCTCAGCCAGAACATACGAAAGGTTTTTCTGTGCAGCTAACTGTTGTGCGCTTAAAGGC
ATTACTTATCTTCCTTTTtCTTTTTATTCCTCCTTAGTATGCCACCAGGAAGTGTGATTA
CGGTTGCAAAAACGGCAAATTGCTTGTTTTATGGCACATTAACGGGGCTTTTGCTG
> contig00004  length=105468   numreads=15535
GGATGCACTTCCGTTTATACCAAAATGTGTCGCGAGAAAATATTAGTGATGCGATGACTG
GACTGTATAATCGCAAAATTTTAACCCCTGAACTGGAGCAGCGGTTGCAGAAACTGGTGC
AATCCGGTTCTTCGGTGATGTTTATTGCTATTGACATGGACAAGTTAAAGCAAATAAATG
ACACCCTCGGTCATCAGGAGGGGGATTTAGCGATTACGTTATTAGCTCAGGCGATTAAAC
[…]
AAAAAATAATTAGCATTAGAATAGTTGCGATAAGCTGCAATAAGCAGAACCACCTTTTTG
GTTTAATATGTCCTTACAAATAGAAATGGGTCTTTACACTTATCTAAGATTTTTCCTAAA
TCGACGCAACTGTACTCGTCACTACACGCACATACAACGGAGGGGGgCTGCGATTTTCAA
TAATGCGTGATGCAGATCACACAAAACACTCAATTACTTAACATAAAT
>contig00005  length=164140   numreads=24046
GTGCATCCGCACCATATTCAGCAAAATTAACGCCGTCAAAATCCAGAACAGAACGCTTTT
CCACGAAGACGTGATAAAGTAAAGAATATCAATGGATAACGAGACACGAATGCCGCCCGG
TAAGTCATGGACGTAACTGACATACTGGAACAGATTATCTTCGCTCTGGTTGATGATAAT
GTCGCGCCCCGAATCGGTATCGGTTAGCGTGACATTGAGAAAACGCCAGAGGAGAGGGCG
[…]
CTGCTGGGCGGCTTTTTTGGCCTGAGCGCGGGCAATAGCCGCGGCAACTGCCGCTTTGCG
CGGATCAACCTGTTCTTCTGGTACCGCATTAGCCTGTTGCTGTTCCAGCTTGCGTGCTTT
GGCGCGTGCAATAGCCGCTTCGACGGCAGCTTTGCGCGGATCAACCTGTTCTTCTGGCTC
CGCATTAGCCGGTTGCTGTTCCCGTTTGCGCGCTTTGGCT
>contig00006  length=6673   numreads=804
CCGCACTCCGCGCTGGCGCCGTCCGGGCAGGCGGTGTGGCTGGAGATCGCCCGCCAGTCG
```

```
GCGCGCGCGGTCAGCGAGCTGGATAGCCGCGGCATCACCACGCGGGATATCCTCTCCGAT
AAAGCCATCGAAAACGCGATGGTGATCCACGCGGCGTTCGGCGGCTCCACCAATTTACTG
CTGCACATTCCGGCCATCGCCCACGCGGCGGGCTGCACGATCCCGGACGTTGAGCACTGG
[…]
TATCTGCCTCCGATTCTCTGCAGAAGCAGAAAGACATTGGATCGAATTCTACAACCAGGT
CGAGTCAGAAATGAGAATGATTGGCCTTCTTTATGATTTTAAGGATTATGCTTCTAAAAT
GGCGGAGAACATGGCGAGGCTTGCTGCCTTACTTCATTACTTCAGCGGTGATGGAGGCGA
TATATCTGTTACC[…]
```

A small portion of the corresponding example file '454AllContigs.qual' is reproduced below:

```
>contig00001  length=52472   numreads=7978
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
25 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 […]
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64>contig00002  length=274   numreads=319
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 39 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64
>contig00003  length=175796   numreads=25215
64 64 20 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 21 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 […]
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
```

```
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 25 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
>contig00004  length=105468   numreads=15535
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 […]
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 61 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 25 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
>contig00005  length=164140   numreads=24046
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 […]
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 61 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 56 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64
>contig00006  length=6673   numreads=804
40 40 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
```

```
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 […]
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 […]
```

13.4.6 454NewblerMetrics.txt File for the GS *De Novo* Assembler Application

Example file listing for 454NewblerMetrics.txt produced by the GS *De Novo* Assembler application (including one Paired End Run). Note the different output compared with the file of the same name produced by the Mapping application (section 13.4.9).

```
/****************************************************************************
**
**      454 Life Sciences Corporation
**         Newbler Metrics Results
**
**      Date of Assembly: 2008/08/24 22:09:13
**      Project Directory: /home/adminrig/working/TiRuns/R_2008_05_29_07_49_18_
FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/test
**      Software Release: 2.0.00.17
**
****************************************************************************/

/*
**  Input information.
*/

runData
{
       file
       {
              path = “/home/adminrig/working/TiRuns/R_2008_05_29_07_49_18_
FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/D_2008_08_20_20_12_38_dragonforce_fullPro-
cessing/sff/FA7UOGX01.sff”;

              numberOfReads = 688270, 688198;
              numberOfBases = 286727561, 273295575;
       }

}

pairedReadData
{
       file
       {
              path = “/home/adminrig/working/TiRuns/R_2008_05_29_07_49_18_
FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/pairedEndReads/ERESL0I01.sff”;
```

```
              numberOfReads = 298149, 445911;
              numberOfBases = 74002163, 60589314;
              numWithPairedRead = 161362;
       }

}

/*
**  Operation metrics.
*/

runMetrics
{
       totalNumberOfReads = 1134109;
       totalNumberOfBases = 333884889;

       numberSearches   = 26843;
       seedHitsFound    = 65603278, 2443.96;
       overlapsFound    = 5854356, 218.10, 8.92%;
       overlapsReported = 5175097, 192.79, 7.89%;
       overlapsUsed     = 1734054, 64.60, 33.51%;
}

readAlignmentResults
{
       file
       {
              path = "/home/adminrig/working/TiRuns/R_2008_05_29_07_49_18_
FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/D_2008_08_20_20_12_38_dragonforce_fullPro-
cessing/sff/FA7UOGX01.sff";

              numAlignedReads     = 684540, 99.47%;
              numAlignedBases     = 270756964, 99.07%;
              inferredReadError  = 0.53%, 1431043;
       }

