Seminar III: R/Bioconductor

Leonardo Collado Torres
lcollado@lcg.unam.mx
Bachelor in Genomic Sciences
www.lcg.unam.mx/~lcollado/

August - December, 2009

Basic microarrays

Intro

Linear regressions

Correlations

limma

Homework

About

- On this short class we'll learn how to do linear regressions, correlations and use the limma package.

Session packages

- Install commands:

```
> install.packages("UsingR")
> source("http://bioconductor.org/biocLite.R")
> biocLite(c("limma"))
```

Quick intro

- The idea behind a linear regression is to fit a line to a given data set. This line will try to pass through the center of the data, such that the data points are close.
- Once you have the linear regression you can predict values :)
- There are a lot of methods of linear regression models, but we'll take a look at the simplest one.

Basic linear regression

- Basic equation:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \tag{1}$$

- Here, ϵ_i is the error, β_0 and β_1 are the regression coefficients, x is the independent variable & y the dependent one.
- In statistics, we generally know x but need to estimate the rest.

Functions

- In R we can do some simple linear regressions using lm (model.formula) where we use `y TILDE x` for the formula. For example:

```
> library(UsingR)
> res <- lm(homedata$y2000 ~ homedata$y1970)
> res

Call:
lm(formula = homedata$y2000 ~ homedata$y1970)

Coefficients:
    (Intercept)  homedata$y1970
     -1.040e+05       5.258e+00
```

Functions

- Instead of just calling `lm`, its better to save the resulting object. Then we can use this object to plot it or obtain more information with functions such as:
  - coef gives us the coefficients
  - residuals to find the residuals
  - predict to predict a value for a given x
- Note that linear regressions are not meant to predict values outside the valid range for your independent variable (x).
- Sometimes its useful to transform your data so that the linear model will be more appropriate. For example, log transform the data.
- Some other linear regression functions which are more resistant to outliers are: lqs and rlm.

Plotting the lm object

- How would you plot the linear regression? Its a line :)

With abline

- Its not with lines but with abline:

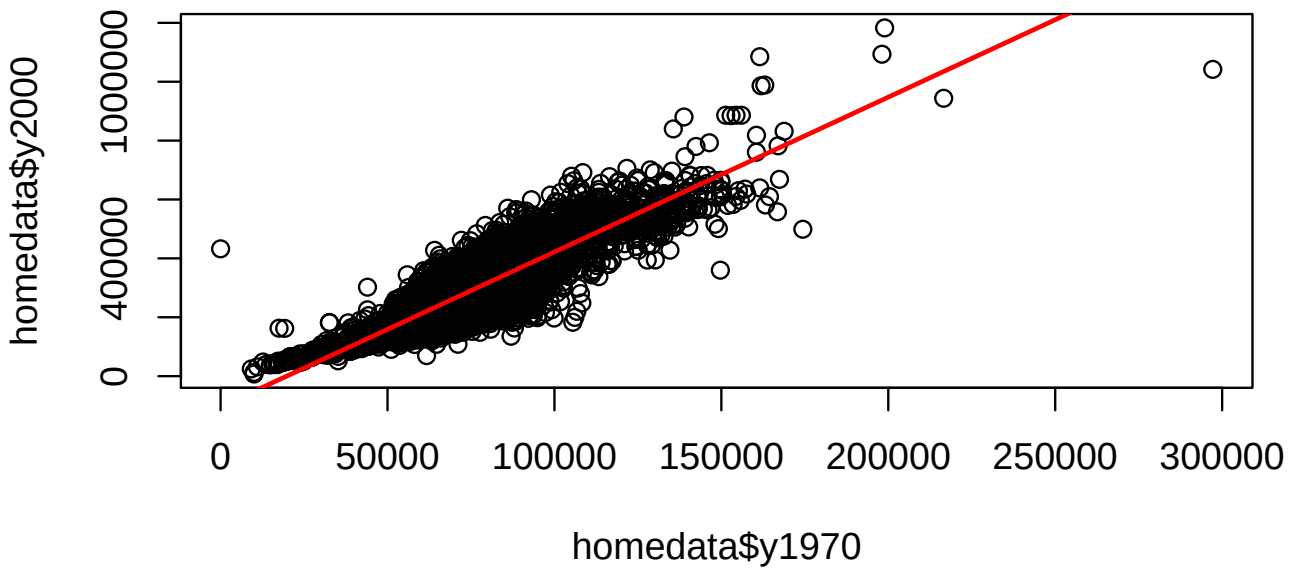
```
> args(abline)

function (a = NULL, b = NULL, h = NULL, v = NULL, reg = NULL,
    coef = NULL, untf = FALSE, ...)
NULL
```

- Now plot the data and the regression line :)

Plot

```
> plot(homedata$y1970, homedata$y2000)
> abline(res, col = "red", lwd = 2)
```



Exercise

1. Using the `kid.weights` data frame, make a plot of weight vs height).

```
> head(kid.weights)
```

```
  age weight height gender
1  58     38     38      M
2 103     87     43      M
3  87     50     48      M
4 138     98     61      M
5  82     47     47      F
6  52     30     24      F
```

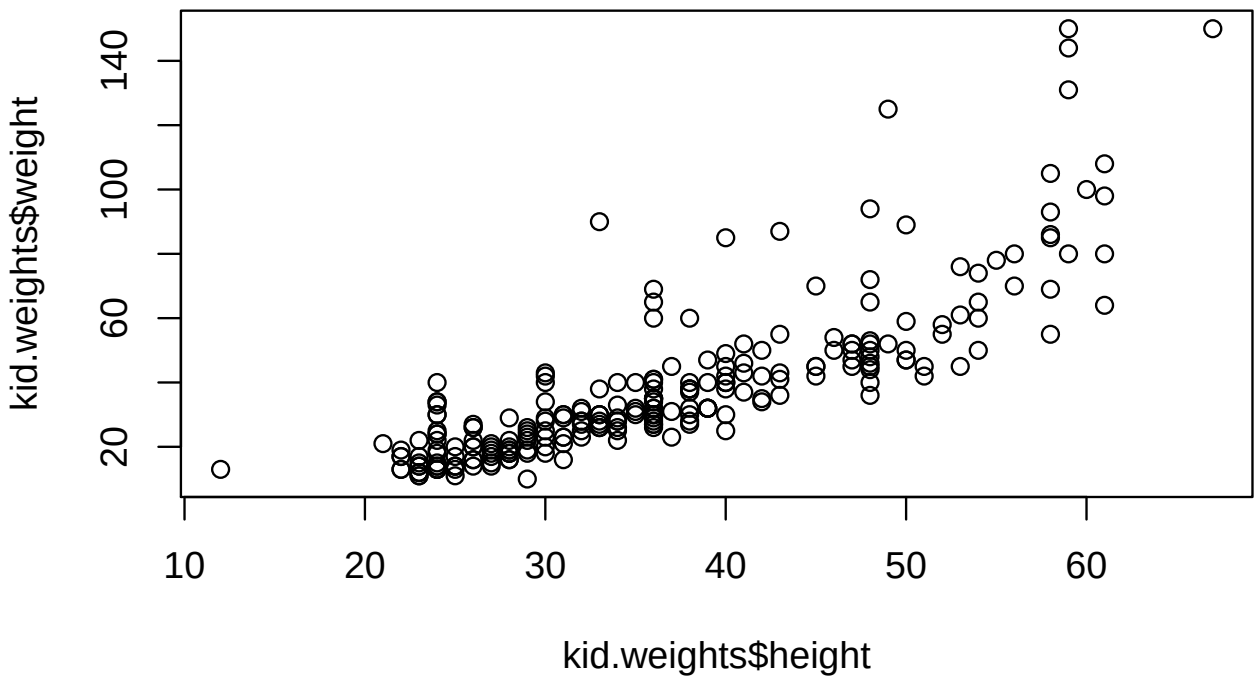
2. Transform one of the variables to make the plot better.
3. Re-make the plot with the transformed variable.

