Analysis of Spotted Microarray Data

John Maindonald

Statistics Research Associates
http://www.statsresearch.co.nz/

Revised August 14 2016

The example data will be for spotted (two-channel) microarrays. Exactly the same approaches are relevant to spotted oligonucleotides.

1 Spotted Microarray Methodology & Tools

Each array ("slide") typically compares expression in genes from one "sample" (type of cell) with expression in genes from another "sample". The common dyes are Cy5 ("red") and Cy3 ("green").

Steps in the process of getting microarray intensity measurements are:

1. cDNA samples are obtained by reverse transcription, one for each of the two mRNA samples. One sample is labeled with "red" dye and one with "green" dye.
2. The samples are mixed and hybridized onto the slide.
3. An image analyser is used to give separate images of the "red" and "green" signals. Think of these as much like digital camera images, but with just two frequencies. The resolution may be 15-30 pixels per spot.
4. Image analysis software is used to extract various statistical summaries of the images of the spots and of their surroundings. This involves:
   (a) Determination of spot boundaries.
   (b) Determination, for each spot, of "red" (R) and "green" (G) signals. Most software offers a choice of alternative summaries of the spot pixel intensities. Typically an average (mean or median) is taken.
   (c) Determination, for each spot, of "red" (Rb) and "green" (Gb) backgrounds.
   (d) Determination of one or more quality measures. These may include: spot size, spot shape (round spots are best, if this is what the printer was supposed to provide), spot intensity and the range of intensity values.

The spots that are of interest are those where is a consistent difference between the red and the green signals. It is usual to take either log(R)-log(G) or log(R-Rb)-log(G-Gb), apply a correction for dye bias, and use this as a measure of differential expression. This leads to a logratio (M) that can be used for further analysis.

Visual checks of the spatial distribution of the separate statistical summaries can be highly revealing. Below, we will obtain spatial plots for each of R (red), Rb (red background), G (green), Gb (green background), M (logratio) and one or more spot quality measures.

1.1 Scanning and Image Analysis

The images with which we will work are from an Affymetrix TM 428 scanner. Image analysis was performed using the CSIRO Spot image analysis software. There are six "**.spot**" files, one for each of six replications of the experiment. Information about the three slides (each with dyeswap

repeats) is in the file **coralTargets.txt**. The file **dk_coral-annotated.gal** holds identication information about the genes on the slides, and about the layout on the slides. [1]

Here are the contents of the file **coralTargets.txt**

```
SlideNumber  FileName      Cy3  Cy5
        221a coral551.spot post  pre
        221b coral552.spot  pre post
        223a coral553.spot post  pre
        223b coral554.spot  pre post
        224a coral555.spot post  pre
        224b coral556.spot  pre post
```

Note that:

`pre` = pre-settlement, i.e., before the larvae have settled

`post` = post-settlement, i.e., after settlement onto the substrate.

1.2 Plotting and Analysis Software

The *limma* package, written by Dr Gordon Smyth and colleagues at WEHI, will be used here. It is reasonably straightforward to use, does a good job of initial data exploration, and offers state of the art abilities for analysis of differential expression.

Additionally, three functions from the *DAAGbio* package will be used.

2 Getting Started

The data files that are used here will be accessed from the **doc** subdirectory in the *DAAGbio* package installation.

```
library(DAAGbio)

## Loading required package:  limma

path2data <- system.file("doc", package="DAAGbio")
```

Note 1: For a new project, a good first step is to start a new directory that holds the data. For the purposes of this vignette, the following files would be placed in that directory: **coral551.spot**, **coral552.spot**, **coral553.spot**, **coral554.spot**, **coral555.spot**, **coral556.spot**, files **coralTargets.txt**, **SpotTypes.txt** and **dk_coral-annotated.gal**. Then set `path2data` to be the path to that directory.

Note 2: As of version 0.6 of *DAAGbio*, the six "**.spot**" files **coral551.spot**, . . . , **coral556.spot** are stored in a compressed format, reducing them to a little over 40% of their original size. They have been created by typing, on a Unix or Unix-like command line:

gzip -9 coral55?.spot

They were then renamed back to ****.spot***. The R file-reading abilities that are used by `read.maimages()` are, from R version 2.10.0 and onwards, able to process such files.

[1] **Note:** Users of Axon scanners are likely to work with the accompanying GenePix image analysis software. This gives files that have the suffix "**.gpr**". Other combinations of scanner and image analysis software will yield files that hold broadly equivalent information, but formatted and labeled differently. ANU users may encounter the combination of Affymetrix TM 428 scanner and Jaguar image analysis software that is available at the RSBS Proteomics and Microarray facility. Jaguar places records in the output file in an order that is different from that used by Spot and GenePix.

Attaching the *limma* package

Start an R session. From the command line, type

```
library(limma)
```

Help for `limma` Type `help.start()`. After a short time a browser window should open. You might want to take a brief look at the User's Guide for *limma*. Click on Packages, then on limma, then on Overview, then on LIMMA User's Guide (pdf)

2.1 Reading Data into R

As a first step, read into R the information about the half-slides, from the file **coralTargets.txt**. Here is how:[2]

```
targets <- readTargets("coralTargets.txt", path=path2data)
targets$FileName      # Display the file names
```

The first command gets the targets information. The names of the files are stored in the `FileName` column.[3] To see the complete information that has been input: `targets`, type:

```
targets
```

Next, the function `read.maimages()` (from the *limma* package) will be used to read in the data. This function puts all the information from the separate files into a single data structure, here called `coralRG`. Use of a name that has the final `RG` is not compulsory, but serves as a useful reminder that the structure has information on the red signal and red background (`R` and `Rb`), and on the green signal and green background (`G` and `Gb`).

```
coralRG <- read.maimages(targets$FileName, source = "spot",
    path=path2data,
    other.columns=list(area="area", badspot="badspot"))
```

The `other.columns` information is an optional extra, which it is however useful to use to read in quality measures.

Various summary information can be obtained for the different half-slides and measures. Foe example:

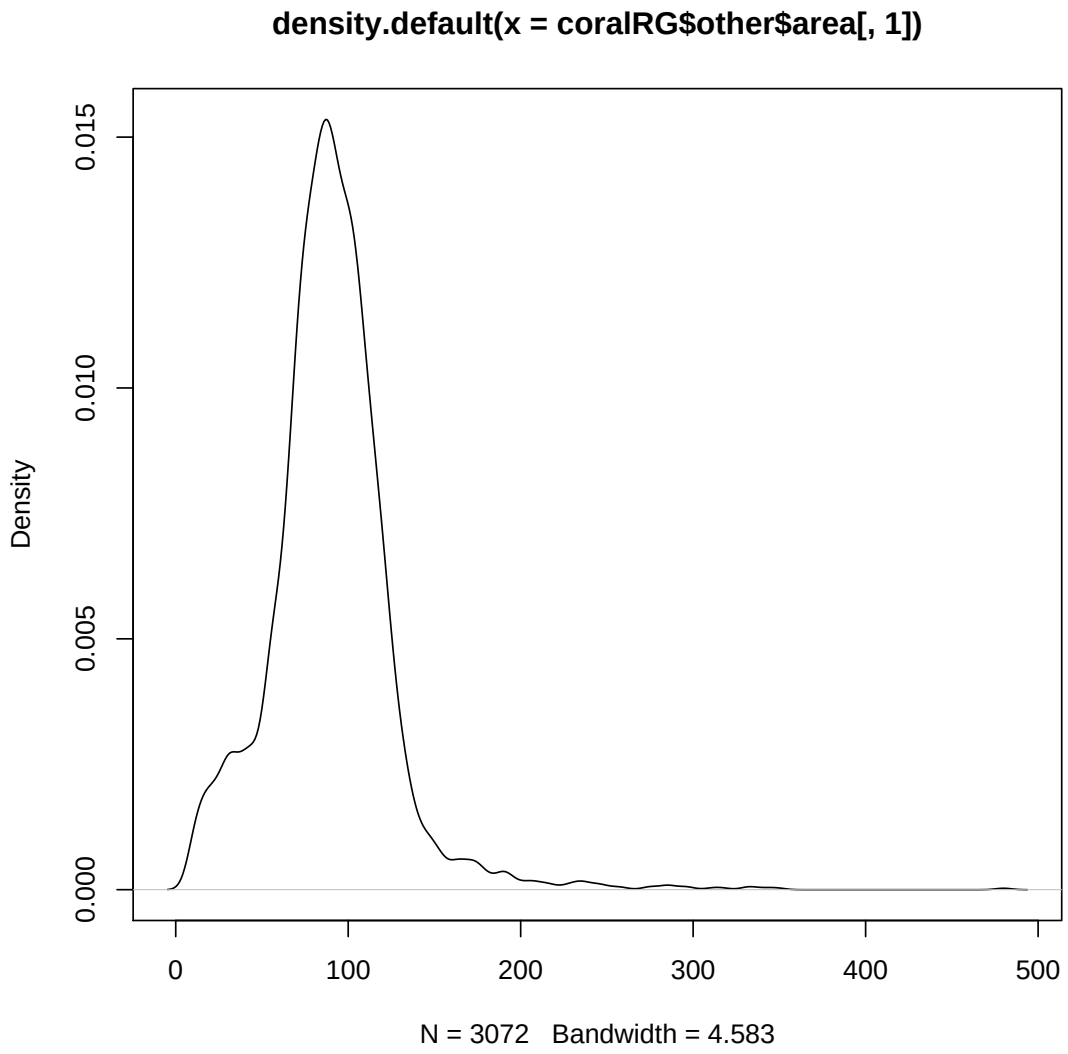
```
summary(coralRG$other$area)
```

However these may not mean much for those who are new to microarray data. Graphs are in general more helpful. For example, here is a graph that will help in judging the extent to which spots are close to the expected 100 units of area.

```
plot(density(coralRG$other$area[,1]))
```

[2] **Note:** The above assumes that tabs have been used as separators, when the file **coralTargets.txt** was created. I have created a second targets file – **coralTargsp.txt**, in which the separators are spaces. For this, enter:
`targets <- readTargets("coralTargsp.txt", path=path2data, sep="")`

[3] `## Another way to get the file names is to enter`
`fnames <- dir(path=path2data, pattern="\\.spot")`
`# Finds all files whose names end in ".spot''`



The *limma* package does actually allow for the use of area as a measure of quality, with information from spots that are larger or smaller than the optimum given a reduced weight in the analysis.