}

pairedReadResults
{
       file
       {
              path = "/home/adminrig/working/TiRuns/R_2008_05_29_07_49_18_
FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/pairedEndReads/ERESL0I01.sff";

              numAlignedReads     = 443334, 99.42%;
              numAlignedBases     = 59354438, 97.96%;
              inferredReadError  = 0.45%, 268437;

              numberWithBothMapped  = 145502;
              numWithOneUnmapped    = 694;
              numWithMultiplyMapped = 14526;
              numWithBothUnmapped   = 640;
       }

}

/*
** Consensus distribution information.
*/
consensusDistribution
{
       fullDistribution
       {
```

```
signalBin =  0.0, 258212;
signalBin =  0.5, 2;
signalBin =  0.6, 9;
signalBin =  0.7, 80;
signalBin =  0.8, 57467;
signalBin =  0.9, 2112296;
signalBin =  1.0, 1017835;
signalBin =  1.1, 2651;
signalBin =  1.2, 58;
signalBin =  1.3, 5;
signalBin =  1.4, 2;
signalBin =  1.5, 111;
signalBin =  1.6, 93;
signalBin =  1.7, 1243;
signalBin =  1.8, 75709;
signalBin =  1.9, 559631;
signalBin =  2.0, 220535;
signalBin =  2.1, 3816;
signalBin =  2.2, 51;
signalBin =  2.3, 7;
signalBin =  2.4, 2;
signalBin =  2.5, 111;
signalBin =  2.6, 100;
signalBin =  2.7, 1730;
signalBin =  2.8, 29523;
signalBin =  2.9, 117070;
signalBin =  3.0, 56202;
signalBin =  3.1, 2705;
signalBin =  3.2, 46;
signalBin =  3.3, 8;
signalBin =  3.4, 11;
signalBin =  3.5, 69;
signalBin =  3.6, 281;
signalBin =  3.7, 2320;
signalBin =  3.8, 12375;
signalBin =  3.9, 25579;
signalBin =  4.0, 12635;
signalBin =  4.1, 1322;
signalBin =  4.2, 85;
signalBin =  4.3, 10;
signalBin =  4.4, 4;
signalBin =  4.5, 72;
signalBin =  4.6, 297;
signalBin =  4.7, 1119;
signalBin =  4.8, 2979;
signalBin =  4.9, 5681;
signalBin =  5.0, 5211;
signalBin =  5.1, 2005;
signalBin =  5.2, 290;
signalBin =  5.3, 48;
signalBin =  5.4, 15;
signalBin =  5.5, 36;
signalBin =  5.6, 121;
signalBin =  5.7, 353;
signalBin =  5.8, 790;
signalBin =  5.9, 1465;
signalBin =  6.0, 1456;
signalBin =  6.1, 757;
signalBin =  6.2, 201;
signalBin =  6.3, 34;
signalBin =  6.4, 6;
signalBin =  6.5, 20;
signalBin =  6.6, 41;
signalBin =  6.7, 102;
signalBin =  6.8, 226;
```

```
                signalBin =  6.9, 265;
                signalBin =  7.0, 291;
                signalBin =  7.1, 214;
                signalBin =  7.2, 93;
                signalBin =  7.3, 32;
                signalBin =  7.4, 7;
                signalBin =  7.5, 10;
                signalBin =  7.6, 25;
                signalBin =  7.7, 25;
                signalBin =  7.8, 44;
                signalBin =  7.9, 41;
                signalBin =  8.0, 62;
                signalBin =  8.1, 40;
                signalBin =  8.2, 19;
                signalBin =  8.3, 5;
                signalBin =  8.4, 6;
                signalBin =  8.5, 2;
                signalBin =  8.7, 2;
                signalBin =  8.8, 7;
                signalBin =  8.9, 4;
                signalBin =  9.0, 2;
                signalBin =  9.1, 4;
                signalBin =  9.2, 2;
                signalBin =  9.3, 1;
                signalBin = 10.2, 1;
                signalBin = 11.1, 1;
        }

        distributionPeaks
        {
                signalPeak = 1, 0.98;
                signalPeak = 2, 1.96;
                signalPeak = 3, 2.96;
                signalPeak = 4, 3.94;
                signalPeak = 5, 4.96;
                signalPeak = 6, 5.98;
                signalPeak = 7, 6.98;
        }

        thresholdsUsed
        {
                threshold = 0, 1, 0.48;
                threshold = 1, 2, 1.38;
                threshold = 2, 3, 2.40;
                threshold = 3, 4, 3.36;
                threshold = 4, 5, 4.40;
                threshold = 5, 6, 5.44;
                threshold = 6, 7, 6.46;

                interpolationAmount = 1.00;
        }
}


/*
**  Consensus results.
*/
```

```
consensusResults
{
       readStatus
       {
              numAlignedReads    = 1127874, 99.45%;
              numAlignedBases    = 330111402, 98.87%;
              inferredReadError = 0.51%, 1699480;

              numberAssembled = 1094510;
              numberPartial   = 8326;
              numberSingleton = 6235;
              numberRepeat    = 23923;
              numberOutlier   = 1115;
              numberTooShort  = 0;
       }

       pairedReadStatus
       {
              numberWithBothMapped   = 145502;
              numberWithOneUnmapped  = 694;
              numberMultiplyMapped   = 14526;
              numberWithBothUnmapped = 640;

              library
              {
                     libraryName     = “ERESL0I01.sff”;
                     pairDistanceAvg = 3594.9;
                     pairDistanceDev = 898.7;
              }
       }

       scaffoldMetrics
       {
              numberOfScaffolds   = 8;
              numberOfBases       = 4607648;

              avgScaffoldSize     = 575956;
              N50ScaffoldSize     = 2497739;
              largestScaffoldSize = 2497739;
       }

       largeContigMetrics
       {
              numberOfContigs   = 212;
              numberOfBases     = 4618284;

              avgContigSize     = 21784;
              N50ContigSize     = 89877;
              largestContigSize = 267929;

              Q40PlusBases      = 4602925, 99.67%;
              Q39MinusBases     = 15359, 0.33%;
       }

       allContigMetrics
       {
              numberOfContigs = 645;
              numberOfBases   = 4774786;
       }
}
```

13.4.7 454ReadStatus.txt

A small portion of two examples '454ReadStatus.txt' files generated by the GS *De Novo* Assembler (for Shotgun and Paired End reads) are reproduced below (truncated, as shown by the bracketed ellipses):

Example for Shotgun reads