Exercise

4. Do a simple linear regression.
5. Plot the line from your resulting lm object.

Part 1

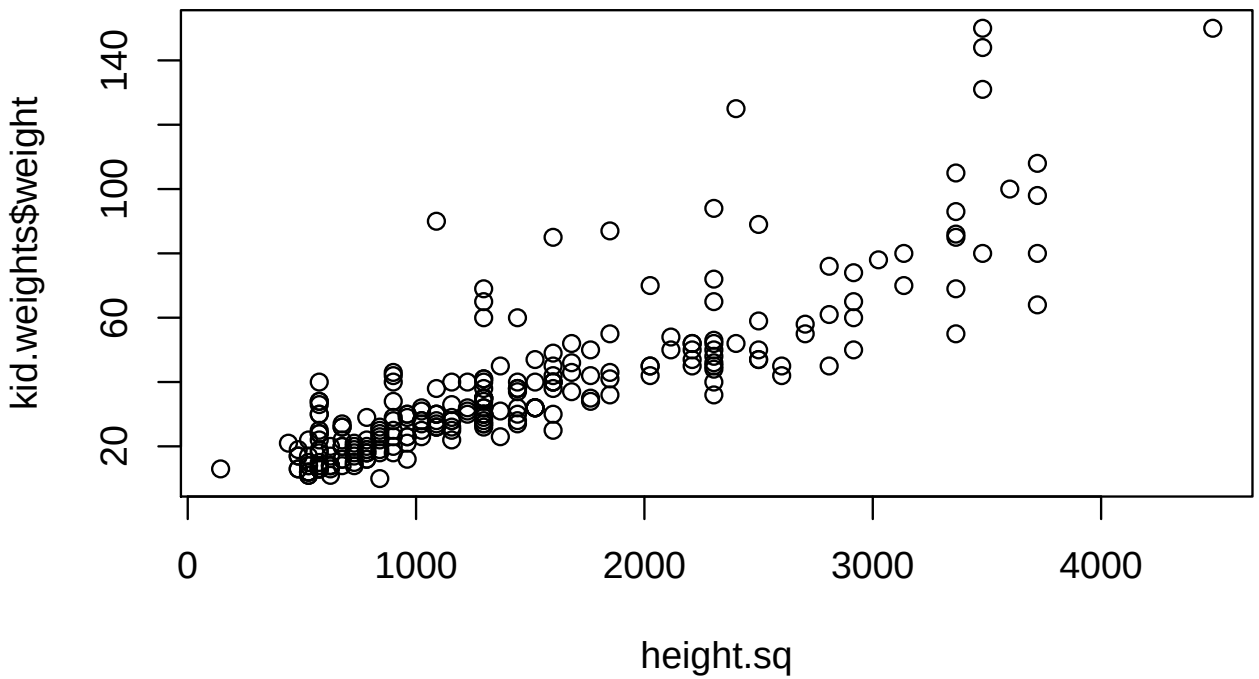
```
> plot(kid.weights$weight ~ kid.weights$height)
```



Parts 2 and 3

```
> height.sq <- kid.weights$height^2
> plot(kid.weights$weight ~ height.sq)
```

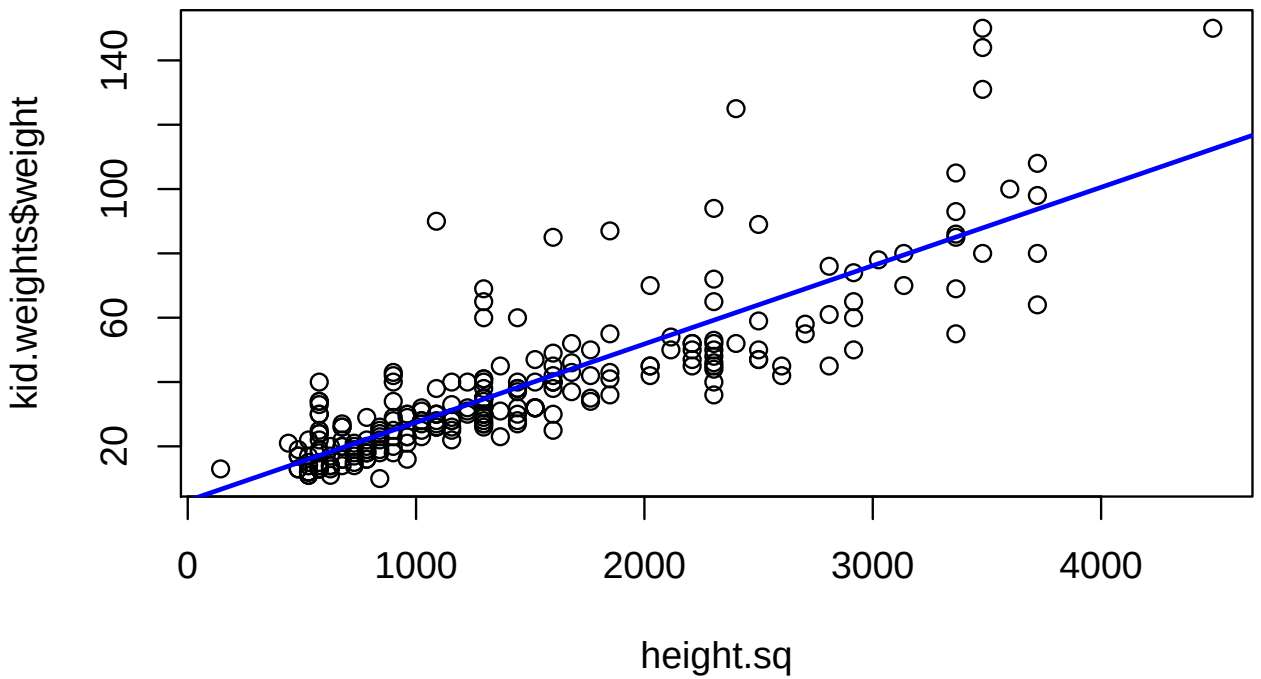
Parts 2 and 3


kid.weights$weight
20
60
100
140
0
1000
2000
3000
4000
height.sq

Parts 4 and 5

```
> height.sq <- kid.weights$height^2
> res2 <- lm(kid.weights$weight ~
+     height.sq)
> plot(kid.weights$weight ~ height.sq)
> abline(res2, col = "blue", lwd = 2)
```

Parts 4 and 5


kid.weights$weight
20
60
100
140
0
1000
2000
3000
4000
height.sq

Correlations

- Many times when you have two variables you'll want to know if they are correlated. A correlation as taken by wiki esp:
  - *indicates the force and direction of a linear relationship between two random variables. Two quantitative variables are considered to be correlated when the values of one of them varies systematically with respect to the homomin values from the other one: if we have two variables (A and B) there is a correlation when the values of A increase so do the values of B and viceversa. The correlation between two variables doesn't imply, by itself, any relation of casuality.*
- In R we can find the coefficient of the Pearson and Spearman correlations easily.

