YASMA - Yet Another Statistical Microarray Analysis
A quick tutorial (for version 0.19)

Lorenz Wernisch

May 14, 2002

Contents

1 Introduction

YASMA is an add-on library for the R statistical package and can be used to analyse simple replicated experiments. We are interested in bacterial genes over- or under-expressed in mutants as compared to the wild type. For this purpose, multiple mRNA samples from different cell cultures are hybridized on several arrays. As long as the same number of arrays is used for each sample a straightforward ANOVA analysis can be applied to the series of experiments (a balanced factorial design). From the standard error of the ANOVA anova analysis (or a bootstrap estimate of it) p-values for differential expression can be derived.

Even if you are not yet familiar with R this tutorial will give you a first idea of how R and YASMA works. Making full use of features of YASMA and R requires some knowledge of R. I recommend going through the R tutorial on the R online help page, especially the introductory chapters and the chapters on arrays, reading data, loops, writing functions, and on statistical models (YASMA supports model formulas).

2 Installing YASMA under Unix

To run YASMA you need R, a statistical package which can be downloaded for free from

http://cran.r-project.org/.

Download the YASMA package from

http://www.cryst.bbk.ac.uk/~wernisch/yasma/yasma_0.19.tgz

and for some additional functions (although not necessary for YASMA core functions) the SMA package from

http://www.stat.berkeley.edu/users/terry/zarray/Software/smacode.html.

Once R is installed, YASMA is added by

```
R INSTALL yasma_0.19.tgz
```

and similarly for the SMA package. For more information on how to install packages see the section on *R Add-On Packages* in the *Frequently Asked Qeustions* of the R online help (see section 3 for online help under R).

3 Running R, loading the packages and data

If R is installed properly, it is invoked by

```
R
```

R is a command line driven interpreter system with convenient editing facilities on the command line. For example, to load the YASMA (and SMA) package type

```
library(yasma)
library(sma)
```

on the command line. Alternatively (the way I prefer to work) cut and paste the above commands directly from the PDF file reader to the command line (possibly line by line entering each command by pressing return). Online help for R commands and all installed R packages is available with

```
help.start()
```

which usually launches an HTML page in your web browser.

The first thing to do is to load some data. The YASMA package contains microarray data on a *trcS* mutant of Mycobacterium tuberculosis provided by Sharon Kendall from Neil Stoker's lab. Load them with

```
data(trcs)
```

4 Reading and writing microarray data files

You load your own data from ASCII files (tab delimited) writing a more complicated command, which looks like

```
my.RG <- read.rg("/home/wernisch/microarrays/data/trcs/",
  gene.col="Gene ID",
  x.col="X Coord", y.col="Y Coord",
  R.filenames=c("70","70rs","71","71rs","72","72rs",
    "73","73rs","95","95rs","96","96rs" ),
  R.prefix="MT-cy3",R.suffix=".dat",
  G.filenames=c("70","70rs","71","71rs","72","72rs",
    "73","73rs","95","95rs","96","96rs" ),
  G.prefix="MT-cy5",G.suffix=".dat",
  R.col="Signal Median",Rb.col="Background Median",
  G.col="Signal Median",Gb.col="Background Median",
  do.prepare=T,start.phrase="Gene ID",end.phrase="End Raw Data",
  experiments=list(S=1:3,A=1:2)
)
```

Note that this only works with a *complete path name* and a data set in this directory. The details of the command are explained in the description of the command "read.rg" in the help tool: click "Packages", "yasma", "read.rg"), alternatively you can view the help within R by `help(read.rg)`.

Essentially, the procedure reads in all relevant data files, strips them of everything before a line containing the start phrase `Gene ID` and of everything following a line containing the end phrase

`End Raw Data` and extracts signal and background expression levels for the "red" and "green" channel from columns named `Signal Median` and `Background Median`, and gene names from column `Gene ID`.