```
Accno            Read Status
FA7UOGX01BOOGQ  Assembled
FA7UOGX01D1EU0  Assembled
FA7UOGX01DSWDG  TooShort
FA7UOGX01BZXFB  Assembled
FA7UOGX01BMD92  Assembled
FA7UOGX01DK9HV  Assembled
FA7UOGX01A9K87  Assembled
FA7UOGX01BMBBG  Assembled
FA7UOGX01CIMW2  Assembled
FA7UOGX01BA6G3  Assembled
FA7UOGX01C6BK1  Singleton
FA7UOGX01CVQBG  Assembled
FA7UOGX01CSJGQ  TooShort
FA7UOGX01A0MV6  TooShort
FA7UOGX01CZ0E4  Assembled
FA7UOGX01A5ML6  Assembled
FA7UOGX01CYLW9  TooShort
FA7UOGX01CP9TE  Assembled
FA7UOGX01CF9AY  PartiallyAssembled
FA7UOGX01CBX29  TooShort
FA7UOGX01BQD2S  Assembled
FA7UOGX01DA158  TooShort
FA7UOGX01BSV9A  Assembled
FA7UOGX01BR90M  Assembled
FA7UOGX01DR3NW  TooShort
FA7UOGX01BDI7S  Assembled
FA7UOGX01B5S6G  Assembled
FA7UOGX01BRXJW  Assembled
FA7UOGX01BBU6R  Repeat
FA7UOGX01DLVHM  TooShort
FA7UOGX01CER97  Repeat
FA7UOGX01DVLW5  Repeat
FA7UOGX01B3Y3N  Assembled
FA7UOGX01BNGZA  Assembled
FA7UOGX01BNJU8  Repeat
FA7UOGX01DTPUE  Assembled
FA7UOGX01CPTUH  TooShort
FA7UOGX01CYCKT  Assembled
FA7UOGX01CZ0LW  Assembled
FA7UOGX01CV5SS  Assembled
FA7UOGX01B3PB3  Assembled
FA7UOGX01BHZ9D  Assembled
FA7UOGX01CZXJJ  Singleton
FA7UOGX01CDJEC  Repeat
FA7UOGX01CP52V  PartiallyAssembled
FA7UOGX01EBOFU  Assembled
FA7UOGX01CFM5N  Assembled
FA7UOGX01CH3LO  Assembled
FA7UOGX01DZKFM  Assembled
FA7UOGX01BHD76  Assembled
FA7UOGX01DR34E  Repeat
```

```
FA7UOGX01CP9TH  Assembled
FA7UOGX01CUAW5  Assembled
FA7UOGX01D1LDC  TooShort
FA7UOGX01CY74T  Assembled
FA7UOGX01BV443  Assembled
FA7UOGX01CMQ7I  Repeat
FA7UOGX01BV9GG  Assembled
FA7UOGX01CDJDN  Assembled
FA7UOGX01DZ08M  Assembled
FA7UOGX01A1UPP  Assembled
FA7UOGX01CQPY2  Assembled
FA7UOGX01B5G14  TooShort
FA7UOGX01A0MG3  Repeat
FA7UOGX01CZKBH  Assembled
FA7UOGX01DAWRP  TooShort
FA7UOGX01EBLK0  Assembled
FA7UOGX01CRXCQ  Assembled
FA7UOGX01EGSC9  Assembled
FA7UOGX01CO1G5  Assembled
FA7UOGX01DIMJ9  Assembled
FA7UOGX01DRUDS  Assembled
FA7UOGX01DUD85  Assembled
FA7UOGX01C5JPY  Assembled
FA7UOGX01CII37  Repeat
FA7UOGX01BDL9X  Assembled
FA7UOGX01B5NNV  Assembled
FA7UOGX01AYFVV  Assembled
FA7UOGX01DQDIU  Singleton
FA7UOGX01CWOCG  Assembled
FA7UOGX01B9X6T  Repeat
FA7UOGX01BT1ES  Assembled
FA7UOGX01CTFFT  Assembled
FA7UOGX01C1XXO  Assembled
FA7UOGX01BPTMC  Assembled
FA7UOGX01EE74K  Assembled[…]
```

Example for Paired End reads

```
ERESL0I01AIXO4_right  Assembled
ERESL0I01AG3II_left   Assembled
ERESL0I01AG3II_right  Assembled
ERESL0I01E10AC        Assembled
ERESL0I01EGSWO_left   Repeat
ERESL0I01EGSWO_right  Assembled
ERESL0I01EVB77        PartiallyAssembled
ERESL0I01AMH5Y        Assembled
ERESL0I01EG704        Assembled
ERESL0I01ADZ9L_left   Assembled
ERESL0I01ADZ9L_right  Assembled
ERESL0I01ANKQK_left   Assembled
ERESL0I01ANKQK_right  Assembled
ERESL0I01AJS2W        Assembled
ERESL0I01AE4OU        Assembled
ERESL0I01AEPBZ_left   Singleton
ERESL0I01AEPBZ_right  Assembled
ERESL0I01DDCQR_left   Assembled
ERESL0I01DDCQR_right  Assembled
ERESL0I01DZRGD_left   Singleton
ERESL0I01DZRGD_right  Assembled
ERESL0I01ANA35_left   Assembled
```

```
ERESL0I01ANA35_right  Assembled
ERESL0I01AHJJH_left   Repeat
ERESL0I01AHJJH_right  Repeat
ERESL0I01ETEZ6_left   Assembled
ERESL0I01ETEZ6_right  Assembled
ERESL0I01AKH9F_left   Assembled
ERESL0I01AKH9F_right  Assembled
ERESL0I01EERCY_left   Assembled
ERESL0I01EERCY_right  Assembled
ERESL0I01CZ64N_left   Assembled
ERESL0I01CZ64N_right  Assembled
ERESL0I01AK1CQ        Outlier
ERESL0I01A85RX        Assembled
ERESL0I01AM4OU        Assembled
ERESL0I01AGDZF_left   Assembled
ERESL0I01AGDZF_right  Assembled
ERESL0I01AIATR        Assembled
ERESL0I01AGSWD        Assembled
ERESL0I01AHF32_left   Assembled
ERESL0I01AHF32_right  Assembled
ERESL0I01ELZ0V        Assembled
ERESL0I01DJ7GD        Assembled
ERESL0I01C7Z7X_left   Assembled
ERESL0I01C7Z7X_right  Assembled
ERESL0I01E3726_left   Assembled
ERESL0I01E3726_right  Assembled
ERESL0I01EJH8M        Assembled
ERESL0I01AJPW1        Assembled
ERESL0I01D4D77_left   Assembled
ERESL0I01D4D77_right  Assembled
ERESL0I01DZQXY        Singleton
ERESL0I01EP9JD        Assembled
ERESL0I01AD5XQ        Assembled
ERESL0I01EN44U_left   Assembled
ERESL0I01EN44U_right  Assembled
ERESL0I01ENWGR_left   Assembled
ERESL0I01ENWGR_right  Assembled […]
```

13.4.8 454AllDiffs.txt

Example file listing for 454AllDiffs.txt (truncated, as shown by the bracketed ellipsis):