Pearson correlation

- It tells us how correlated are two variables.[1]
- For more information, check wikipedia.
- In R we can find the Pearson correlation using the function cor:

```
> cor(homedata$y1970, homedata$y2000)
[1] 0.8962155
> cor(maydow$max.temp[-1], diff(maydow$DJA))
[1] 0.01028846
```

[1] Its independent of the scale of the variables.

Spearman rank correlation

- This one is useful if the relationship between the two variables is not lineal, but increases. More info at wiki.
- To use this one, we need to order the data from smallest to biggest into ranks. We can do so using the function rank:

```
> args(rank)

function (x, na.last = TRUE, ties.method = c("average", "first",
    "random", "max", "min"))
NULL
```

- In R we can calculate this correlation using `cor(rank(x), rank(y))` or simply using the argument `method` from the `cor` function.

```
> cor(rank(homedata$y1970), rank(homedata$y2000))

[1] 0.8878185
```

Spearman rank correlation

```
> cor(maydow$max.temp[-1], diff(maydow$DJA),
+     method = "spearman")

[1] 0.1315711
```

Quick exercise

1. Find Pearson correlation coefficient for the `kid.weights` data set (height versus weight).
2. Find the Spearman rank correlation coefficient for the same data.
3. Are the two variables correlated? The answer could be different.

Solution

```
> cor(kid.weights$height, kid.weights$weight)

[1] 0.8237564

> cor(kid.weights$height, kid.weights$weight,
+     method = "s")

[1] 0.8822136
```

- They are correlated with both methods :)

limma: intro

- Limma info.
- `limma` was created to analyse microarrays, linear relationships and to find the genes diferentially expressed.
- Some packages derived from limma are `limmaGUI` and `affylmGUI` while `marray` is in some way its competitor.
- We'll only take a peak at part of the package because it's very extense.

Problem

- In the basic situation with `limma` we work with 4 measurements per gene in a microarray. Two colors are used: Cy3 and Cy5. The first measurements are like this: WT Cy3, Experiment Cy5. Then the colors are exchanged for the second set.
- We have data from *zebrafish*, which is used to study the early development in vertebrates. Swirl is a point mutation for the gene BMP2 that affects the dorsal/ventral axis of the body. Our objective is to use the data from this experiment to find the genes with an expression level altered in this mutant compared to the WT.

Data

- To start, please download these files into the same directory:
  - fish.gal
  - swirl.1.spot
  - swirl.2.spot
  - swirl.3.spot
  - swirl.4.spot
  - SpotTypes.txt
  - SwirlSample.txt
- Then open `R` from that directory (browse to it in Unix), or use `setwd`.

readTargets

- Use the function readTargets to read the table describing our experiment.

```
> library(limma)
> targets <- readTargets("SwirlSample.txt")
> targets
```

```
  SlideNumber     FileName       Cy3
1          81 swirl.1.spot     swirl
2          82 swirl.2.spot wild type
3          93 swirl.3.spot     swirl
4          94 swirl.4.spot wild type
        Cy5      Date
1 wild type 2001/9/20
2     swirl 2001/9/20
```

readTargets

```
3 wild type 2001/11/8
4     swirl 2001/11/8
```

- Our input files are not raw files because they were read with an Axos scaner to produce `TIFF` images which were then analysed with the `SPOT` software.

read.maimages

- Using read.maimages we can read the files generated by SPOT.
- We can read the *foreground* and *background* intensities (green and red colors).

```
> RG <- read.maimages(targets$FileName,
+      source = "spot")
```

```
Read swirl.1.spot
Read swirl.2.spot
Read swirl.3.spot
Read swirl.4.spot
```

- With our object targets we can get the file names. Now check RG.

```
> RG
```

readGAL

- How many data points do we have? We have 8...
- In the `GAL` file we have the name of each gene associated with a data point. We can read this information with readGAL:

```
> RG$genes <- readGAL("fish.gal")
```

getLayout

- Now we have lots of information on our microarray, but there is more :)
- Using getLayout we can get the settings for the microarry printer

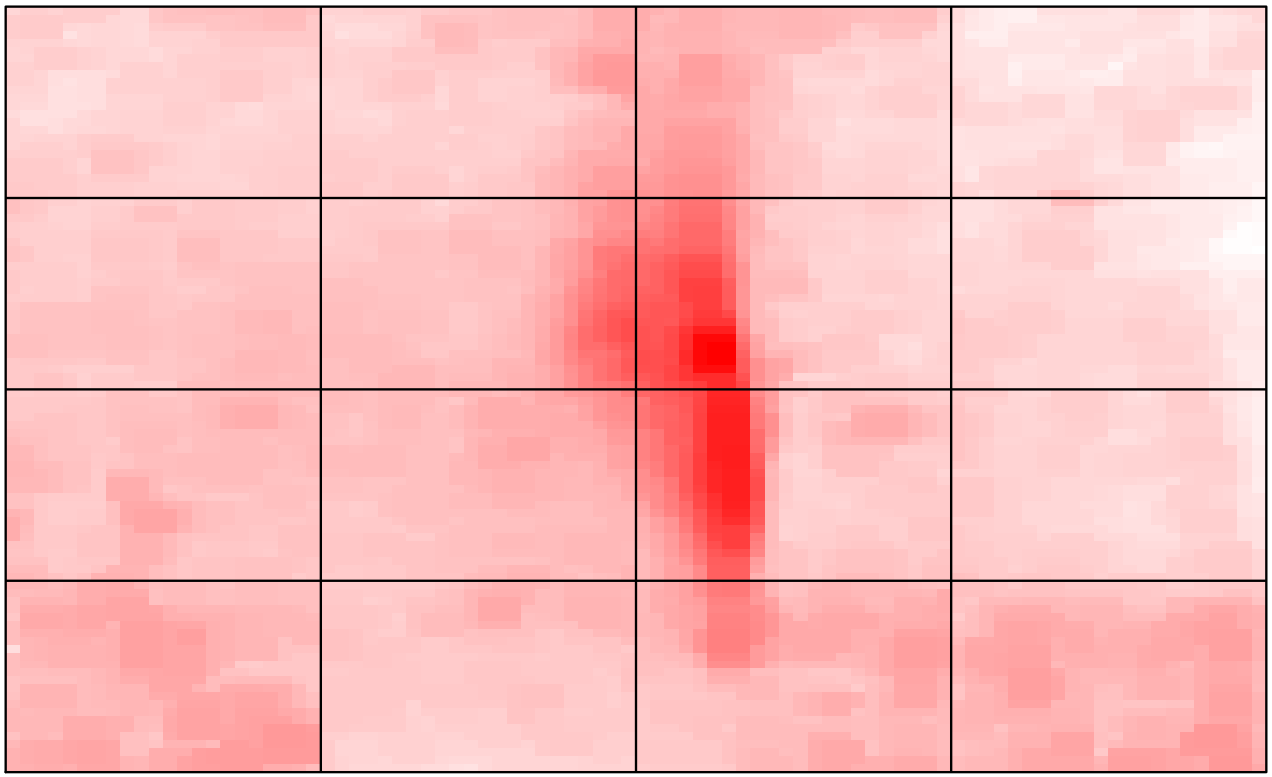
```
> RG$printer <- getLayout(RG$genes)
```