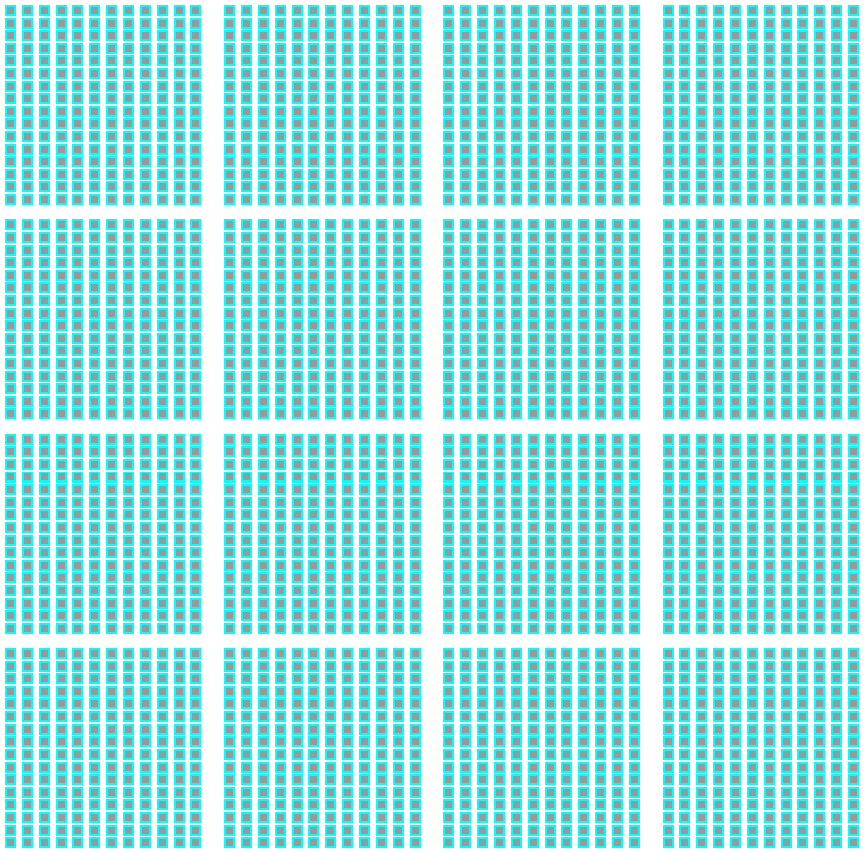
Sequence annotation and related information

Next, additional information will be tagged on to the `coralRG` structure, first gene annotation information that is read from the **.gal** file, and then information on the way that the half-slides were printed that can be deduced from the gene annotation file.

```
coralRG$genes <- readGAL(path=path2data)
coralRG$printer <- getLayout(coralRG$genes)
coralRG$printer
```

These half slides were printed in a 4 by 4 layout, corresponding to the 4 by 4 layout of the printhead. Each tip printed an array of 16 rows by 12 columns. To see the order in which the spots were printed, attach the *DAAGbio* package, and run the function `plotprintseq()`, thus: and enter:

```
plotprintseq()
```



The Spot Types File

This is optional, but strongly recommended. Spots can be grouped into at least four categories – there are "genes" (really, partial gene sequences), negative controls, blanks and differentially expressed controls. Genes may or may not be differentially expressed, the next three categories should not show evidence of differential expression, and the differentially expressed controls should mostly show evidence of differential expression. Plots that label points according to their types can be beautiful, insightful and, if they find nothing amiss, reassuring.

Here is what the file looks like (note that tabs must be used as separators, for the code below to work without modification):

```
SpotType        ID      Name    Color
gene            *       *       black
negative        *       [1-9]*  brown
blank           *       -       yellow
diff-exp ctl    *       DH*     blue
```

Any color in the long list given by the function `colors()` is acceptable. Try for example `royalblue` or `hotpink` for calibration spots. They are much less effective. You could use `coral` in place of `black` for the "gene"!!

The following extracts the spot type information from the file, and appends it to the data structure that I have called `coralRG`:

```
spottypes<-readSpotTypes(path=path2data)
coralRG$genes$Status <- controlStatus(spottypes, coralRG)

## Matching patterns for: ID Name
## Found 3072 gene
## Found 7 negative
## Found 30 blank
## Found 21 diff-exp ctl
## Setting attributes: values Color
```

3 Plots

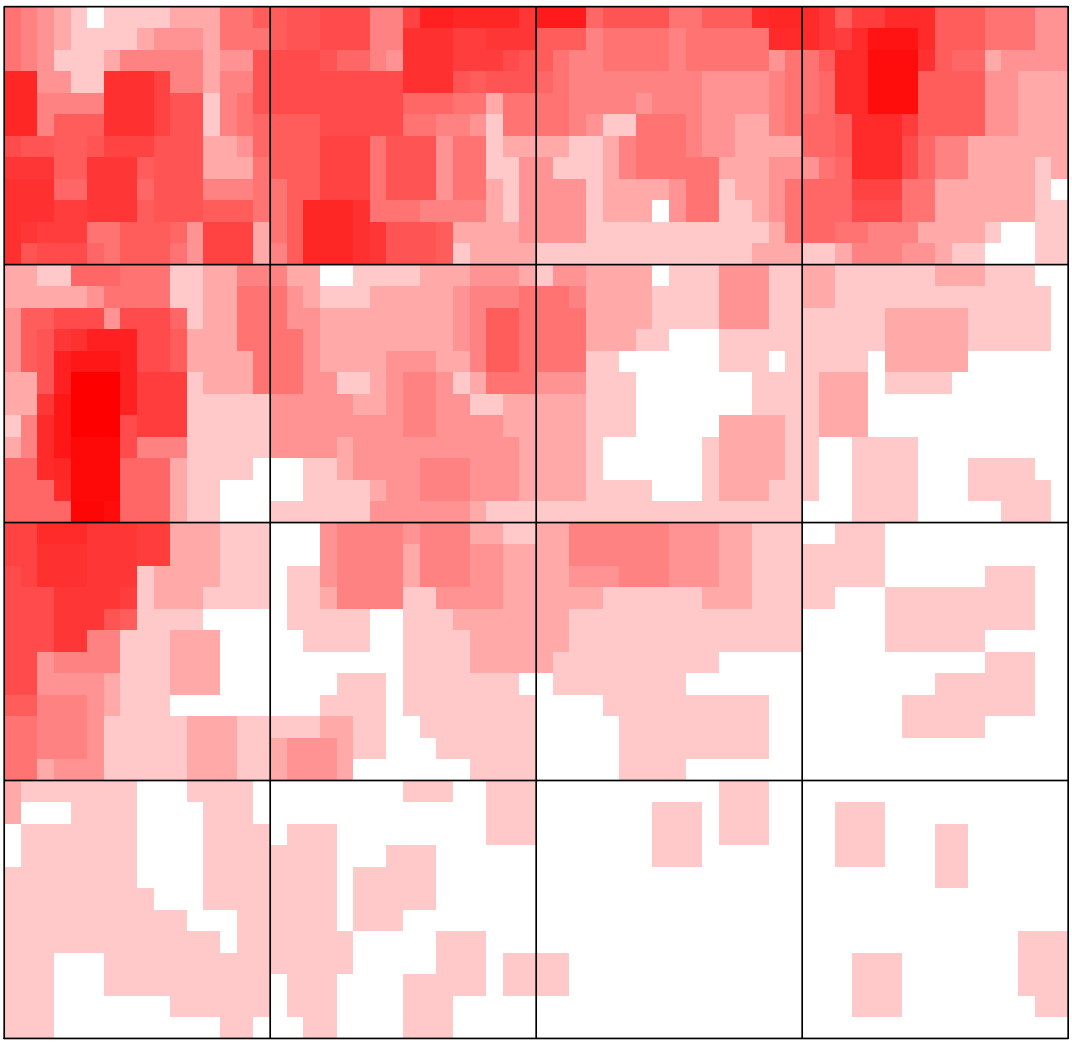
3.1 Spatial Plots

These plots use colour scales to summarize, on a layout that reflects the actual layout of the slide, information on the spots. The idea is to check for any strong spatial patterns that might indicate some lack of uniformity that has resulted from the printing (e.g., one print tip printing differently from the others), or from uneven conditions in the hybridization chamber. There may be surface features, perhaps due to a hair or to the mishandling of one corner of the slide, this may show up on the plot.

What is finally important is of course the effect on the log-ratios (the M-values). Some unevenness in the separate foreground and background intensities can be tolerated, providing that it leads to much the same proportional change in both channels.

There are two possibilities – to use `imageplot()` from *limma*, or to use my function `imgplot()` that is included in the *DAAGbio* package; see the appendux. Here are plots that use `imageplot()`:

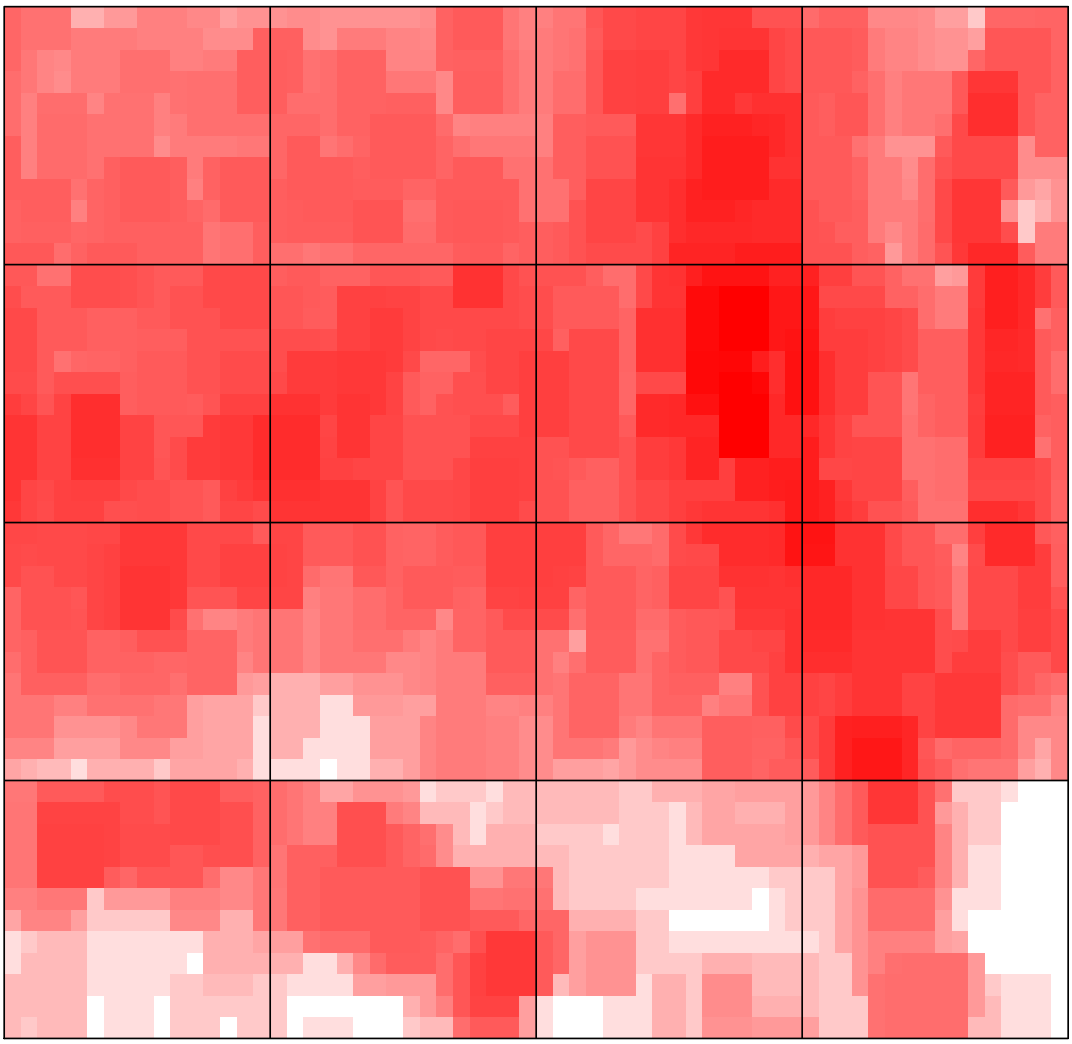
```
imageplot(log2(coralRG$Rb[, 1]+1), layout = coralRG$printer,
          low="white", high="red")
```