```
>Reference       Start   End      Ref     Var      Total   Var
>Accno            Pos    Pos      Nuc     Nuc      Depth   Freq
______________________________

>gi|49175990|ref|NC_000913.2|     19530   19530    T       -        42       7%

Reads with Difference:
gi|49175990|ref           19501+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 19556
                                                               *
FA7UOGX01DK4M4              236- GTCGAGAATGTGATGGAAGA-GTACCATGC-ATTTT-T-G-CTGA-GCGGATGATTCAGCA 182
FA7UOGX01A5E5Z              433- GTCGAGAATGTGATGGAAGA-GTACCATGC-ATTTT-T-G-CTGA-GCGGATGATTCAGCA 379
FA7UOGX01DQF00              244-           TGATGGAAGAAGTACCATGC-ATTTT-TTG-CTGA-GCGGATGATTCAGCA 198
                                                               *

Other Reads:
                                                               *
FA7UOGX01DNS4X               84- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT---G-CTGA-GCGGATGATTCAGCA 30
FA7UOGX01C7617              116- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT---G-CTGA-GCGGATGATTCAGCA 62
FA7UOGX01A11JO              376+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 431
FA7UOGX01EXG9R  (3)          92- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 37
FA7UOGX01DZ7ML  (2)         130- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 75
FA7UOGX01B4NI4              161- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 106
FA7UOGX01CNPUC              343+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 398
FA7UOGX01AVWGJ  (2)         186- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 131
```

```
FA7UOGX01EHJ54              337+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGAN-CGGATGATTCAGCA 392
FA7UOGX01BOM7F               90- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT---G-CTGA-GCGGATGATTCAGCA 36
FA7UOGX01ELCDS  (2)         253- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 198
FA7UOGX01BPN5S              204- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 149
FA7UOGX01B1WNO  (2)         278+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 333
FA7UOGX01BTYWG              260- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 205
FA7UOGX01DJTZL              257+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-TTG-CTGA-GCGGATGATTCAGCA 313
FA7UOGX01DP221              247+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 302
FA7UOGX01CX78S              226+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 281
FA7UOGX01CSJ02              162- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT---G-CTGA-GCGGATGATTCAGCA 108
FA7UOGX01ELZBV              204+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT---G-CTGA-GCGGATGATTCAGCA 258
FA7UOGX01A2L5T              277- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 222
FA7UOGX01DS08E  (2)         175+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 230
FA7UOGX01D52O5              172+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT---G-CTGA-GCGGATGATTCAGCA 226
FA7UOGX01EYTA1              165- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-TTG-CTGA-GCGGATGATTCAGCA 109
FA7UOGX01EEFT0              161+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 216
FA7UOGX01C9ZS6  (2)         373- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 318
FA7UOGX01COLQH              103+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 158
FA7UOGX01DFWYF              350- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 295
FA7UOGX01CT9W5              399- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 344
FA7UOGX01EKS5T              408- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 353
FA7UOGX01BMAKV              370- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 315
FA7UOGX01B5RCC               52+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 107
FA7UOGX01C0NN5              294- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT---G-CTGA-GCGGATGATTCAGCA 240
FA7UOGX01DGUAF              428- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-GTCTGA-GCGGATGATTCAGCA 372
FA7UOGX01EVWR1               45+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 100
FA7UOGX01B09PV              444- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-TTG-CTGA-GCGGATGATTCAGCA 388
FA7UOGX01BQ6GT              403- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 348
FA7UOGX01DWTT0              312- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 257
FA7UOGX01EBGB8               15+ GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTTGT---CTGA-GCGGA           60
FA7UOGX01CEJGP              472- GTCGAGAATGTGATGGAAGA-GTACCATGCTATTTT-T-G-CTGA-GCGGATGATTCAGCA 417
                                                               *


-----------------------------

>gi|49175990|ref|NC_000913.2|     39630   39630    -       G       50       12%

Reads with Difference:
gi|49175990|ref           39603+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 39655
                                                            *

FA7UOGX01AU017              273+ AGCGCGCCGCATCTTCAAAGGCGCACAGT-CGCCG-TACCGTAGTTGGTGAGGG-CT- 326
FA7UOGX01AQ3NU              191+ AGCGCGCCGCATCTTCAAAGGCGCACAGT-CGCCG-TACCGTAGTTGGTGAGGG-CT- 244
FA7UOGX01AHRH4              167+ AGCGCGCCGCATCTTCAAAGGCGCACAGT-CGCCG-TACCGTAGTTGGTGAGGG-CTT 221
FA7UOGX01AMTIR              166+ AGCGCGCCGCATCTTCAAAGGCGCACAGT-CGCCGGTACCGTAGTTGGTGAGGG-CT- 220
FA7UOGX01BZPJS              108+ AGCGCGCCGCATCTTCAAAGGCGCACAGT-CGCCG-TACCGTAGTTGGTGAGGG-CT- 161
FA7UOGX01BJS4A               90+ AGCGCGCCGCATCTTCAAAGGCGCACAGT-CGCCGGTACCGTAGTTGGTGAGGG-CT- 144
                                                            *