imageplot

- Similar to `image`, we can use imageplot to explore our microarray.
- It's helpful to explore the *background*.

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

imageplot

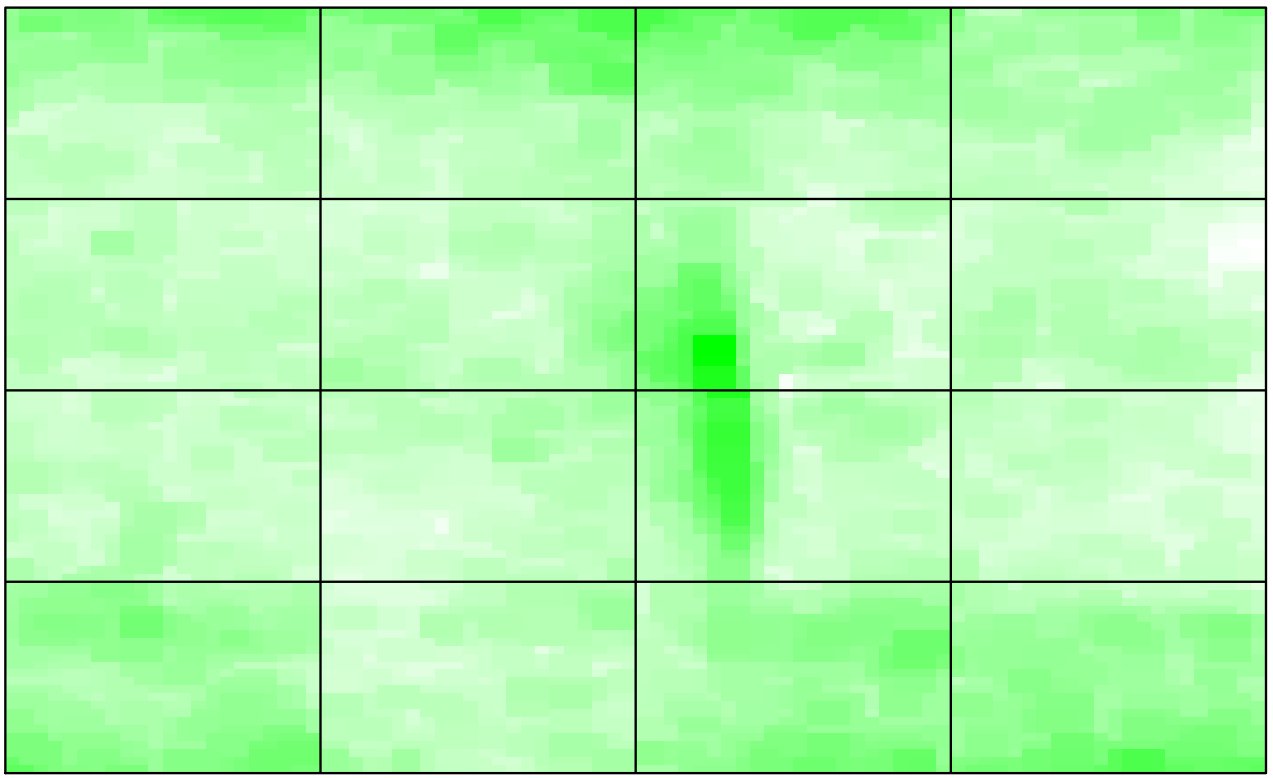


z−range 5.9 to 11.1 (saturation 5.9, 11.1)

imageplot green

```
> imageplot(log2(RG$Gb[, 1]), RG$printer,
+     low = "white", high = "green")
```

imageplot green



z−range 6.2 to 8.2 (saturation 6.2, 8.2)

normalizeWithinArrays

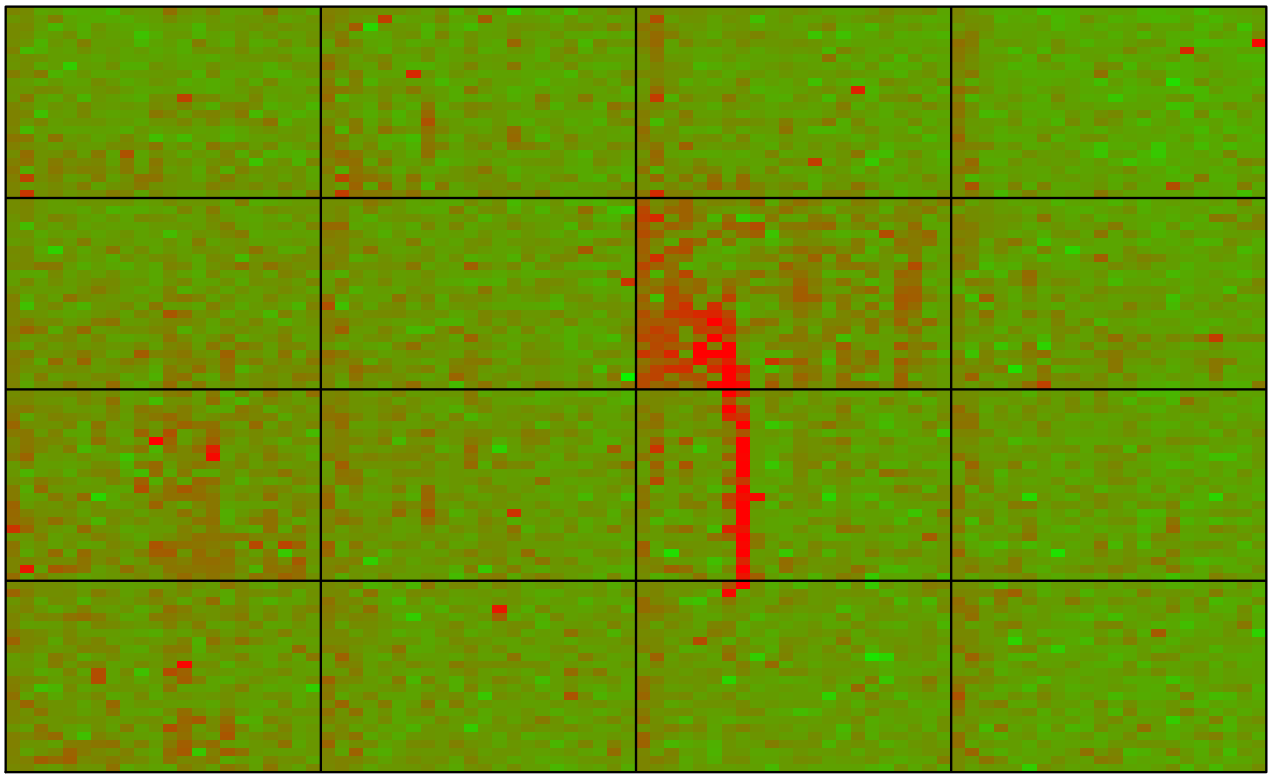
- Just by looking at that first array we realized that we need to normalize the data.
- Using normalizeWithinArrays we can normalize the data using its *log-ratio* such that the mean of these will be 0..

```
> MA <- normalizeWithinArrays(RG,
+      method = "none")
```