An important parameter of function `read.rg` is `experiments`. It describes the layout of the experiments for later use in the ANOVA analysis. In the above example the data are obtained from 3 mRNA cultures (S=1:3), each on 2 microarrays (A=1:2), and for each array we have two spot quantifications (Q=1:2). This resulted in $12 = 3 \times 2 \times 2$ array data sets: sets 1–4 from culture(sample) 1, sets 5–8 from culture 2, sets 9–12 from culture 3. Sets 1,2 are two spot quantifications of the first array of sample 1, sets 3,4 are two spot quantifications of the second array of sample 1, and so on. This is also the layout of the data set `trcs` which you loaded above.

The next step is to get rid of spots containing other material than genes.

```
trcs.RG <- rg.remove.containing(trcs.RG,c("Cy3","Cy5","Carry-over",
    "Spot","50%","GAPDH","B-actin","23s","16s","5s","10sa","PCR"))
```

Alternatively, use the command `rg.keep.containing` if, for example, all genes containing "Rv" should be kept. We are left with 3924 genes as is seen by typing

```
length(trcs.RG$genes)
```

Write the `RG` structure to a series of files (eg "RGcopy") by

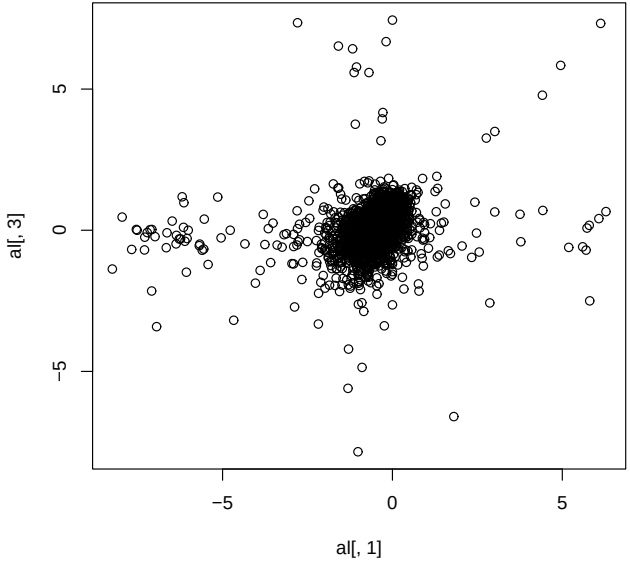
```
rg.write(trcs.RG,"RGcopy")
```

This generates a series of files with names starting with "RCopy-" in the current working directory. This command is useful if one wants to manipulate the `RG` structure and use the result in other analysis packages.

5 Data quality

Let us have a first plot of the data. Plot $\log(R/G)$ from experiment 1 over $\log(R/G)$ from experiment 3 (same sample, different arrays) by

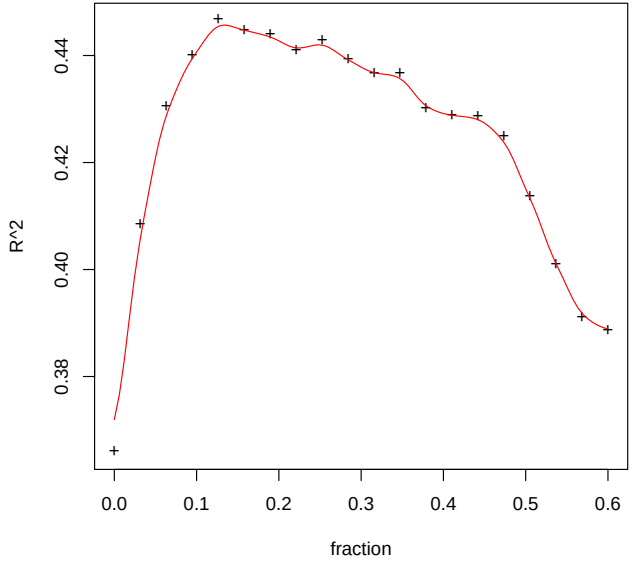
```
al <- rg.log.matrix(trcs.RG)
plot(al[,1],al[,3])
```



Ideally all points should lie on a line. But there is only a weak hint of correlation in this plot.