Other Reads:
                                                            *

FA7UOGX01CNMTP               34- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-T                      1
FA7UOGX01A6JO3               38- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCG                  1
FA7UOGX01BOS60               86- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 34
FA7UOGX01BT0JY  (2)         100- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 48
FA7UOGX01B2XG7              380+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 432
FA7UOGX01EGPAS  (2)         342+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 394
FA7UOGX01DNE49              329+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 381
FA7UOGX01AGHFW              326+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 378
FA7UOGX01DWFOM              323+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGGGCT- 376
FA7UOGX01D1QTG  (2)         174- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 122
FA7UOGX01EKSVI              316+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 368
FA7UOGX01BBCSW               36- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TAC                    1
FA7UOGX01AVRLX              202- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 150
FA7UOGX01C2S66              297+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 349
FA7UOGX01DG8NK              181- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 129
FA7UOGX01BLZ7D              200- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 148
FA7UOGX01ESD72              275+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 327
FA7UOGX01EV3XZ              268+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 320
FA7UOGX01B6NNA              261+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 313
FA7UOGX01EBPOL              251- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 199
FA7UOGX01APZ9G              252- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 200
FA7UOGX01BHDWX  (3)         231+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGT                 269
FA7UOGX01ANWUC              202- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 150
FA7UOGX01DCZ9L  (2)         233- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 181
FA7UOGX01ALF3Y              218+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 270
FA7UOGX01BXJ3U              205+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 257
FA7UOGX01BEPVP              314- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 262
FA7UOGX01AVTQB  (2)         322- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 270
FA7UOGX01CKDH7              178+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 230
FA7UOGX01DN1YY              171+ AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 223
FA7UOGX01C0VKV              156- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 104
FA7UOGX01A1H8Q  (2)         352- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 300
```

```
FA7UOGX01BRE21              255- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 203
FA7UOGX01E2966               81- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 29
FA7UOGX01C97W2              360- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 308
FA7UOGX01CXG46              366- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 314
FA7UOGX01B31FH              295- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 243
FA7UOGX01AWL03              334- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 282
FA7UOGX01BRAPE              392- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 340
FA7UOGX01C5CR9              241- AGCGCGCCGCATCTTCAAAGGCGCACA-TTCGCCG-TACCGTAGTTGGTGAGGG-CT- 188
FA7UOGX01DID3F  (2)         382- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 330
FA7UOGX01BKU30              421- AGCGCGCCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 369
FA7UOGX01ERM0C                1+       CCGCATCTTCAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGG-CT- 47
FA7UOGX01DED3G              427-                CAAAGGCGCACA-T-CGCCG-TACCGTAGTTGGTGAGGGGCT- 389
                                                            *


-----------------------------

>gi|49175990|ref|NC_000913.2|     66709   66709    -       T       31       16%

Reads with Difference:
gi|49175990|ref           66683+ TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 66734
                                                              *
FA7UOGX01CULNF              374- TCGCAGCGCGTAGGCCTGAT-AA-GGACGTCGCCAGCGTCGCATCAGGC-GTTGAA-T 321
FA7UOGX01EIYJR  (2)         116- TCGCAGCGCGTAGGCCTGAT-AA-G-ACGTCGCCAGCGTCGCATCAGGC-GTTGAA-T 64
FA7UOGX01C1QOA  (2)         201-           TAGGCCTGAT-AA-G-ACGTCGCCAGCGTCGCATCAGGC-GTTGAA-T 159
FA7UOGX01BYKCG              215-           TAGGCCTGAT-AA-G-ACGTCGCCAGCGTCGCATCAGGC-GTTGAA-T 173
FA7UOGX01CK0DY              246-           TAGGCCTGAT-AA-G-ACGTCGCCAGCGTCGCATCAGGC-GTTGAA-T 204
                                                              *


Other Reads:
                                                              *
FA7UOGX01B1PN1               50- TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGA    1
FA7UOGX01ESZ1V               38- TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGC                 1
FA7UOGX01CNLCM              404+ TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCAT               443
FA7UOGX01BJH4W              339+ TCGCAGCGCGTAGGCCTGATTAA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 391
FA7UOGX01CCASD              313+ TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 364
FA7UOGX01ETHT3              170+ TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 221
FA7UOGX01AI0ST              148+ TCGCAGCGCGTAGGCCTGATTAA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTG     197
FA7UOGX01BWFFM               84- TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 33
FA7UOGX01EDDYM              123- TCGCAGCGCGTAGGCCTGAT-AACG-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 71
FA7UOGX01B9JN9              109+ TCGCAGCGCGTAGGCCTGATTAA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 161
FA7UOGX01EE8OD              206- TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 155
FA7UOGX01EK5MJ               57+ TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 108
FA7UOGX01DUBNW               46+ TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 97
FA7UOGX01CGAFK  (2)         172- TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 121
FA7UOGX01DBAZO              175- TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 124
FA7UOGX01DWP87               33+ TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 84
FA7UOGX01C36VY  (2)          28+ TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 79
FA7UOGX01CE1EG              438- TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 387
FA7UOGX01DIBHC                6+ TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 57
FA7UOGX01EN06G              476- TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGCCGTTGAACT 423
FA7UOGX01D78X9  (2)           1+ TCGCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 52
FA7UOGX01CVQMK              163-   GCAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 114
FA7UOGX01AW34Z              108-    CAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 60
FA7UOGX01ERO82              223-    CAGCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 175
FA7UOGX01DQJ73  (2)         267-      GCGCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 221
FA7UOGX01C7OB7              356-        GCGTAGGCCTGAT-AA-G-ACG-CGCCAGCGTCGCATCAGGC-GTTGAA-T 312
                                                              *
```

[…]

13.4.9 454NewblerMetrics.txt File for the GS Reference Mapper Application

Example file listing for 454NewblerMetrics.txt produced by the GS Reference Mapper application. Note the different output compared with the file of the same name produced by the Assembly application (section 13.4.6).

```
/***************************************************************************
**
**      454 Life Sciences Corporation
**         Newbler Metrics Results
**
**      Date of Mapping: 2008/08/20 12:10:36
**      Project Directory: /home/adminrig/working/TiRuns/R_2008_05_29_07_49_18_
FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/D_2008_08_19_18_48_14_dragonforce_
fullProcessing/map_reg1_ecoli_v2
**      Software Release: 2.0.00.17
**
***************************************************************************/

/*
**  Input information.
*/

referenceSequenceData
{
       file
       {
              path = “/home/adminrig/genomes/ecoliGold.fas”;
              numberOfReads = 1;
              numberOfBases = 4639679;
       }

}

runData
{
       file
       {
              path = “/home/adminrig/working/TiRuns/R_2008_05_29_07_49_18_
FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/D_2008_08_19_18_48_14_dragonforce_
fullProcessing/sff/FA7UOGX01.sff”;

              numberOfReads = 688270, 688198;
              numberOfBases = 286727561, 273295575;
       }