- Lets check if we got a satisfying result:

```
> imageplot(MA$M[, 1], RG$printer,
+      zlim = c(-3, 3))
```

normalizeWithinArrays



z−range −2.7 to 4.4 (saturation −3, 3)

o_O

- The function `imageplot` rotates the array, such that the group at the bottom left corner is the first one.
- In this last plot we observe a red line. Till tells us that there was some powder or that the microarray was damaged there.
- The data from that zone will have suspicious values.

plotMA

- In microarrays, its useful to make a "MA" plot.
- In these, we plot the ratio R vs G versus the intensity of the point.
- The value M is determined by:

$$M = log_2(R) - log_2(G)$$

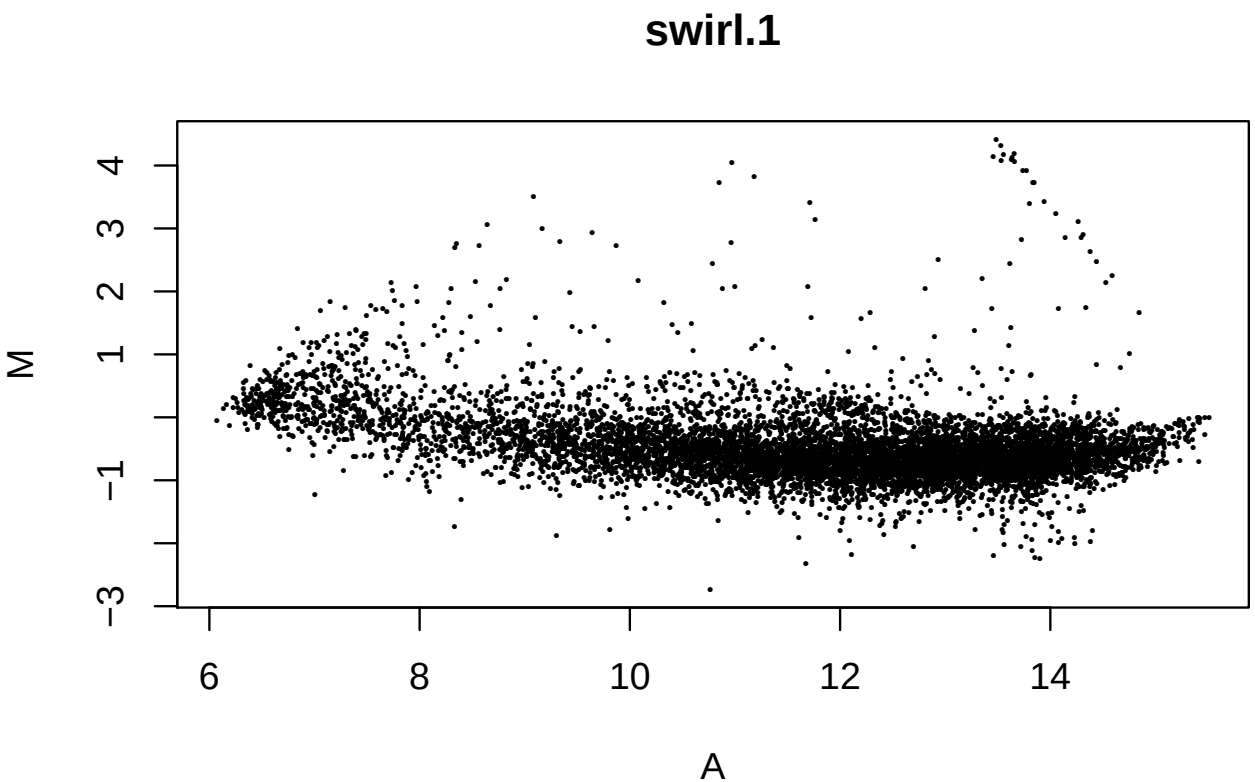
- THe value A represents the intensity and is given by:or:

$$A = (log_2(R) + log_2(G))/2$$

- We can make this kind of plot using plotMA.

```
> plotMA(MA)
```

plotMA

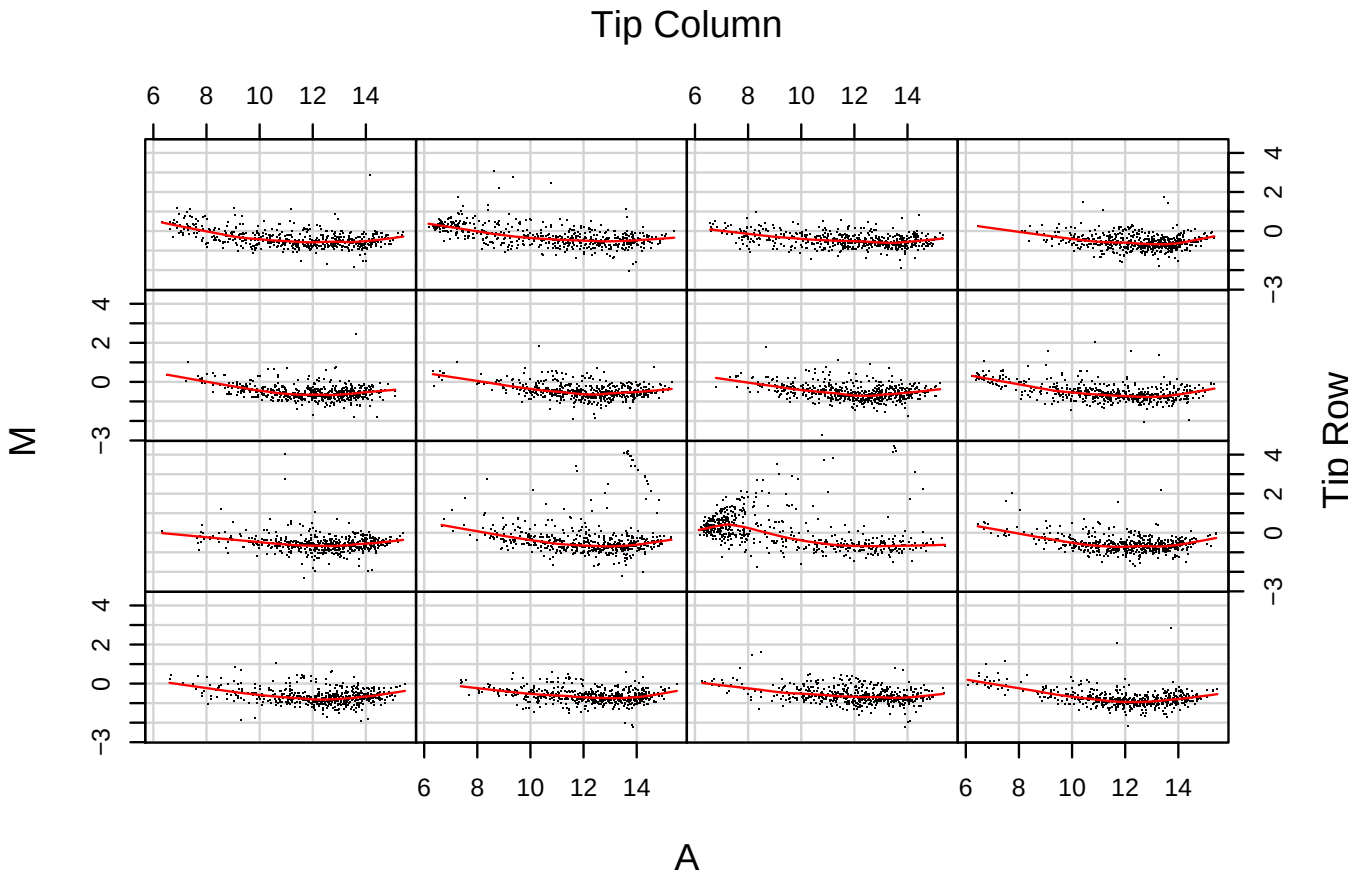


plotPrintTipLoess

- In the previous plot we can see how the values derived from the damaged zone are in the top right hand corner.
- When we have lots of data, we need to deal with *outliers*. That's why we use functions such as lowess and loess. `lowess` makes a "locally weighted polynomial regression". Its older hence why it doesn't use the formula notation.
- We can use plotPrintTipLoess to visualize all the data from our first array and the loess curve which we'll use to normalize the data.

```
> plotPrintTipLoess(MA)
```

plotPrintTipLoess

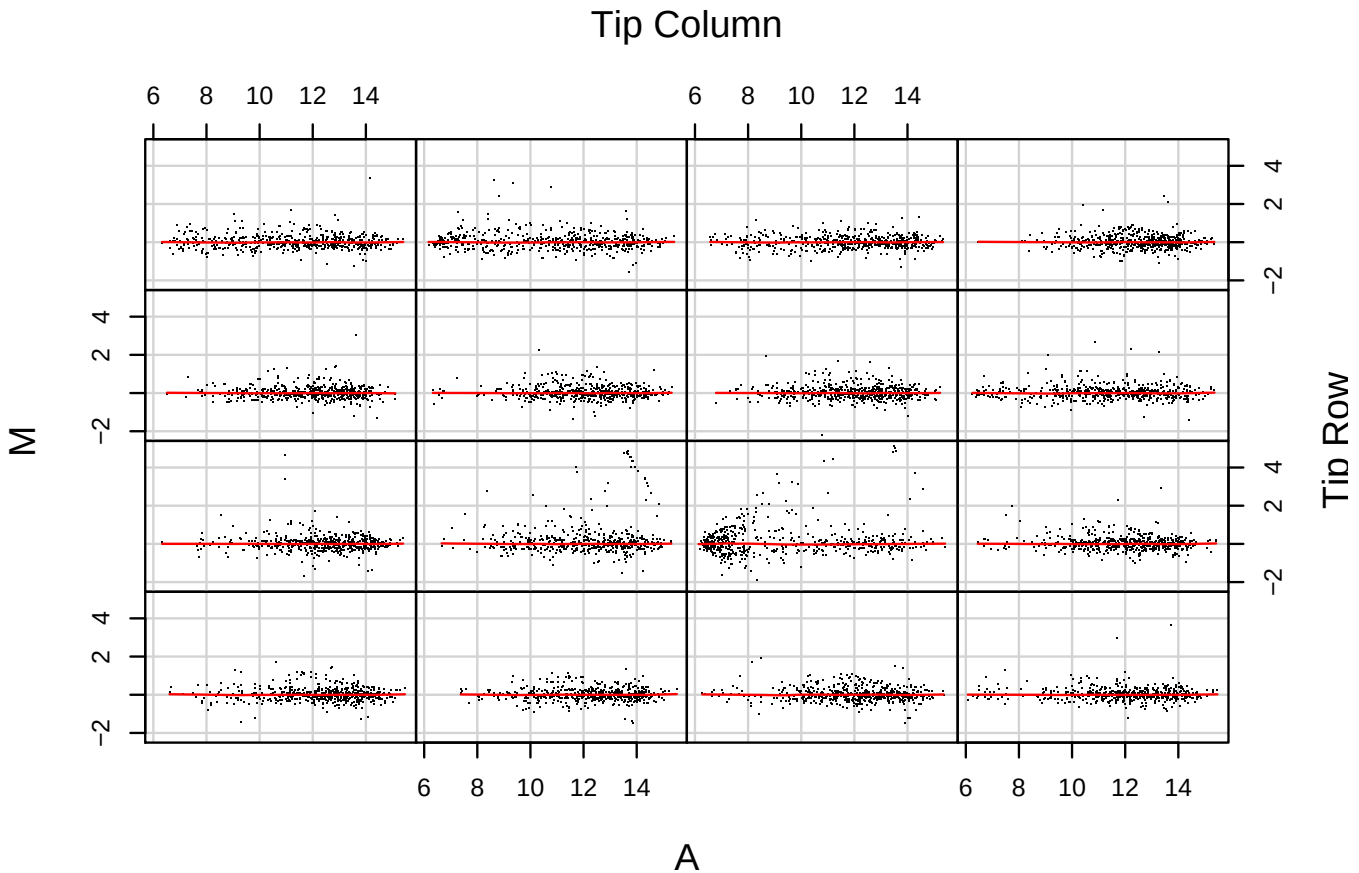


Normalizing

- Now lets normalize the data using default params and lets look at the same plot again.
- In reality we are only normalizing the M values for each array.

```
> MA <- normalizeWithinArrays(RG)
> plotPrintTipLoess(MA)
```