z−range 0 to 4.7 (saturation 0, 4.7)

This should be repeated for each different half-slide, for both red and green, and similarly for the green background. To get the plot for the second half-slide, type:

```
imageplot(log2(coralRG$Rb[, 2]+1), layout = coralRG$printer,
          low="white", high="red")
```



z−range 0 to 7.4 (saturation 0, 7.4)

The plots can be obtained all six at once, in a three rows by two columns layout. For this, use the function `xplot()`, included in the *DAAGbio* package.

As an example, to get the six plots for the red channel on the screen, type:

```
x11(width=7.5, height=11)
xplot(data = coralRG$R, layout = coralRG$printer, FUN=imageplot)
```

(Under Macintosh OSX with the Aqua GUI, specify `quartz(width=7.5, height=11)` to use the quartz device. It is also possible to send the output to a hard copy. Type, e.g.:

```
quartz(width=7.5, height=11)
xplot(data = coralRG$R, layout = coralRG$printer, FUN=imageplot,
      device=pdf)
```

Other possibilities for `device` are `device="ps"`, `device="png"`, `device="jpeg"` and `device="bmp"`. For `device="png"` and `device="jpeg"` the parameters `width` and `height` will need to be specified, in pixels, in the call to `sixplot()`.

3.2 MA plots & Normalization

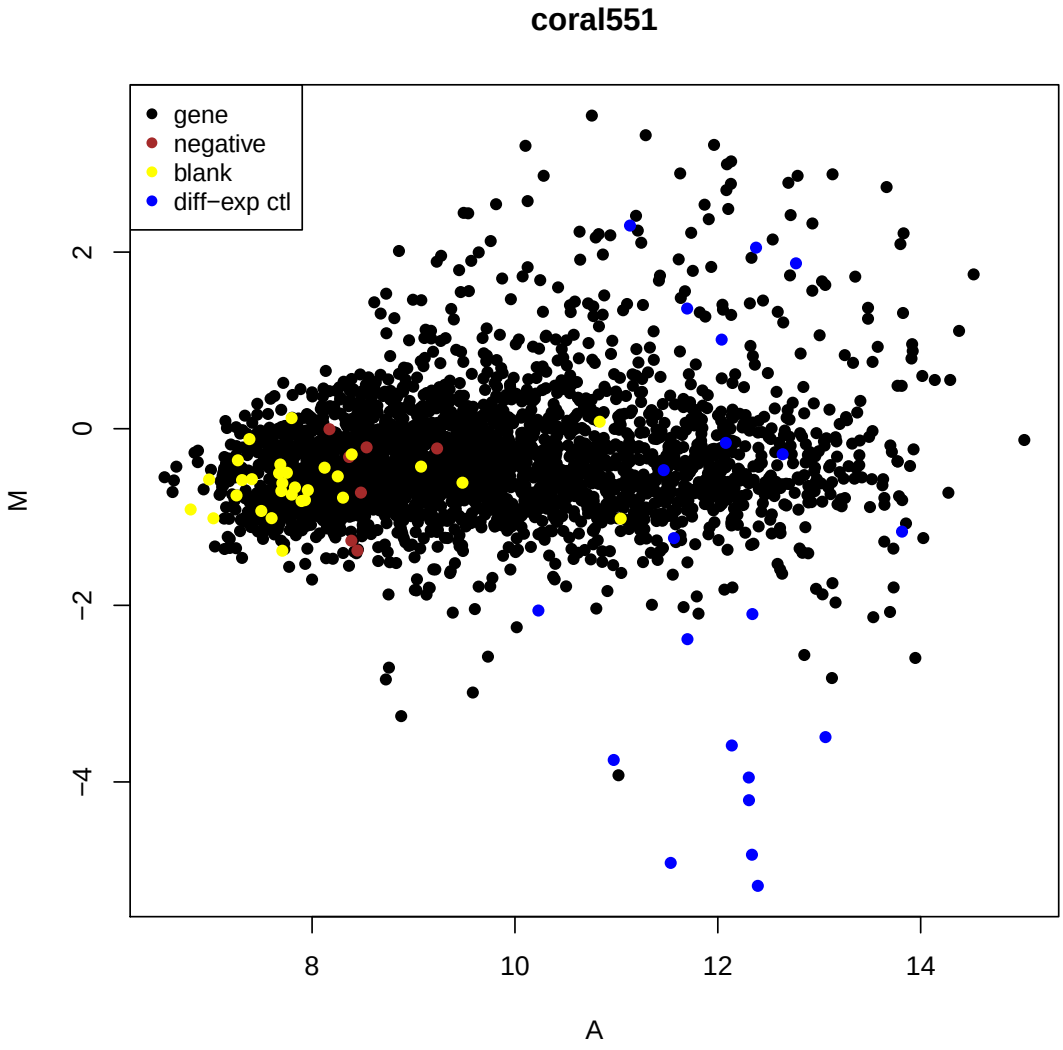
In this overview, the default background correction will be used, in which the background is in each case subtracted from the corresponding measured signal.[4]

At this point normalization is an issue. The Cy3 channel ("green") typically shows up with higher intensity than the Cy5 channel ("red"). The measure of differential expression is

$$\mathrm{M} = \log(\text{redintensity}) - \log(\text{greenintensity}) = \log(\frac{\text{redintensity}}{\text{greenintensity}})$$

First, check what happens if we do not normalize. Try the following:

```
plotMA(coralRG, array=1)
```



This plots the unnormalized log-ratios (the M-values) against the the averages of the log-intensities for the two separate channels, i.e., against what are called the A-values.

[4]For image files from Spot, this usually works fairly well. GenePix gives image files in which the intensities can be much too high. There are alternative to subtracting off the background that may be desirable, perhaps ignoring it altogether.

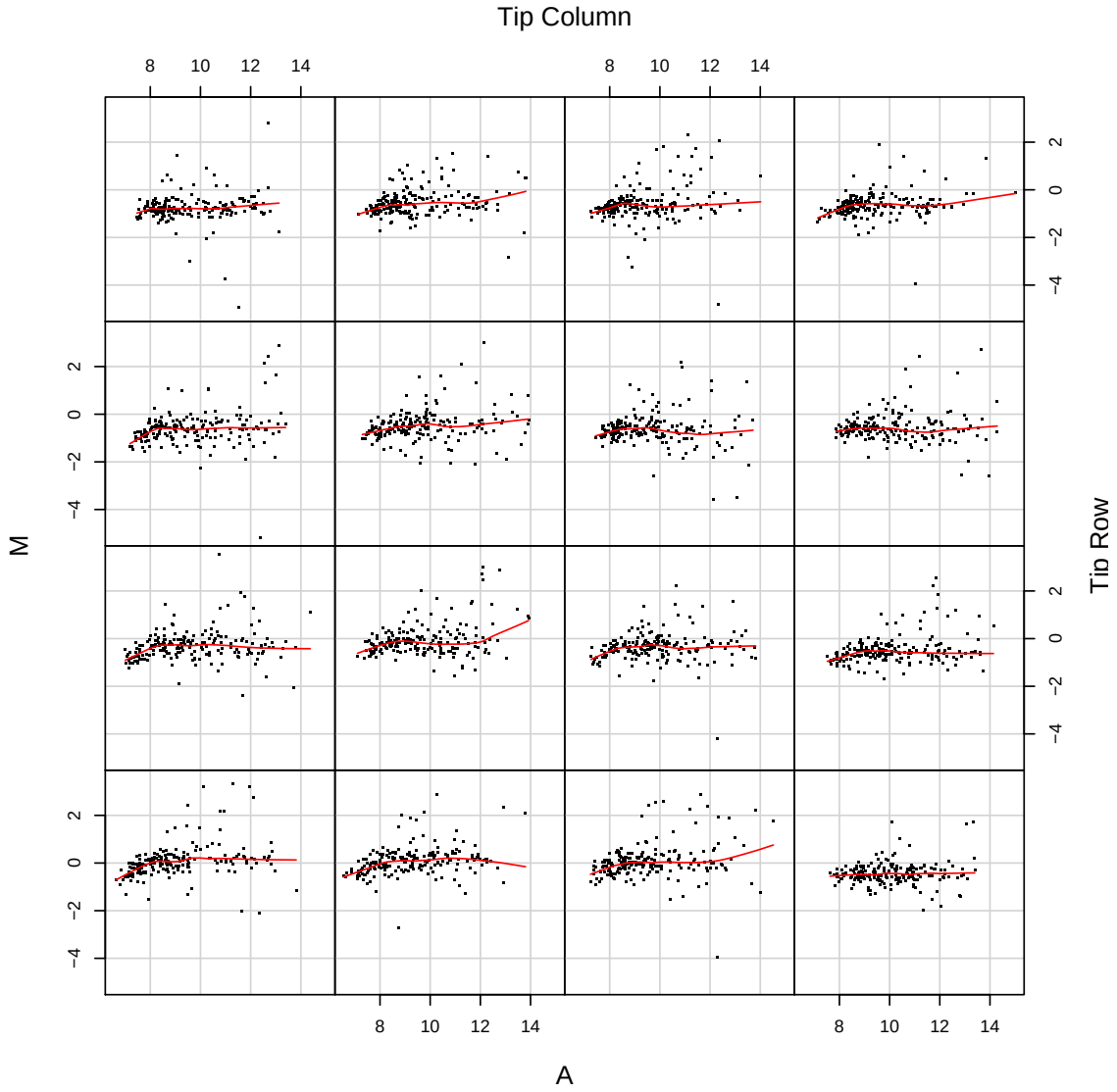
To see all six arrays in a single plot, precede the six plots (first with `array=1`, then `array=1`, ...) with:

```
oldpar <- par(mfrow=c(3,2), mar=c(5.1, 4.1, 1.1, 0.6))
## When done with the 3 by 2 layout, be sure to type
par(oldpar)     # This returns to the original settings.
```

In some of the plots, the dye bias is rather strongly density dependent.