}

/*
**  Operation metrics.
*/

runMetrics
{
       numberOfReferenceSequences  = 1;
       totalReferenceNumberOfBases = 4639679;

       totalNumberOfReads = 683576;
       totalNumberOfBases = 273104067;
       numberSearches   = 683576;
       seedHitsFound    = 24543594, 35.90;
       overlapsFound    = 1474047, 2.16, 6.01%;
       overlapsReported = 718883, 1.05, 2.93%;
       overlapsUsed     = 667911, 0.98, 92.91%;
}

readMappingResults
{
       file
       {
              path = "/home/adminrig/working/TiRuns/R_2008_05_29_07_49_18_
FLX12070284_lmcdade_200xRev10G70xVFRxEcoli/D_2008_08_19_18_48_14_dragonforce_fullPro-
cessing/sff/FA7UOGX01.sff";

              numMappedReads     = 678289, 98.56%;
              numMappedBases     = 271272773, 99.26%;
              inferredReadError  = 0.48%, 1280520;
       }

}

/*
** Consensus distribution information.
*/
consensusDistribution
{
       fullDistribution
       {
              signalBin =  0.0, 178037;
              signalBin =  0.6, 1;
              signalBin =  0.7, 129;
              signalBin =  0.8, 63764;
              signalBin =  0.9, 1604862;
              signalBin =  1.0, 824802;
              signalBin =  1.1, 3507;
              signalBin =  1.2, 81;
              signalBin =  1.3, 6;
              signalBin =  1.4, 3;
              signalBin =  1.5, 1;
              signalBin =  1.6, 38;
              signalBin =  1.7, 1660;
              signalBin =  1.8, 69948;
              signalBin =  1.9, 418661;
              signalBin =  2.0, 179052;
              signalBin =  2.1, 4856;
              signalBin =  2.2, 64;
              signalBin =  2.3, 6;
              signalBin =  2.4, 1;
              signalBin =  2.5, 6;
              signalBin =  2.6, 97;
              signalBin =  2.7, 2025;
              signalBin =  2.8, 25293;
              signalBin =  2.9, 85583;
              signalBin =  3.0, 45702;
              signalBin =  3.1, 3503;
              signalBin =  3.2, 88;
              signalBin =  3.3, 5;
              signalBin =  3.4, 4;
              signalBin =  3.5, 25;
              signalBin =  3.6, 288;
              signalBin =  3.7, 2172;
              signalBin =  3.8, 9702;
              signalBin =  3.9, 18539;
```

```
        signalBin =  4.0, 10311;
        signalBin =  4.1, 1539;
        signalBin =  4.2, 112;
        signalBin =  4.3, 10;
        signalBin =  4.4, 19;
        signalBin =  4.5, 63;
        signalBin =  4.6, 331;
        signalBin =  4.7, 1005;
        signalBin =  4.8, 2435;
        signalBin =  4.9, 4051;
        signalBin =  5.0, 3715;
        signalBin =  5.1, 1754;
        signalBin =  5.2, 371;
        signalBin =  5.3, 65;
        signalBin =  5.4, 14;
        signalBin =  5.5, 31;
        signalBin =  5.6, 108;
        signalBin =  5.7, 267;
        signalBin =  5.8, 603;
        signalBin =  5.9, 1036;
        signalBin =  6.0, 1045;
        signalBin =  6.1, 675;
        signalBin =  6.2, 241;
        signalBin =  6.3, 67;
        signalBin =  6.4, 15;
        signalBin =  6.5, 15;
        signalBin =  6.6, 37;
        signalBin =  6.7, 79;
        signalBin =  6.8, 149;
        signalBin =  6.9, 190;
        signalBin =  7.0, 214;
        signalBin =  7.1, 174;
        signalBin =  7.2, 93;
        signalBin =  7.3, 36;
        signalBin =  7.4, 10;
        signalBin =  7.5, 9;
        signalBin =  7.6, 12;
        signalBin =  7.7, 21;
        signalBin =  7.8, 33;
        signalBin =  7.9, 27;
        signalBin =  8.0, 41;
        signalBin =  8.1, 36;
        signalBin =  8.2, 18;
        signalBin =  8.3, 9;
        signalBin =  8.4, 1;
        signalBin =  8.5, 4;
        signalBin =  8.6, 1;
        signalBin =  8.7, 2;
        signalBin =  8.8, 2;
        signalBin =  8.9, 3;
        signalBin =  9.0, 5;
        signalBin =  9.1, 2;
        signalBin =  9.3, 3;
        signalBin =  9.4, 1;
        signalBin = 10.1, 1;
}

distributionPeaks
{
        signalPeak = 1, 0.98;
        signalPeak = 2, 1.96;
        signalPeak = 3, 2.96;
        signalPeak = 4, 3.94;
        signalPeak = 5, 4.96;
```

```
                signalPeak = 6, 5.98;
                signalPeak = 7, 7.00;
        }

        thresholdsUsed
        {
                threshold = 0, 1, 0.48;
                threshold = 1, 2, 1.46;
                threshold = 2, 3, 2.44;
                threshold = 3, 4, 3.40;
                threshold = 4, 5, 4.38;
                threshold = 5, 6, 5.42;
                threshold = 6, 7, 6.46;

                interpolationAmount = 1.00;
        }
}


/*
**  Consensus results.
*/
consensusResults
{
        readStatus
        {
                numMappedReads    = 678289, 98.56%;
                numMappedBases    = 271272773, 99.26%;
                inferredReadError = 0.48%, 1280520;

                numberFullyMapped     = 666241, 96.81%;
                numberPartiallyMapped = 1670, 0.24%;
                numberUnmapped        = 5287, 0.77%;
                numberRepeat          = 10378, 1.51%;
                numberTooShort        = 4622, 0.67%;
        }

        largeContigMetrics
        {
                numberOfContigs   = 47;
                numberOfBases     = 4611132;

                avgContigSize     = 98109;
                N50ContigSize     = 191651;
                largestContigSize = 434919;

                Q40PlusBases      = 4605929, 99.89%;
                Q39MinusBases     = 5203, 0.11%;

                numUndercalls     = 56;
                numOvercalls      = 44;
                numHCUndercalls   = 1;
                numHCOvercalls    = 0;
                consensusAccuracy  = 99.9978%;
                HCconsensusAccuracy  = 100.0000%;
        }

        allContigMetrics
        {
                numberOfContigs = 47;
                numberOfBases   = 4611132;
        }
}
```

13.4.10 454NewblerProgress.txt

Example file listing for a 454NewblerProgress.txt file generated by the GS Reference Mapper (truncated, as shown by bracketed ellipsis). (The file of the same name generated by the GS *De Novo* Assembler is slightly different.)