Normalizing

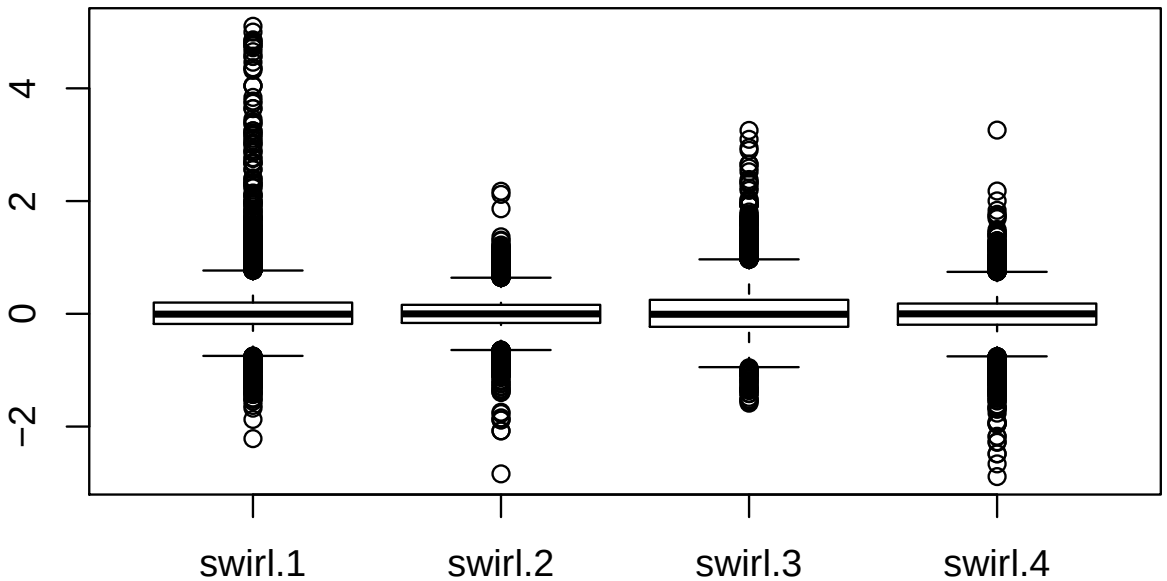


Between arrays?

- Next we can ask ourselves if we need to normalize between the 4 microarrays.
- To do so, we'll use the basic `R` function `boxplot`:

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

Between arrays?

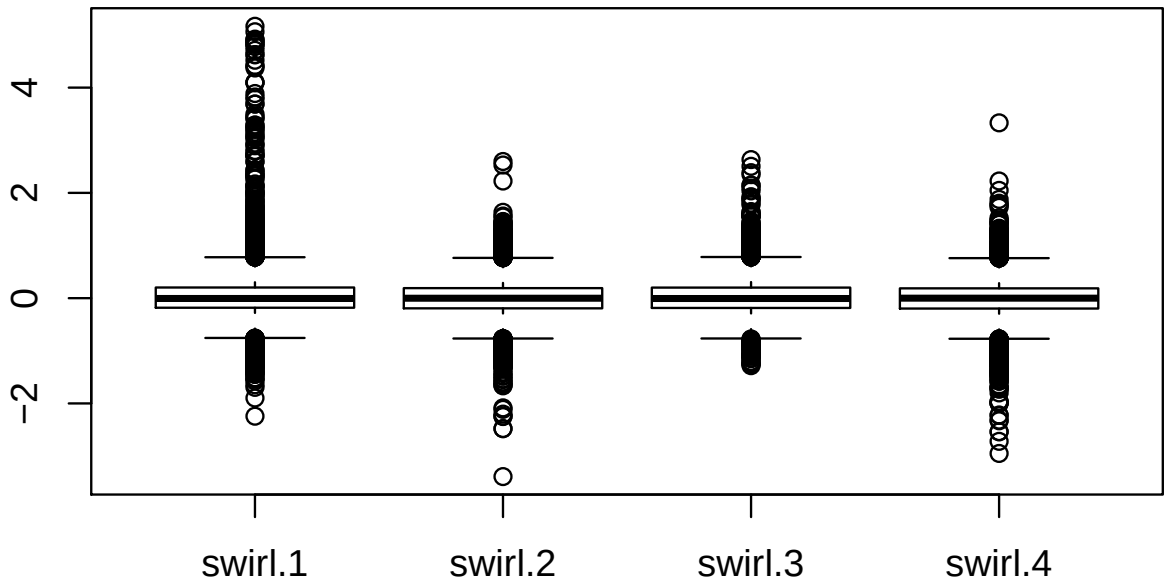


normalizeBetweenArrays

- Because we can observe variation between the arrays, lets normalize them. Many times this is not necessary.
- Let use normalizeBetweenArrays with the default method and look at the result with `boxplot`.
- The default method is `Aquantile` which makes sure that the A values have the same empirical distribution in the arrays without changing the M values.

```
> MA <- normalizeBetweenArrays(MA,
+     method = "scale")
> boxplot(MA$M ~ col(MA$M), names = colnames(MA$M))
```

normalizeBetweenArrays



lmFit

- Now we'll use a linear model to estimate (predict) the value M for every gene.
- First we need to specify the experimental design; when every color was used:

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

- Next we find our linear regression using the function lmFit which is especifically designed for microarrays.

```
> fit <- lmFit(MA, design)
```