It is also possible to do the following:

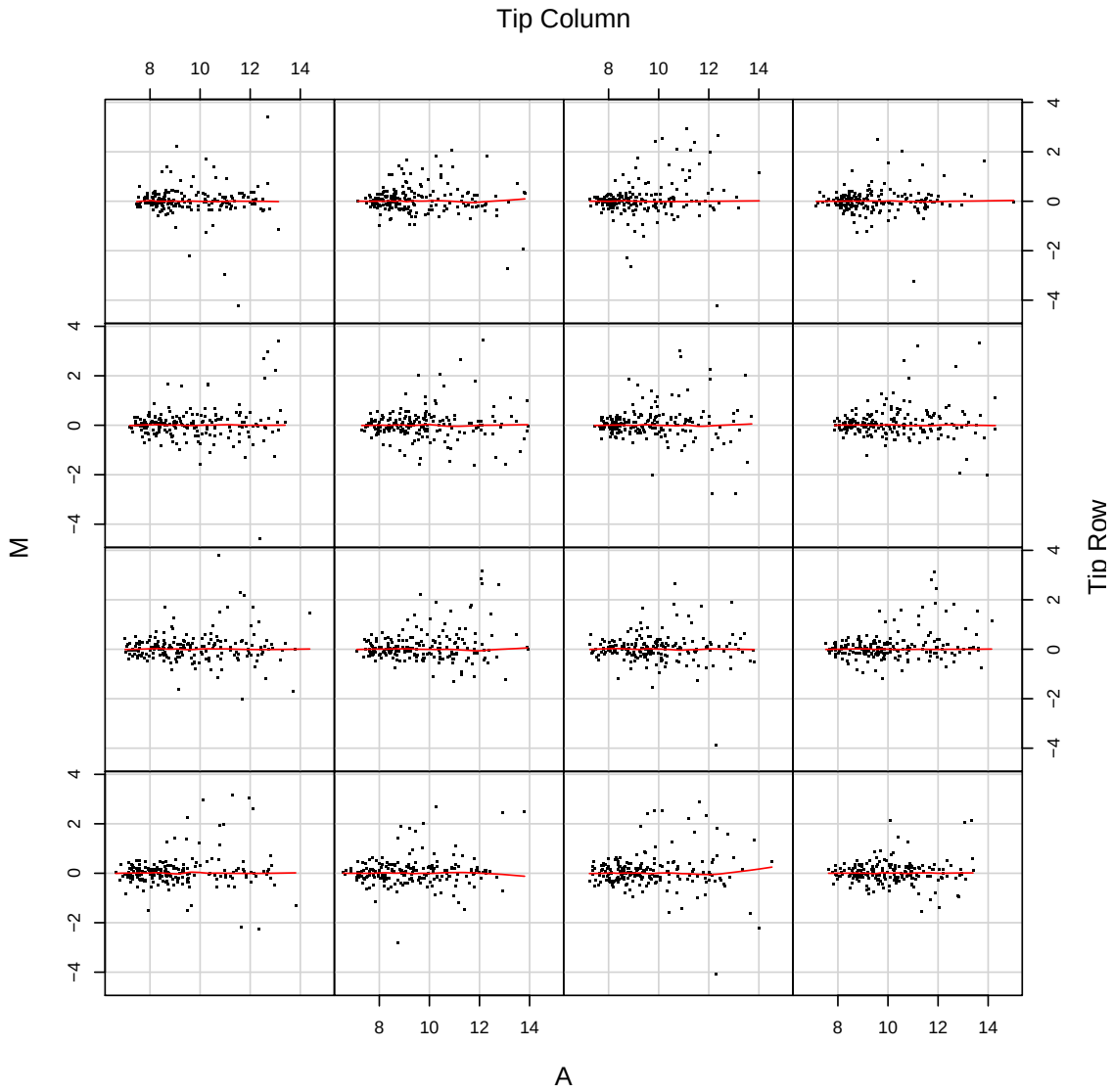
```
rawMA <- normalizeWithinArrays(coralRG, method = "none")
plotPrintTipLoess(rawMA, array=1)
```



A different curve is fitted for each of the print tip groups. There does seem to be some difference in dye bias between the different print tip groups.

Next, we apply print tip loess normalization, and check the MA plots:

```
MA <- normalizeWithinArrays(coralRG, method = "printtiploess")
plotPrintTipLoess(MA)
```



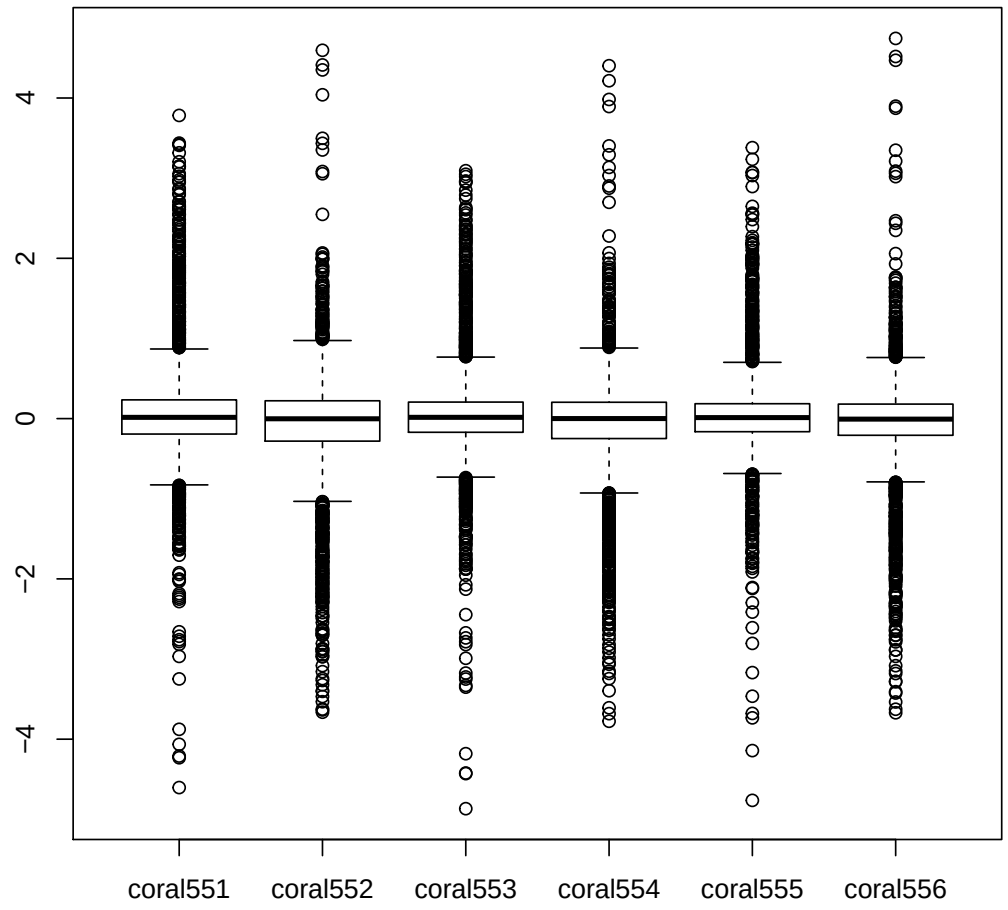
Next, we check whether normalization seems required between arrays:

```
boxplot(MA$M ~ col(MA$M), names = colnames(MA$M))
```

(There is a helpful explanation of boxplots at: `http://davidmlane.com/hyperstat/A38393.html`)

As the arrays seem to have different spreads of M-values, we scale normalize between arrays, and repeat the boxplot:

```
nMA <- normalizeBetweenArrays(MA)
boxplot(nMA$M ~ col(nMA$M), names = colnames(nMA$M))
```



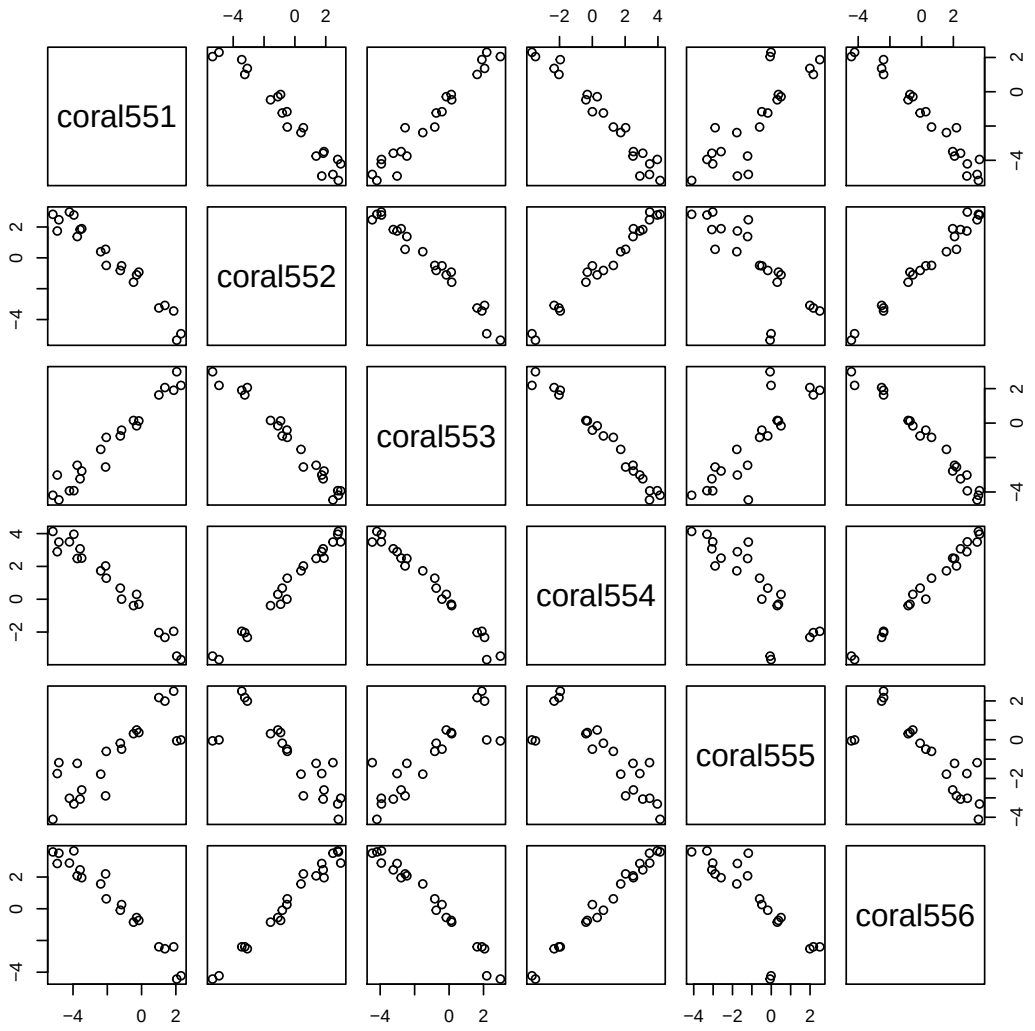
The scaling ensures that the medians, and the upper and lower quartiles, agree across the different slides.