```
Mapping computation starting at: Wed Aug 20 12:10:30 2008  (v2.0.00.17)
Reading reference file ecoliGold.fas...
Indexing FA7UOGX01.sff...

  -> 688270 reads, 286727561 bases.
Setting up overlap detection...
Building a tree for 386639 seeds...
Mapping reads...

  -> 1000 of 683576
  -> 2000 of 683576
  -> 3000 of 683576
  -> 4000 of 683576 […]
  -> 681000 of 683576
  -> 682000 of 683576
  -> 683000 of 683576
  -> 683576 of 683576
Computing signals...

  -> 0 of 4639679...
  -> 10000 of 4639679...
  -> 20000 of 4639679...
  -> 30000 of 4639679...
  -> 40000 of 4639679... […]
  -> 4610000 of 4639679...
  -> 4620000 of 4639679...
  -> 4630000 of 4639679...
  -> 4639679 of 4639679
Generating output...

  -> Reading flowgrams...

  -> 0 of 5178053...
  -> 10000 of 5178053...
  -> 20000 of 5178053...
  -> 30000 of 5178053...
  -> 40000 of 5178053... […]
  -> 5150000 of 5178053...
  -> 5160000 of 5178053...
  -> 5170000 of 5178053...
  -> 5178053 of 5178053...
Mapping computation succeeded at: Wed Aug 20 12:20:21 2008
```

13.4.11 454AlignmentInfo.tsv

A small portion of an example '454AlignmentInfo.tsv' file is generated by the GS Reference Mapper reproduced below (truncated, as shown by bracketed ellipsis). (The file of the same name generated by the GS *De Novo* Assembler is slightly different.)

```
Position  Reference Consensus  Quality Score  Depth  Signal  StdDeviation
>gi|49175990|ref|NC_000913.2| 1
1         A         A          64             17     0.92    0.06
2         G         G          64             17     0.94    0.07
3         C         C          64             17     0.95    0.11
4         T         T          64             21     4.09    0.12
5         T         T          64             21     4.09    0.12
6         T         T          64             21     4.09    0.12
7         T         T          64             21     4.09    0.12
8         C         C          64             21     0.96    0.07
9         A         A          64             22     1.01    0.08
10        T         T          64             22     1.95    0.09
11        T         T          64             22     1.95    0.09
12        C         C          64             22     1.03    0.11
13        T         T          64             22     0.99    0.07
14        G         G          64             22     0.96    0.07
15        A         A          64             22     0.96    0.10 […]
```

13.4.12 454ReadStatus.txt

A small portion of an example '454ReadStatus.txt' file generated by the GS Reference Mapper is reproduced below (truncated, as shown by bracketed ellipsis):

```
Read            Mapping     Mapped         % of Read
Accno           Status      Accuracy(%)    Mapped
FA7UOGX01BOOGQ  Full        90
FA7UOGX01D1EU0  Full        97
FA7UOGX01DSWDG  TooShort
FA7UOGX01BZXFB  Full        100
FA7UOGX01BMD92  Full        100
FA7UOGX01DK9HV  Full        98
FA7UOGX01A9K87  Full        92
FA7UOGX01BMBBG  Full        96
FA7UOGX01CIMW2  Full        89
FA7UOGX01BA6G3  Full        96
FA7UOGX01C6BK1  Unmapped
FA7UOGX01CVQBG  Full        94
FA7UOGX01CSJGQ  TooShort
FA7UOGX01A0MV6  TooShort
FA7UOGX01CZ0E4  Full        97
FA7UOGX01A5ML6  Full        98
FA7UOGX01CYLW9  TooShort
FA7UOGX01CP9TE  Full        99
FA7UOGX01CF9AY  Partial     92             84
FA7UOGX01CBX29  TooShort
FA7UOGX01BQD2S  Full        100
FA7UOGX01DA158  TooShort
FA7UOGX01BSV9A  Partial     93             77
FA7UOGX01BR90M  Full        94
FA7UOGX01DR3NW  TooShort
FA7UOGX01BDI7S  Full        93
FA7UOGX01B5S6G  Full        98
```

```
FA7UOGX01BRXJW      Full         98
FA7UOGX01BBU6R      Unmapped
FA7UOGX01DLVHM      TooShort
FA7UOGX01CER97      Unmapped
FA7UOGX01DVLW5      Unmapped
FA7UOGX01B3Y3N      Full         96
FA7UOGX01BNGZA      Unmapped
FA7UOGX01BNJU8      Unmapped
FA7UOGX01DTPUE      Full         100
FA7UOGX01CPTUH      TooShort
FA7UOGX01CYCKT      Partial      96             85
FA7UOGX01CZ0LW      Full         100
FA7UOGX01CV5SS      Full         98
FA7UOGX01B3PB3      Full         98
FA7UOGX01BHZ9D      Full         93
FA7UOGX01CZXJJ      Unmapped
FA7UOGX01CDJEC      Unmapped
FA7UOGX01CP52V      Full         96
FA7UOGX01EBOFU      Full         99
FA7UOGX01CFM5N      Full         99
FA7UOGX01CH3LO      Full         100
FA7UOGX01DZKFM      Full         98
FA7UOGX01BHD76      Full         94              […]
```

13.4.13 454RefStatus.txt

Example file listing for a 454RefStatus.txt file. In this case, the Reference FASTA file contained only one sequence (to which any reads uniquely mapped).