The question arises how many points should be removed to get a good correlation between the experiments. We don't want to remove too many and lose genes. An indication is given by a plot of the overall correlation R^2 of all experiments over the percentage (quantile) of genes removed from each array.

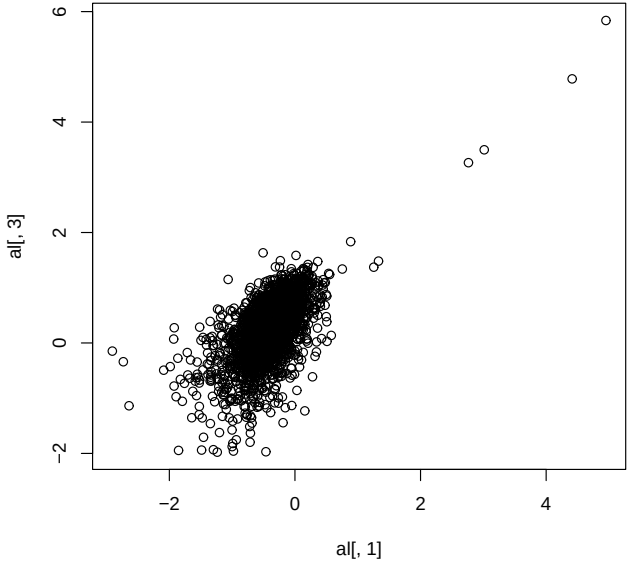
```
rg.rsq.plot(trcs.RG)
```



The plot suggests that removal of about 10% of the points would lead to a good correlation among the experiments, so let's do that.

We remove genes with their expression levels in the lower 10% (also called the lower 0.1 quantile) in any one array and draw the plot again

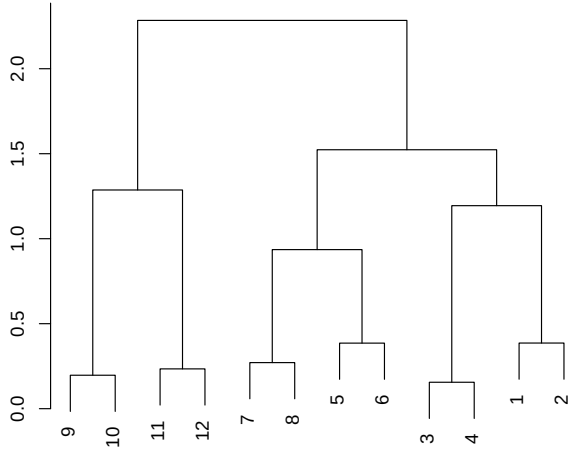
```
RG <- rg.remove.quantile(trcs.RG,0.1)
al <- rg.log.matrix(RG)
plot(al[,1],al[,3])
```



This looks much better!

Another interesting question is how well the experiments correlate with each other. A clustering tree representing rank correlations of $\log_2(R/G)$ values on the various arrays is

```
rg.cor(RG)
```



The correlation table is shown below. The correlation is not very good actually. The two quantifications correlate well (1 and 2), also correlation within samples is reasonable (1 and 3). But between samples correlation is very low. This is due to the difficulties of working with the slowly growing organism that is difficult to handle.

	1	2	3	4	5	6	7	8	9	10	11	12
1	1											
2	0.78	1										
3	0.56	0.47	1									
4	0.56	0.47	0.92	1								
5	0.36	0.3	0.55	0.53	1							
6	0.37	0.33	0.56	0.54	0.78	1						
7	0.43	0.43	0.48	0.47	0.61	0.63	1					
8	0.41	0.43	0.46	0.46	0.58	0.61	0.85	1				
9	0.082	0.016	0.097	0.089	0.29	0.26	0.25	0.24	1			
10	0.16	0.13	0.14	0.14	0.33	0.29	0.33	0.33	0.9	1		
11	0.14	0.12	0.11	0.099	0.31	0.3	0.31	0.3	0.48	0.49	1	
12	0.16	0.13	0.12	0.11	0.33	0.31	0.34	0.33	0.5	0.52	0.88	1