Checks on the Differentially Expressed Controls

The differentially expressed controls ought to show similar evidence of differential expression on all sets of results. Do they?

We extract the M-values for the differentially expressed controls from the total data.

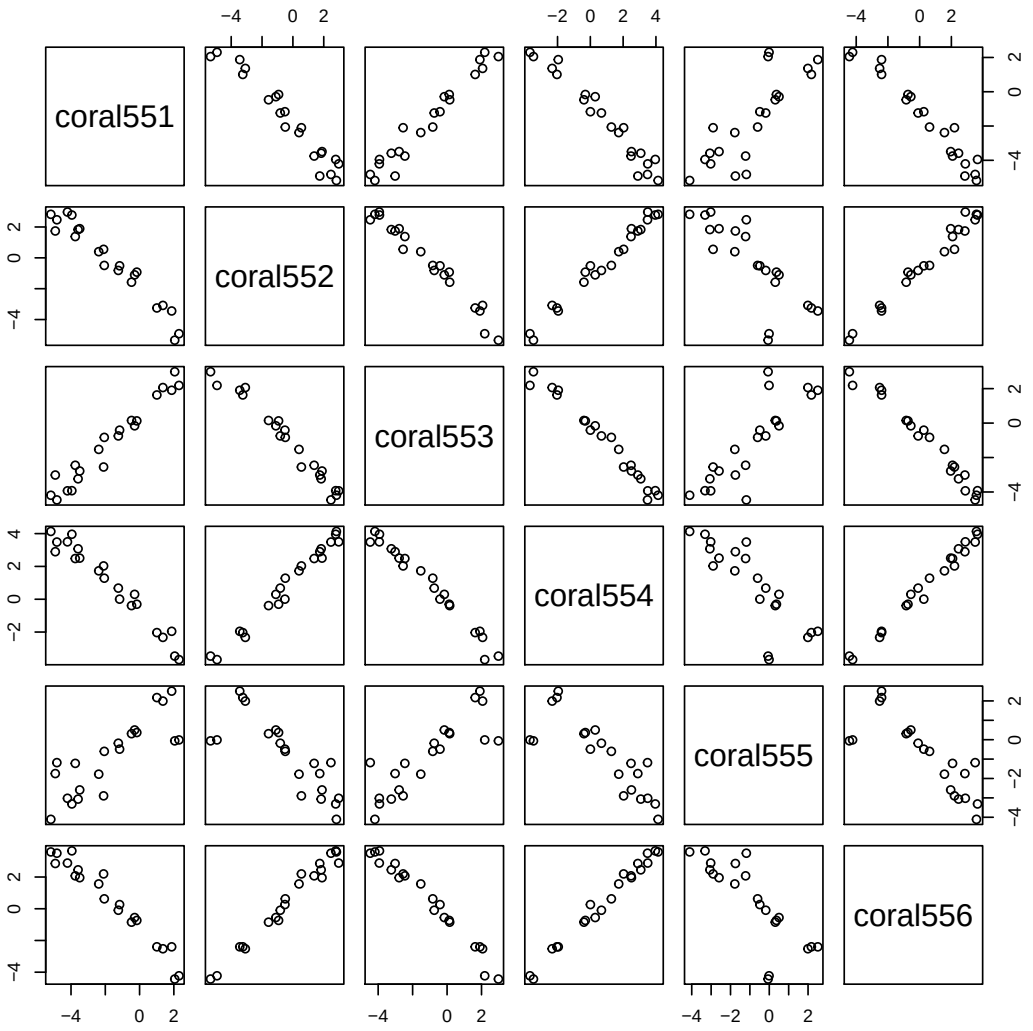
```
wanted <- coralRG$genes$Status == "diff-exp ctl"
rawdeM <- rawMA$M[wanted, ]
pairs(rawdeM)
```



The differentially expressed controls line up remarkably well across the different half-slides. Notice that half-slide 5 is the odd one out. (Why is the correlation sometimes positive and sometimes negative? What is the pattern?)

We now repeat this exercise with the normalized data:

```
wanted <- coralRG$genes$Status == "diff-exp ctl"
deM <- nMA$M[wanted, ]
pairs(rawdeM)
```

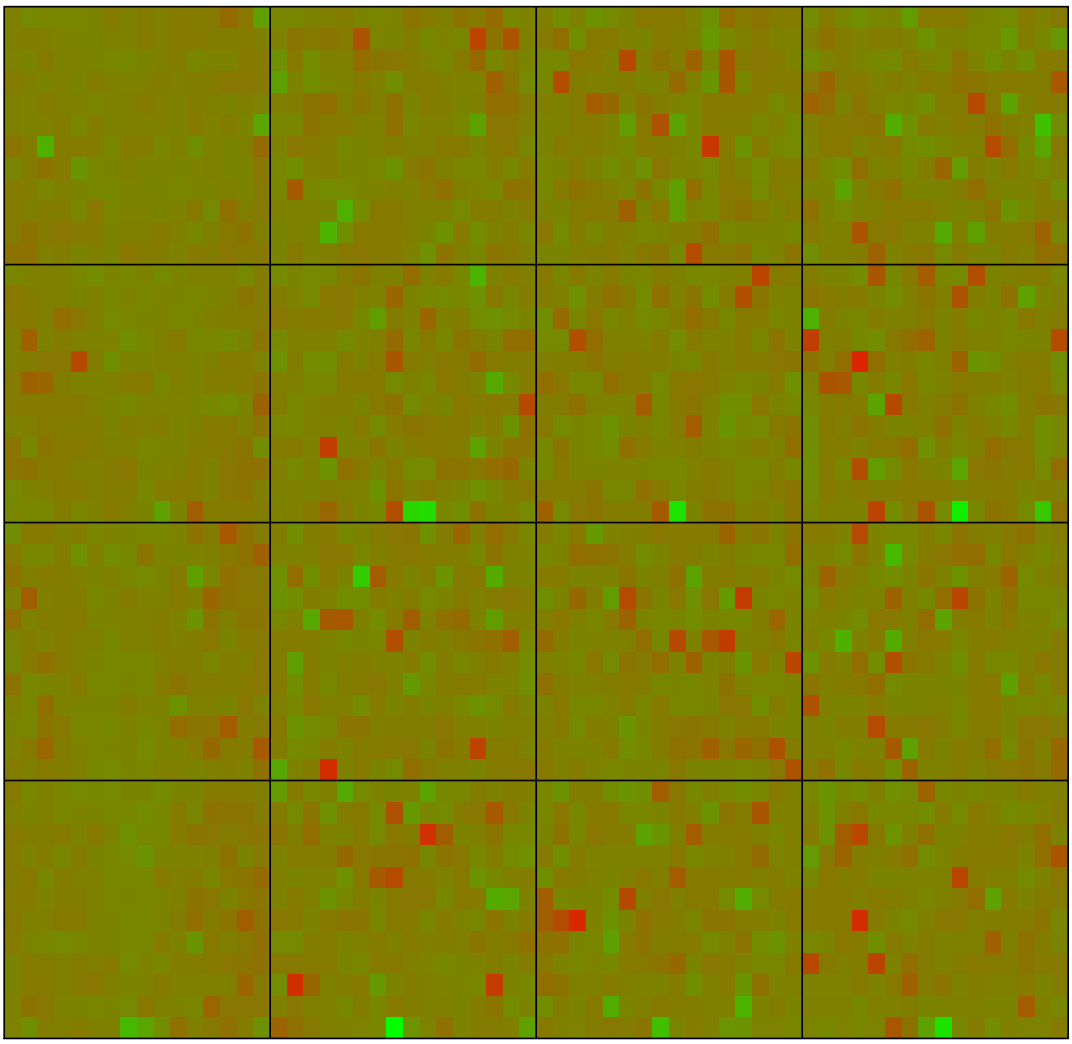


Half-slide 5 is still a maverick. The differentially expressed controls from the other half-slides line up beautifully.
[Dear me! Every family of more than two or three has a misfit. We'll banish half-slide 5 from our assemblage, at least until we can think of something better to do with it.]

More plots

Use `imageplot()` with the M-values on all these half-slides, and look especially at half-slide 5, thus:

```
imageplot(nMA$M[,5], layout=coralRG$printer)
```



z−range −4.8 to 3.4 (saturation −4.8, 4.8)

Something bad clearly happened to the left side of this half-slide!