```
Reference                     Num Unique     Pct of All      Pct of     Pct Coverage
Accession                     Matching Reads Unique Matches  All Reads  of Reference
gi|49175990|ref|NC_000913.2|  667911         100.0%          97.1%      99.39%
```

13.4.14 sffinfo summary

The format of the text summary output is the following, with a common text header showing the common SFF file header values, then one entry per read:

```
% sffinfo 454Reads.sff C3U5GWL01CBXT2
Common Header:
       Magic Number:        0x2E736666
       Version:             0001
       Index Offset:        119232960
       Index Length:        3242248
       # of Reads:          162112
       Header Length:       208
       Key Length:          4
       # of Flows:          168
       Flowgram Code:       1
       Flow Chars:
TACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGT
ACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACG
       Key Sequence: TCAG

>C3U5GWL01CBXT2
       Run Prefix:          R_2004_09_22_16_59_10_
       Region #:            1
       XY Location:         0838_3960
       Read Header Len:     32
       Name Length:  14
       # of Bases:          106
       Clip Qual Left:      5
       Clip Qual Right:     81
       Clip Adap Left:      0
       Clip Adap Right:     0

Flowgram:       0.99    0.06    1.09    0.08    0.10    0.96    0.10    1.00    1.08
0.14    0.04    1.08    0.12    2.87    0.10    0.98    0.09    1.08    0.08    0.08
0.95    0.12    0.97    0.06    0.11    1.05    0.11    0.04    2.05    0.15    1.01
0.06    0.16    1.01    1.04    0.11    1.00    0.12    0.11    1.99    0.19    0.93
1.03    0.13    2.05    0.17    0.11    1.01    0.15    2.89    0.14    0.99    1.06
0.17    1.97    0.16    0.13    2.02    0.14    0.99    0.10    1.97    1.03    0.12
0.08    1.99    0.15    0.09    0.96    1.10    0.12    2.00    0.17    0.99    2.04
0.12    0.11    2.12    0.13    0.09    2.07    0.15    0.09    1.02    0.10    0.10
0.99    0.14    0.11    1.03    0.16    0.06    1.02    0.17    1.02    0.07    0.17
1.06    0.08    0.05    1.04    1.08    3.10    0.11    1.01    0.15    1.97    0.11
0.12    1.98    0.13    0.99    1.03    0.12    1.03    0.07    2.94    0.15    0.10
2.85    0.25    0.10    0.87    1.18    0.17    0.96    1.06    0.13    1.02    0.17
0.12    0.94    0.14    0.89    0.12    1.05    0.11    0.99    1.01    0.14    0.08
1.00    1.05    0.94    0.09    0.16    1.04    0.13    0.08    1.94    1.10    0.07
0.08    1.08    0.10    3.96    0.17    0.93    0.10    0.18    1.01    1.01    0.11
1.86    0.13    0.16    0.98    0.20
Flow Indexes:   1       3       6       8       9       12      14      14      14
16      18      21      23      26      29      29      31      34      35      37
40      40      42      43      45      45      48      50      50      50      52
53      55      55      58      58      60      62      62      63      66      66
69      70      72      72      74      75      75      78      78      81      81
84      87      90      93      95      98      101     102     103     103     103
105     107     107     110     110     112     113     115     117     117     117
120     120     120     123     124     126     127     129     132     134     136
138     139     142     143     144     147     150     150     151     154     156
156     156     156     158     161     162     164     164     167

Bases:  tcagTGAAAGATCATTCACTGGACTTGAAAGTCCAAGAACAATAGGACCAATTGCATCATACCCTCCAAGTCTTTGG
GCGActgagacacgcaacaggggataggc
Quality Scores: 37      38      36      37      38      38      56      2       1
37      38      36      36      38      51      5       37      37      38      37
51      5       35      38      51      5       37      57      3       1       37
38      50      4       51      5       37      50      4       38      50      4
36      37      51      5       37      51      5       51      4       51      4
38      37      38      38      38      38      38      38      60      4       1
37      50      4       50      4       37      38      38      58      4       1
55      1       1       33      35      36      38      38      35      34      38
37      37      37      38      35      38      49      4       37      38      64
4       1       1       35      37      37      47      2       36
```

13.4.15 MID Configuration File

The off-instrument installation includes a file with the following name and location: mapasm_software_root_dir/config/MIDConfig.parse. This file contains the default MID set information, and can be edited or copied by users to create their own MID set configurations. The GS *De Novo* Assembler and the GS Reference Mapper applications, and the sfffile and sffvolume commands read this file by default, so any additional MID sets included in this file can be given on the command line. Also, the "-mcf" option for these applications can be used to specify a different configuration file, whose syntax must match the syntax described below.

```
/*
**
** MIDConfig.parse
**
** This file contains the multiplex sequences used by the Genome Sequence
** MID library kits, and may contain user-defined sets of multiplex
** identifiers.  This file is used by the post-run applications to access
** the defined MID sets.
**
** To use your own MID set, you can either copy this file to a local
** directory, add or edit your own sets (see below), then use the
** „-mcf“ option of the mapper and assembler to specify the MID
** configuration file.  Or, you can edit and save this file, to have
** your MID sets accessed by default by the post-run applications.
**
** To create a new MID set, copy the examples at the end of the file into
** the top section, then edit the text as follows:
**
**    * The name of the MID set should begin the group (appear above the
**      open brace ‚{‚)
**
**    * Each line in the MID set should contain three values after the
**      equals sign:
**         - A name for the specific MID sequence
**         - The DNA sequence of the MID
**         - The number of errors allowed in matching to the sequence
**
**    * The syntax of the line must be preserved (the „mid = „ beginning,
**      the semi-colon at the end of the line, the open and close braces
**      for the set.
**
**
** Note:  The names below use a combination of uppercase and lowercase
**        characters, but all matching to the names is insensitive to
**        case (so, for example „gsmids“ will match the MID set below).
**
*******************************************************************************

/*
** User-defined MID sets.
*/




/*
** IMPORTANT:  DO NOT EDIT BELOW THIS LINE.
**
** Genome Sequencer MID sets.
*/
```

```
GSMIDs
{
        mid = „MID1“, „ACGAGTGCGT“, 2;
        mid = „MID2“, „ACGCTCGACA“, 2;
        mid = „MID3“, „AGACGCACTC“, 2;
        mid = „MID4“, „AGCACTGTAG“, 2;
        mid = „MID5“, „ATCAGACACG“, 2;
        mid = „MID6“, „ATATCGCGAG“, 2;
        mid = „MID7“, „CGTGTCTCTA“, 2;
        mid = „MID8“, „CTCGCGTGTC“, 2;
        mid = „MID9“, „TAGTATCAGC“, 2;
        mid = „MID10“, „TCTCTATGCG“, 2;
        mid = „MID11“, „TGATACGTCT“, 2;
        mid = „MID12“, „TACTGAGCTA“, 2;
        mid = „MID13“, „CATAGTAGTG“, 2;
        mid = „MID14“, „CGAGAGATAC“, 2;
}
```

Published by

Roche Diagnostics GmbH
Roche Applied Science
68298 Mannheim
Germany

05346681001① 1008