Rank correlation only measures the agreement in the order of values is. The usual Pearson correlation can be obtained by

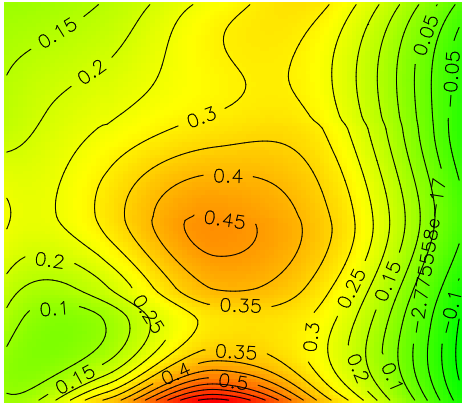
```
rg.cor(RG,method="pearson")
```

The Pearson correlation is slightly higher than rank correlation, probably due to outliers along the diagonal.

6 Normalization

Dyes hybridize differently on microarrays. The effect is seen when plotting a two dimensional smoothing (loess) surface fitting $\log_2(R/G)$ values of spots over their coordinates, for example, for array 9

```
tmp <- rg.loess.array.normalize(rg.project(RG,9))
```



Such surfaces are subtracted in array normalization

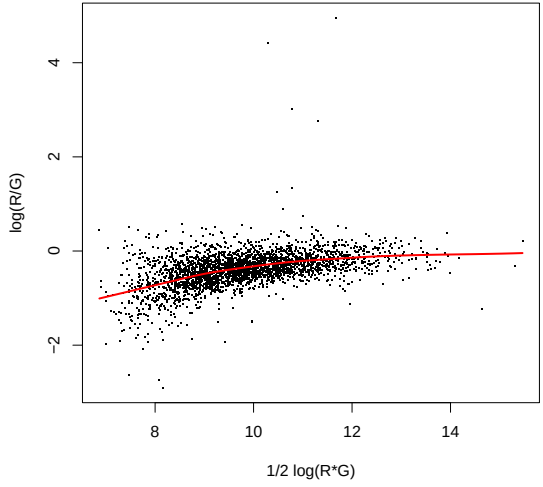
```
RG.norm <- rg.loess.array.normalize(RG)
```

This improves overall R^2 from 0.436 to 0.454:

```
rg.rsq(RG); rg.rsq(RG.norm)
```

A different style of normalization fits a smoothing curve to the log difference $\log_2(R/G)$ over total intensity $0.5 \log_2(R+G)$.

```
tmp <- rg.loess.normalize(rg.project(RG,1))
```



Again intensity dependend normalization can be applied to all arrays:

```
RG.i.norm <- rg.loess.normalize(RG)
```

An alternative normalization is linear regression normalization `rg.linear.normalize(RG)`. For the following analysis we just take the simplests normalization and subtract the mean log ratio value from each array:

```
RG <- rg.scale.normalize(RG)
```

7 ANOVA analysis

An ANOVA analysis gives an overview of the sources of variation in a data set. We have already seen that the differences between samples are large. This can be more formally assessed by the following steps. First a special array of $\log(R/G)$ values suitable for balanced factorial ANOVA is produced

```
a <- rg2log.array(RG)
```

Then an ANOVA routine is applied to this array. We want to inspect all the variability due to the difference in R and G, which we call varieties V, the genes G, the samples S, and the arrays A. These factors are also called "main effects". The YASMA commands are

```
av <- bfaov(~(V+G+S/A)^2,a)
anova(av)
```

It is convenient to work with *model formulas* (see the R tutorial for more details) when specifying which effects should be analyzed by ANOVA. The letters used in the formula are the ones provided in parameter `experiments=list(S=1:3,A=1:2)` when reading input with `read.rg` plus the special factors `V` (which is the R and G) and `G` (the genes) which always exist.