4 Tests for Differential Expression

We fit a simple statistical model that allows for the possible effect of the dye swap, and which we can use as the basis for checks for differential expression. Recall that M = log(redintensity) − log(greenintensity). Here is how this connects with treatments:

Slide	221a	221b	223a	223b	224a	224b
	log(pre/post)	log(post/pre)	log(pre/post)	log(post/pre)	log(pre/post)	log(post/pre)
Xply by	-1	1	-1	1	0	1

Results for the first and third half-slide are multiplied by -1, so that log(pre/post) (which equals log(pre) − log(post) becomes log(post/pre). The 0 for the fifth half-slide will omit it altogether. This is achieved by placing these numbers into a "design vector".

```
design <- c(-1, 1, -1, 1, 0, 1)
```

Fitting the statistical model

To fit a model that reflects this design, specify:

```
fit <- lmFit(nMA, design)
```

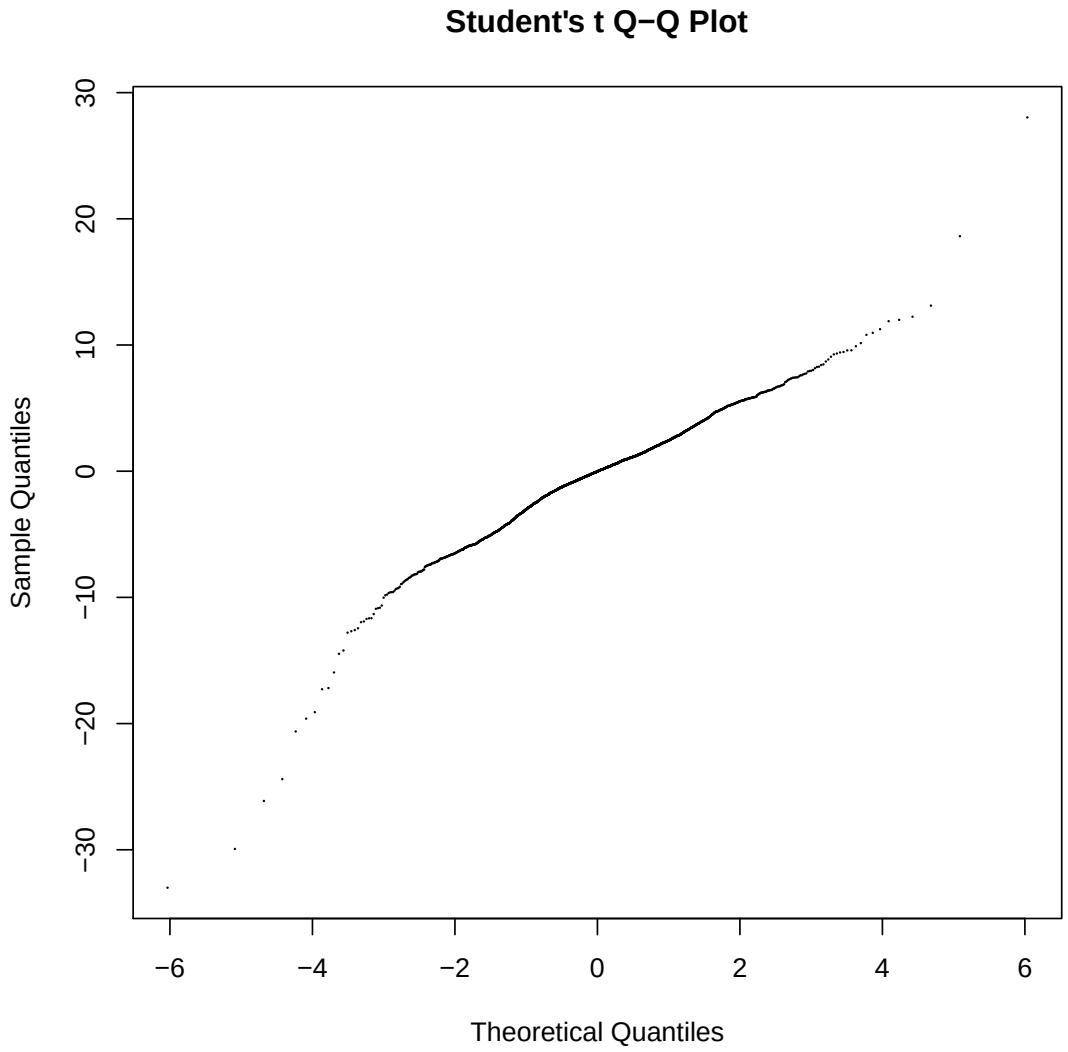
This basically gives t-test results for all the different spots. Any attempt at interpretation has to take account of the large number of tests that have been performed. Also, because there are just 5 sets of usable M-values, the variances that are used in the denominators of the t-statistics, and hence the t-statistics themselves, are more unstable than is desirable.

The calculations that now follow take us into much more speculative territory, where different software uses different approaches. My view is that the calculations that are demonstrated are close to the best that are at present available: They do three things:

- They use information from the variances for all spots, combining this with the spot-specific information, to get more stable t-statistics.
- They make an adjustment for the multiplicity of tests, leading to adjusted t-statistics that can pretty much be compared with the usual one-sample t-critical values.
  [There are many ways to do this. Also the function requires a prior estimate of the proportion of genes that are differentially expressed, by default set to 0.01, i.e. there is a large element of subjective judgment.]
- For each gene they give an odds ratio (B = Bayes factor) that it is differentially expressed.
  [This relies on the same prior estimate of the proportion of genes that are differentially expressed.]

From the fit object, we calculate empirical Bayes moderated t-statistics and do a qq-plot:

```
efit <- eBayes(fit)
qqt(efit$t, df = efit$df.prior + efit$df.residual, pch = 16,
    cex = 0.2)
```



The majority of genes lie on a line that goes through the centre of the plot. Those that lie off this (below it on the left, or above it on the right) are most likely to be differentially expressed. [5]

Next we print out a table that shows the top 50 differentially expressed genes, in order of the moderated t-statistic values:

```
options(digits = 3)
topvals <- topTable(efit, number = 50)
topvals
```

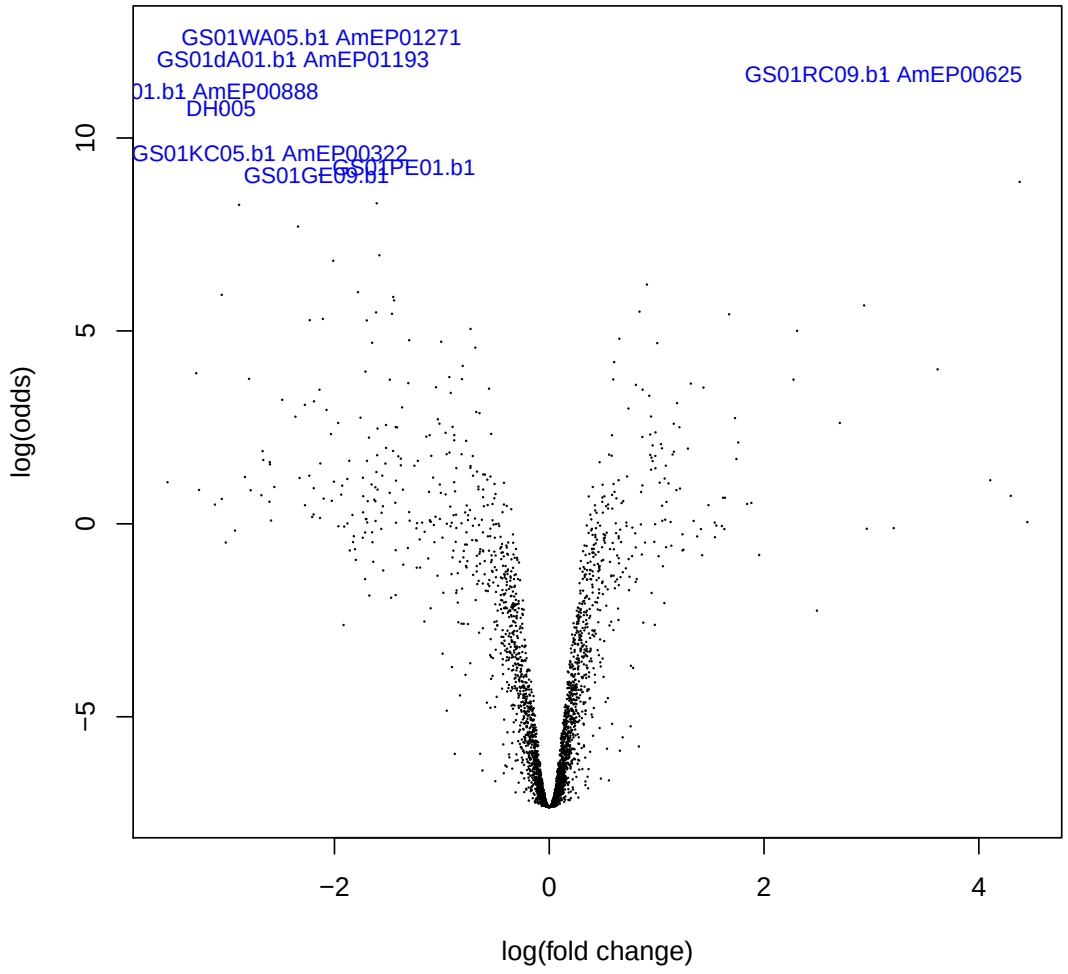
We store the results in `topvals` prior to displaying them, as they will be used later. Notice that an adjusted p-value of 0.05 corresponds to a B-statistic of about 3.6. This may be a reasonably cut-off point.

The order of genes on the list is much more secure than the B-values and P-values. The extent to which the statistics are affected by the prior probability will be demonstrated in the exercise below.

[5] If the results from the different genes were independent, the line on which most of the "genes" lie would be a 45° line, with a slope of 1.0. If you wish, type `abline(0, 1, col="red")` to put in such a line anyway. The line on which most of the genes lie has a much steeper slope than this 45° line.

Here is another type of plot:

```
plot(efit$coef, efit$lods, pch = 16, cex = 0.2, xlab = "log(fold change)",
    ylab = "log(odds)")
ord <- order(efit$lods, decreasing = TRUE)
top8 <- ord[1:8]
text(efit$coef[top8], efit$lods[top8], labels = coralRG$genes[top8,
    "Name"], cex = 0.8, col = "blue")
```



Exercise: Change the prior probability to 0.02, i.e.

```
efit.02 <- eBayes(fit, prop=0.02)
topTable(efit.02, number = 50)
```

Observe what difference this makes to the list. Try also `prob=0.1`. A good way to see the effect is to plot the `P.Value` or `B` from the separate fits, one against the other:

```
efit.1 <- eBayes(fit, prop=0.1)
B.1 <- topTable(efit.1, number = 3072)$B
B.01 <- topvals$B
points(B.01, B.1, col="gray")
```

Do the equivalent plots for `P.Value`.