The model formula `~(V+G+S/A)^2` indicates that we want to have information about all the variability due to the main effects and their pairwise interactions (hence the square `^2`). A further complication is that the array effect A is "nested" within the samples S, which means that array 1 from sample 1 and array 1 from sample 2, for example, have the same number for convenience only and not because there is any close relationship between them. This is indicated by `S/A`.

The following ANOVA table is generated.

effects	df	SS	MS
V	1	$< 1.0e-15$	$< 1.0e-15$
G	2750	58374	21.227
S	2	6133.5	3066.8
VG	2750	2447.2	0.88988
VS	2	$6.9445e-26$	$3.4723e-26$
GS	5500	9424.1	1.7135
A,SA	3	40.877	13.626
VA,VSA	3	$6.9445e-26$	$2.3148e-26$
GA,GSA	8250	5633.3	0.68283
residuals	46762	4486.5	0.095944

Two columns are important here, the "effects" and the "MS" column. Effects are the factors discussed above. The variability that stems from a particular effect can be read off the MS (mean squared) column. The higher this number, the stronger the contribution to overall variability. A must be combined with SA due to the nesting mentioned above). At the bottom line "residuals" is an account of all the variability that is not listed in the main table. The residual MS is the amount of variability due to other factors which we consider as random. Since we normalized all arrays to zero mean of log ratios, many effects and interactions involving variety are practically zero.

One disadvantage of this ANOVA analysis is that all all effects are assumed to be fixed. That is, the above numbers only tell us something about random fluctuation we would get if we took the same arrays and cultures again. Since we are more interested in the variation of future experiments, an analysis of variance components is more appropriate.

8 Analysis of variance components

As mentioned above, samples S and arrays A are random representatives from a large pool of potential future experiments. The corresponding effects are *random effects*. An analysis of variance components takes that into account. It builds on a standard ANOVA analysis but derives variances differently (for more details see the manuscript
http://www.cryst.bbk.ac.uk/~wernisch/yasma/replicates.pdf)

First, we construct a matrix of log ratio values

```
a <- rg2log.array(RG,log.ratio=T)
```

then we apply ANOVA analysis and derive variance components from it:

```
av <- bfaov(~G + S/A + G:S + G:S:A,a)
rml <- reml.bfaov(av,c("S","A"),~G + S/A + G:S + G:S:A)
print.reml(rml)
```

The resulting variance components look something like

```
S         -2.3834e-05
G,S          0.065568
A,S       -4.0552e-05
G,A,S         0.08771
res          0.047697
```

The variance components σ_S (S) and σ_A (A,S) are small due to normalization and the large number of data points contributing to them (degrees of freedom). Of most interest are the variance compons σ^2_{GS} (G,S), σ^2_{GSA} (G,A,S), and σ^2 (res), which measure log ratio variability with cultures, arrays, and quantifications.

9 Optimal designs

An equation relating variance of log ratio estimates from replicates to variance components is:

$$\mathrm{Var}(\widehat{G}_g) = \frac{1}{n_S}\left(\sigma_S^2 + \sigma_{GS}^2\right) + \frac{1}{n_S n_A}\left(\sigma_{SA}^2 + \sigma_{GSA}^2\right) \quad + \frac{1}{n_S n_A n_Q}\sigma^2 \tag{1}$$

Here $\widehat{G}_g$ is the average log ratio of gene g, σ_S^2, σ_{GS}^2, σ_{SA}^2, σ_{GSA}^2, and σ^2 are the variance components of the corresponding effects and n_S, n_A, and n_Q are the number of samples, of arrays per sample, and of quantifications per array (here they are 3, 2, and 2, respectively).

The routine `optim.design` uses this equation to calculate an optimal design when costs for cultures (say, 50 units), arrays (10 units), and quantifications (1 unit) are given, and total costs are limited (say by 200 units):

```
optim.design(av,c("S","A","Q"),c(50,10,1),limit=200)
```

The result shows that 2 cultures, 4 arrays per culture, and 2 quantifications per array are the optimal combination of replications on each level, with total costs of 196. The variance of log ratio estimates per gene is then about 0.047.