5 More refined analyses

5.1 Reading Data into R, with weight information incorporated

Read the data in, with weighting information included:

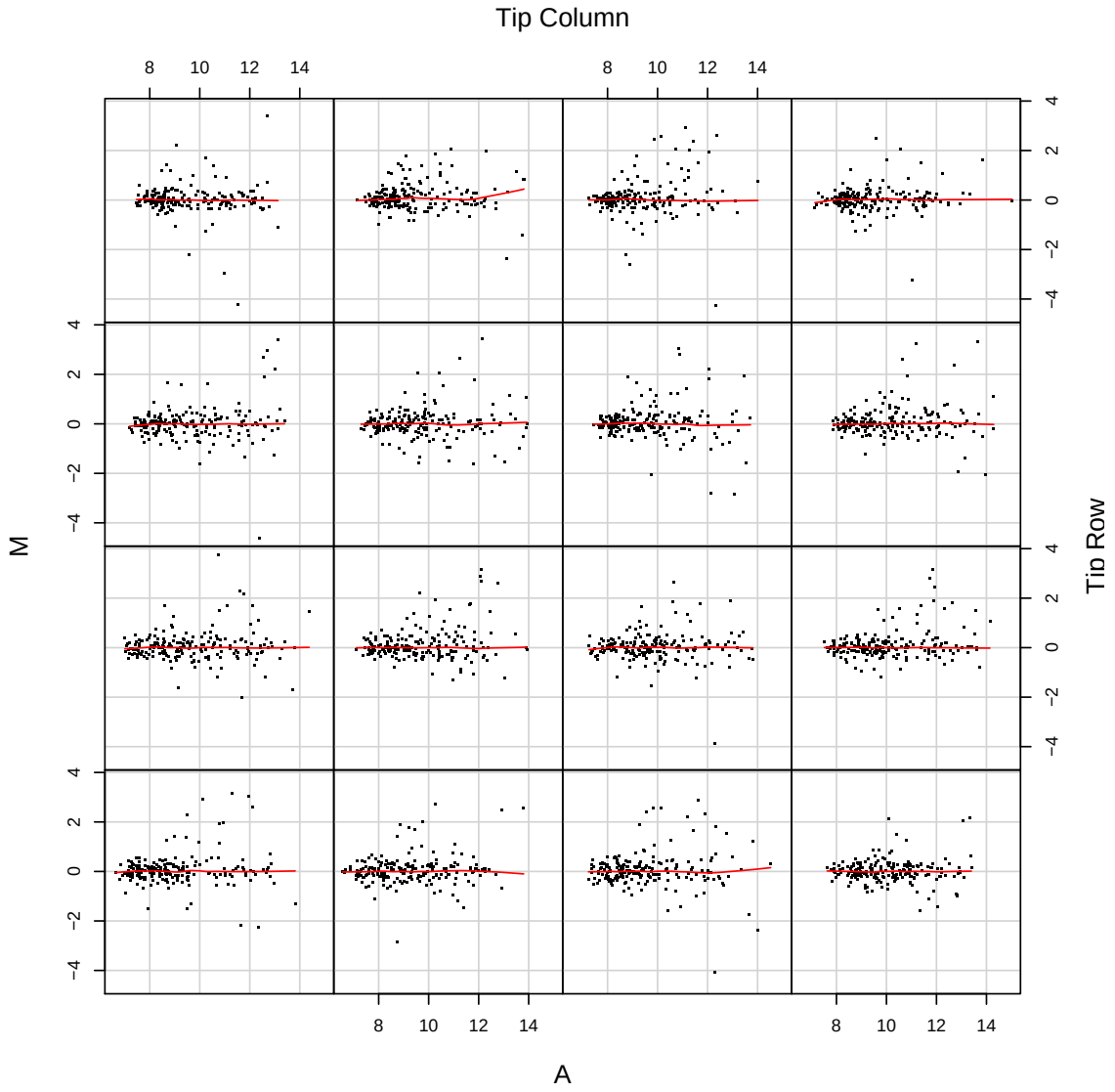
```
coral2RG <- read.maimages(targets$FileName,
                          source = "spot",
                          path=path2data,
                          wt.fun=wtarea(100))
coral2RG$genes <- readGAL(path=path2data)
coral2RG$printer <- getLayout(coral2RG$genes)
```

The weights will be used automatically by functions that operate on `coral2RG`.

5.2 MA plots & Normalization

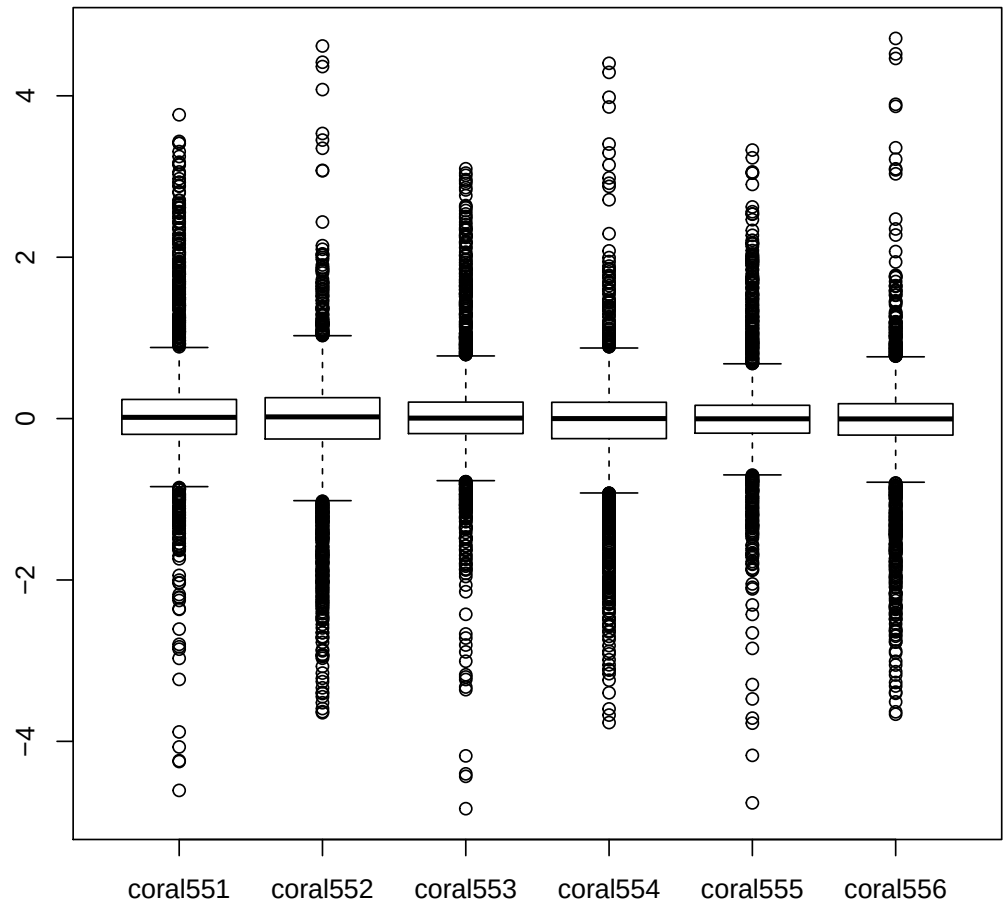
Apply print tip loess normalization, and check the MA plots:

```
MA2 <- normalizeWithinArrays(coral2RG, method = "printtiploess")
plotPrintTipLoess(MA2)
```



Next, we check whether normalization seems required between arrays, then scale normalize between arrays, and repeat the boxplot:

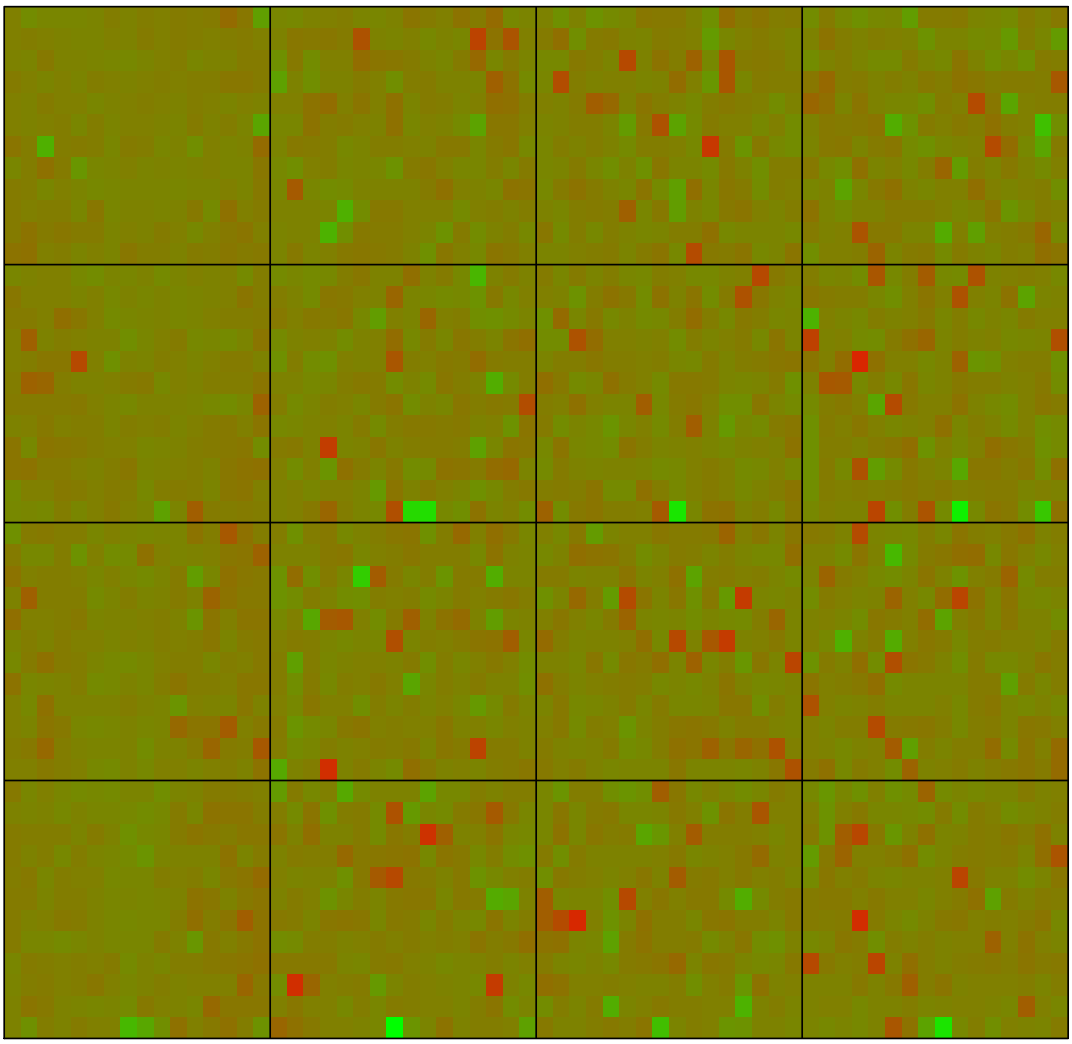
```
boxplot(MA2$M ~ col(MA2$M), names = colnames(MA2$M))
nMA2 <- normalizeBetweenArrays(MA2)
boxplot(nMA2$M ~ col(nMA2$M), names = colnames(nMA2$M))
```



Slide 5, again!