From the variance of the log ratio the fold-change which counts as significant is easily derived:

```
ng <- length(RG$genes)
2^qnorm(0.01/ng,0,sqrt(0.047),lower=F)
```

which is about 1.96, that is, a two-fold overexpression would be significant at a 0.01 level, assuming a simple Bonferroni correction for the number `ng` of genes tested.

10 Significance of differential expression

Significance analysis is done by bootstrapping the residuals of the ANOVA model and generate artificial data. From them the variability in the estimations for the effect of variety (R or G) on gene expression can be derived more reliably. To start a bootstrapping process write

```
formulas <- list(~G + S + A, ~G + S + A + G:S,~G + S + A + G:S + G:S:A)
levels.seq <- c("S","A")
```

which describes the levels at which the hierarchical bootstrap needs to be performed.

```
ou <- over.under.level(av,rml,rounds=500,
            formulas=formulas,levels.seq=levels.seq,
            RG=RG,span=0.4,stats.only=F)
ldf <- data.over.under(ou,pvalue=0.01,method="bonferroni")
```

This will take a while since first three loess curves have to be fitted to the residuals (the plots are shown during the process) and then 500 bootstrap samples are generated. This is reasonably fast only if the bootstrap routine, which is written in C, has compiled successfully. Finally we get the list of differentially expressed genes by

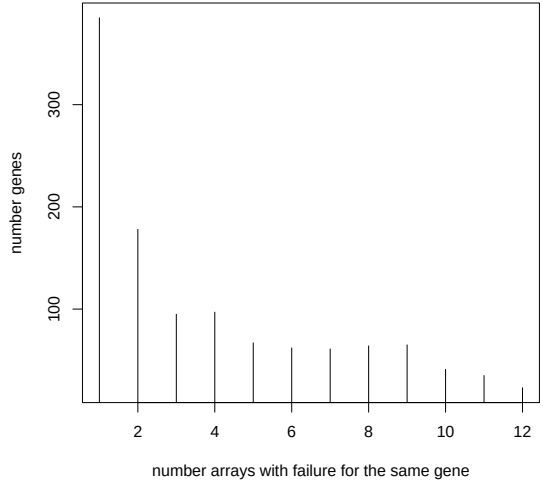
```
ldf$over.b
```

They are saved to extra files automatically when using `write.over.under`. For more details see again the help files and the manuscript http://www.cryst.bbk.ac.uk/~wernisch/yasma/replicates.pdf

11 Missing data

For many points in our data set the background intensity is actually larger than the signal itself. Earlier we removed genes which had a value in the lowest 10% quantile in a single channel of a single array. This is necessary since we cannot afford to have missing values in a balanced factorial design. But instead of removing the complete gene we can interpolate the missing values using the ANOVA design itself. Essentially, selfconsistent averages according to the ANOVA model formula replace the missing values.

```
rg.failure.histogram(trcs.RG,0.1)
```



The diagram shows that most genes have low values in 1 or 2 arrays/channels.

Consequently, we want to keep those genes.

```
RG <- rg.remove.quantile(trcs.RG,0.1,level=3,set.na=T)
```

removes genes that have low values on at least 3 channels/arrays and keeps genes with weak values on one or two channels/arrays (these values are set to `NA`).

Specify an ANOVA model as basis for interpolation

```
a.ip <- rg2log.array(RG)
av.ip <- bfaov(~(S/A + V)^2 + G + V:G,a.ip)
a <- interpolate.bfaov(av.ip,a.ip)
RG <- log.array2rg(a)
```

Array `a` now contains the interpolated data and `RG` the interpolated `R` and `G` values. Continue now with the ANOVA on the array `a` as usual.

An alternative solution that completely avoids negative intensity values is *not to subtract background* from the signal at all:

```
RG <- rg.zero.bg(trcs.RG)
```

There is no need in this case to remove data points. The correlation between replicated experiments improves dramatically. Now repeat the ANOVA analysis on the new `RG` data.