Use `imageplot()` with the M-values on all these half-slides, and look especially at half-slide 5, thus:

```
imageplot(nMA2$M[,5], layout=coral2RG$printer)
```



z−range −4.8 to 3.3 (saturation −4.8, 4.8)

Use of the quality information has not made much difference to this plot:

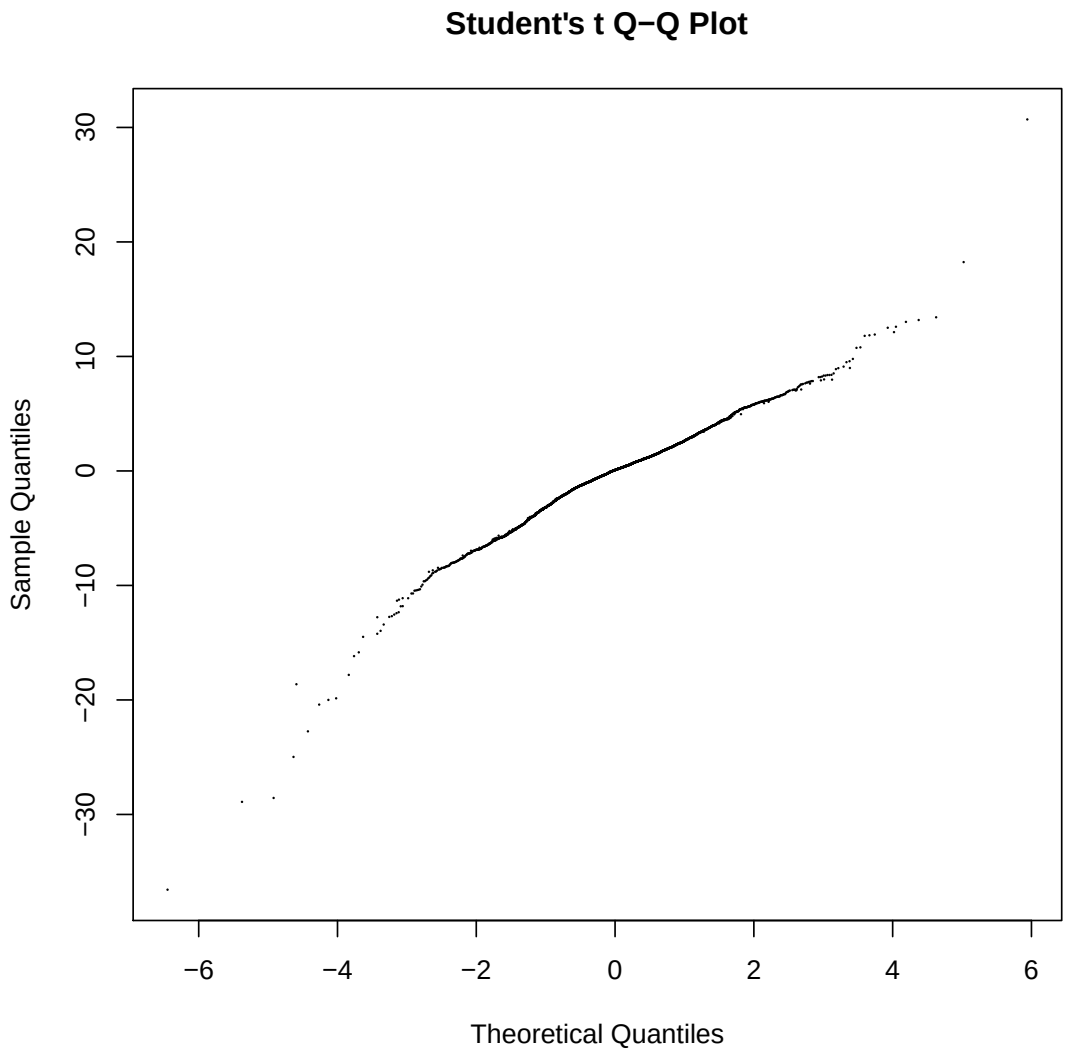
Fitting the statistical model

To fit a model that reflects this design, specify:

```
design <- c(-1, 1, -1, 1, 0, 1)
fit2 <- lmFit(nMA2, design)
```

From the fit object, we calculate empirical Bayes moderated t-statistics and do a qq-plot:

```
efit2 <- eBayes(fit2)
qqt(efit2$t, df = efit2$df.prior + efit2$df.residual, pch = 16,
    cex = 0.2)
```



Next we print out a table that shows the top 50 differentially expressed genes, in order of the moderated t-statistic values:

```
options(digits = 3)
topTable(efit2, number = 50)
```

Now see what different the use of weights has made to the list (the vector `topvals` was found earlier):

```
## Get & store results with & without weights
topvals2 <- topTable(efit2, number = 50)
cbind(row.names(topvals), row.names(topvals2))
```

Now check how many of the genes are in common across both lists:

```
sum(row.names(topvals)%in%row.names(topvals2))
```

Appendix

Installation of the R software

First install R (2.10.0 or later). (Versions are available for Unix, Linux, Windows and Macintosh OS X.)

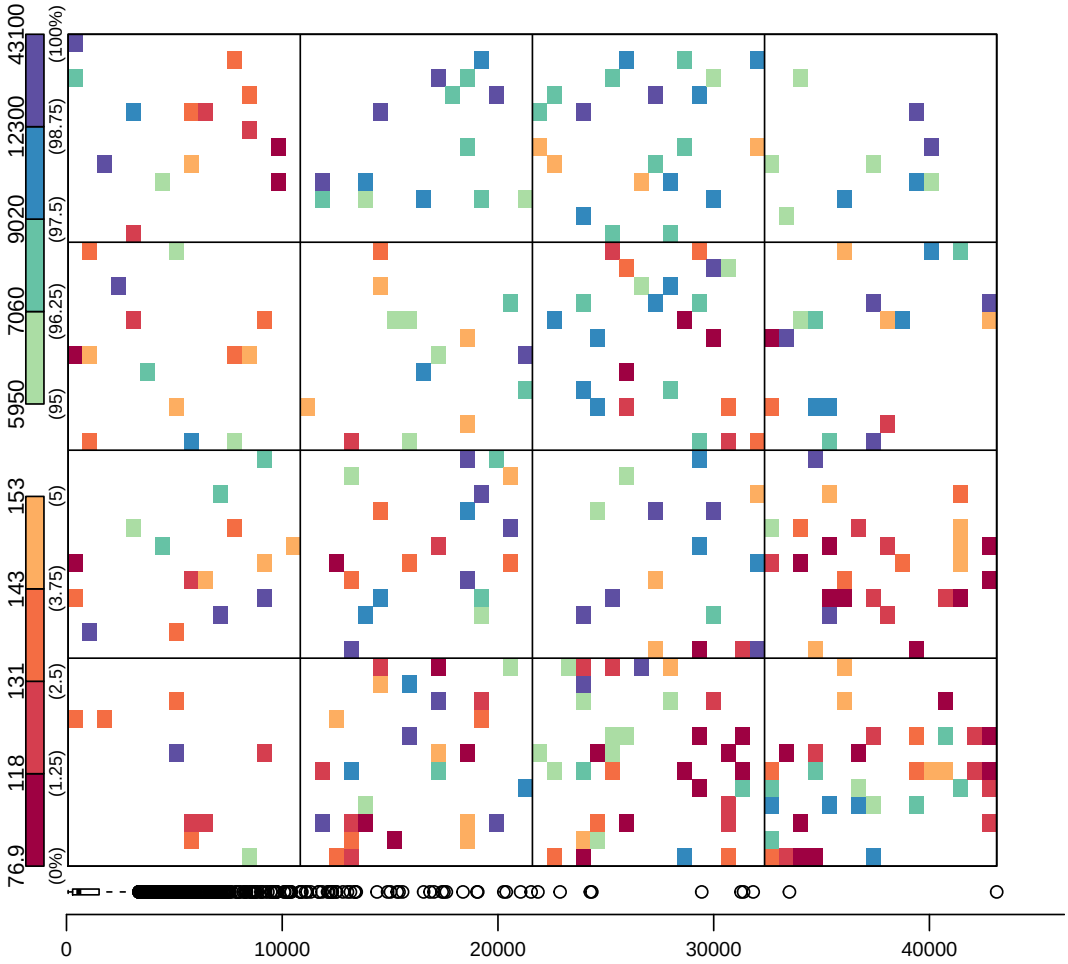
Install the *limma* and *DAAGbio* packages. Install *DAAGbio* package from CRAN.

See `http://www.bioconductor.org/packages/release/bioc/html/limma.html` for instructions on installing *limma*.

5.3 `imgplot()` as an alternative to `imageplot()`

The function `imgplot()` gives a display that is a bit different from `imageplot()`. Try:

```
imgplot(coralRG$R[, 1], layout = coralRG$printer)
```



By default, this shows just the smallest 5% of intensities and the largest 5% of intensities.

Use these to look for parts of the half-slide that all show a consistent difference from the rest of the half-slide. Remember however that what is important is the pattern that appears after

background correction (if any), normalization and the calculation of the M-values.

How was this document produced?

The document **marray-notes.Rnw** can be copied into the working directory.

Alternatively, from within an R session, enter:

```
library(knitr)
knit("marray-notes.Rnw")
```

This produces a LaTeX document that can then be processed through the LaTeX document preparation system to give a postscript or pdf file.

This is a great way to document your eventual analyses. Changes to the code in the **.Rnw** document are automatically reflected in the LaTeX document that comes from `knit()`. See `help(knit)` for information on documentation.

References

Smythe, G., Thorne, T., and Wettenhall, J. 2004. limma: Linear Models for Microarray Data User's Guide. This document is included with *limma* distributions.

Beare, R. and Buckley, M. 2004. Spot:cDNA Microarray Image Analysis Users Guide. Available from `http://spot.cmis.csiro.au/spot/spotmanual.php`.

Acknowledgements

Many thanks to Gaby Samuels, Eldon Ball, David Hayward and Rob Saint (Centre for Molecular Genetics of Development) for permission to use the coral data for this workshop. The associated research was supported by ARC grants DP0343727 (John Maindonald), S4116004 (Lauretta Grasso, Gaby Samuels, Eldon Ball, David Hayward and Rob Saint, through CMGD). Conrad Burden did a marvelous job of checking these notes for mistakes.