- The resulting object has lots of information. Take a look!

```
> fit
```

t tests

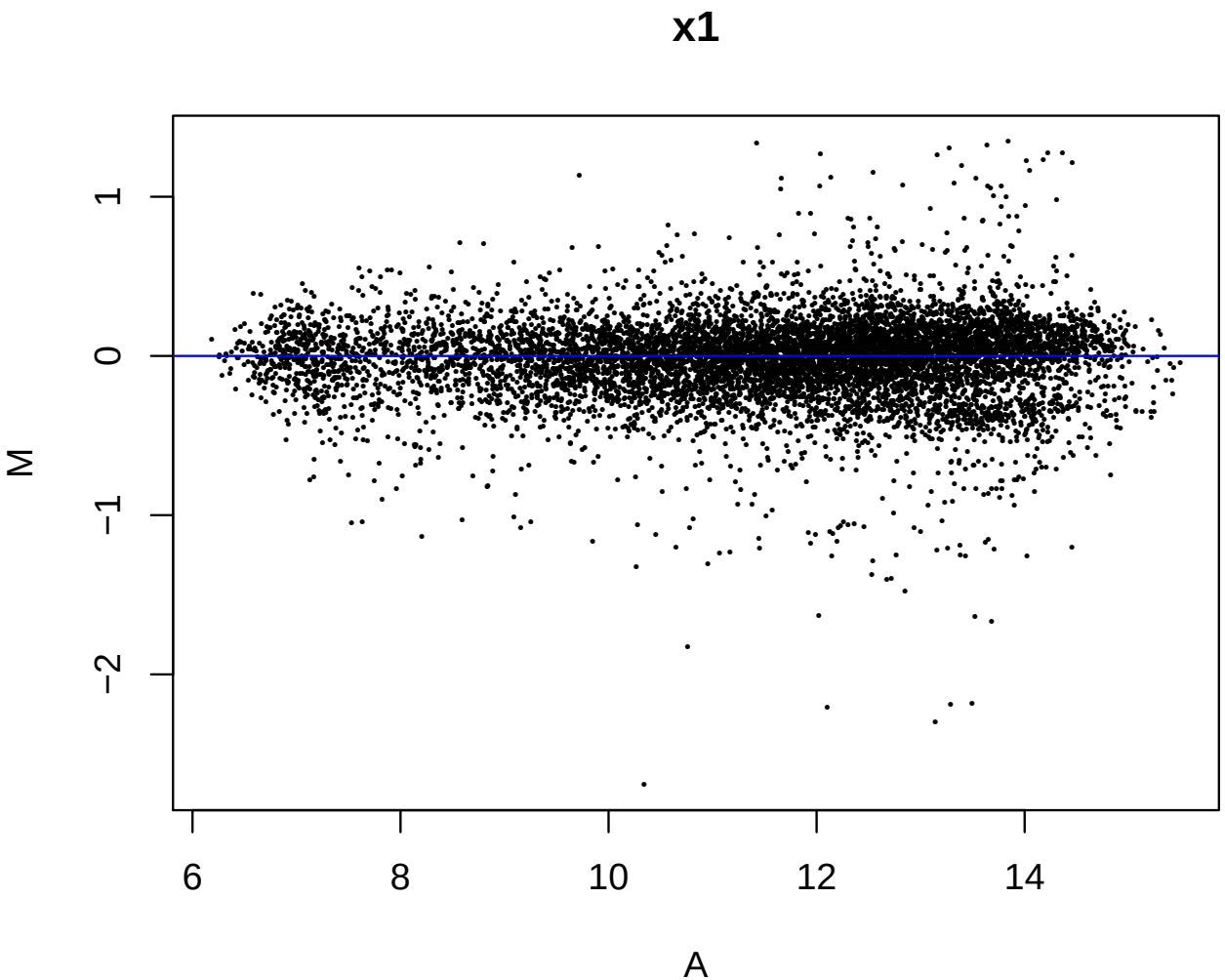
- In our objet `fit`, `coefficients` is the mean M value while `sigma` is the standard deviation for each gene.
- We can now make t tests to compare the mutant with the WT for every gene:

```
> ordinary.t <- fit$coef/fit$stdev.unscaled/fit$sigma
```

- Now we can make a plot with the mean M and A values for every gene:

```
> plotMA(fit)
> abline(0, 0, col = "blue")
```

t tests



eBayes

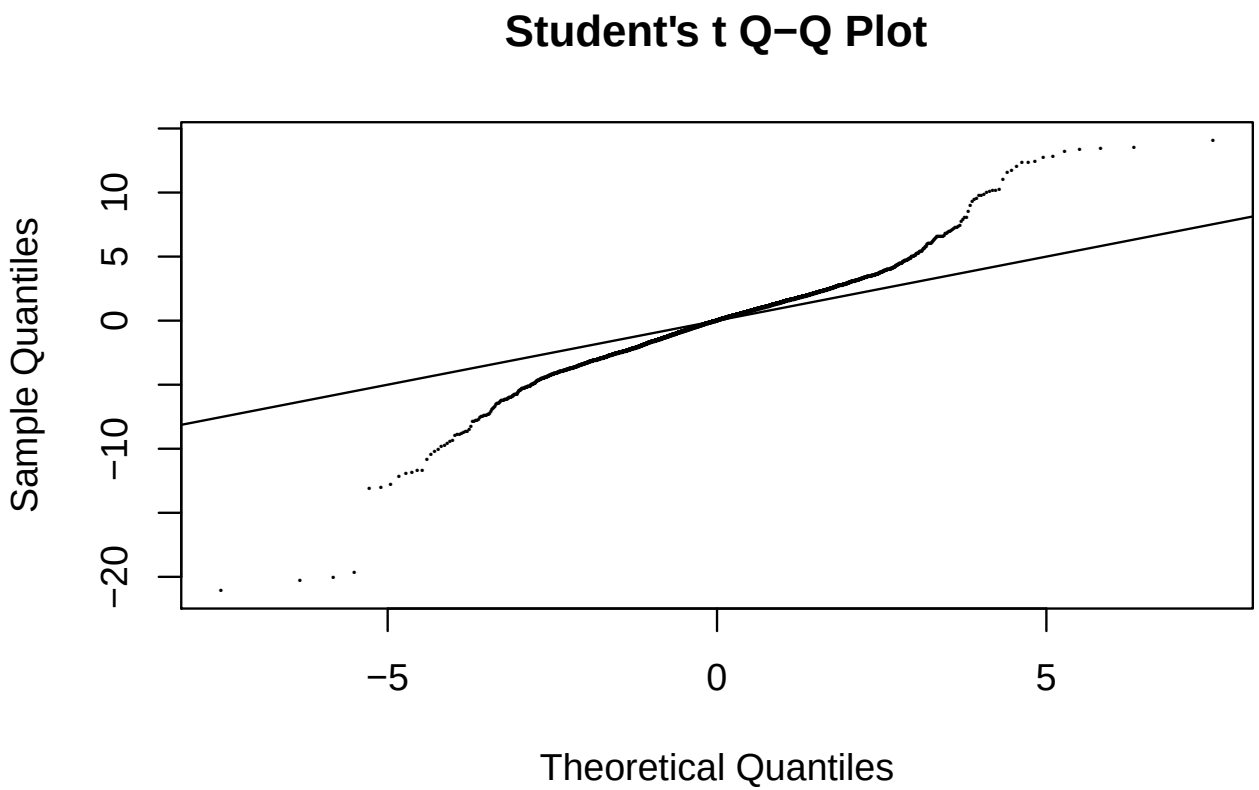
- According to the `limma` creators, its better to use t tests moderated with empirical Bayes; aka, using eBayes.
- With this we can find the differentially expressed genes.
- Actually, `eBayes` uses the empirical Bayes estimation to minimize the standar errors towars a common value.

  ```
  > fit <- eBayes(fit)
  ```
- Next, lets make a QQ plot to check if we have differentially expressed genes. Lets use qqt instead of qq because we want to compare against the t distribution quantiles and not versus a normal dist.

eBayes

```
> qqt(fit$t, df = fit$df.prior +
+     fit$df.residual, pch = 16,
+     cex = 0.2)
> abline(0, 1)
```

eBayes


Student's t Q−Q Plot
Sample Quantiles
−20
−10
0
5
10
−5
0
5
Theoretical Quantiles

topTable

- We have a lot of differentially expressed genes! :)
- To find which they are, we use the function topTable. An important argument is `adjust.method`, because with it we specify how we want to correct our p values.
- For example, with the following code you can look at the top 30 DEGs adjusting the p values by FDR.

```
> topTable(fit, number = 30, adjust = "BH")
```

- I'll show you one:

```
> topTable(fit, number = 1, adjust = "BH")
```

topTable

```
      Block Row Column      ID Name
3721      8   2      1 control BMP2
          logFC  AveExpr         t
3721 -2.205288 12.10451 -21.06952
           P.Value    adj.P.Val       B
3721 1.028468e-07 0.0003572816 7.96075
```

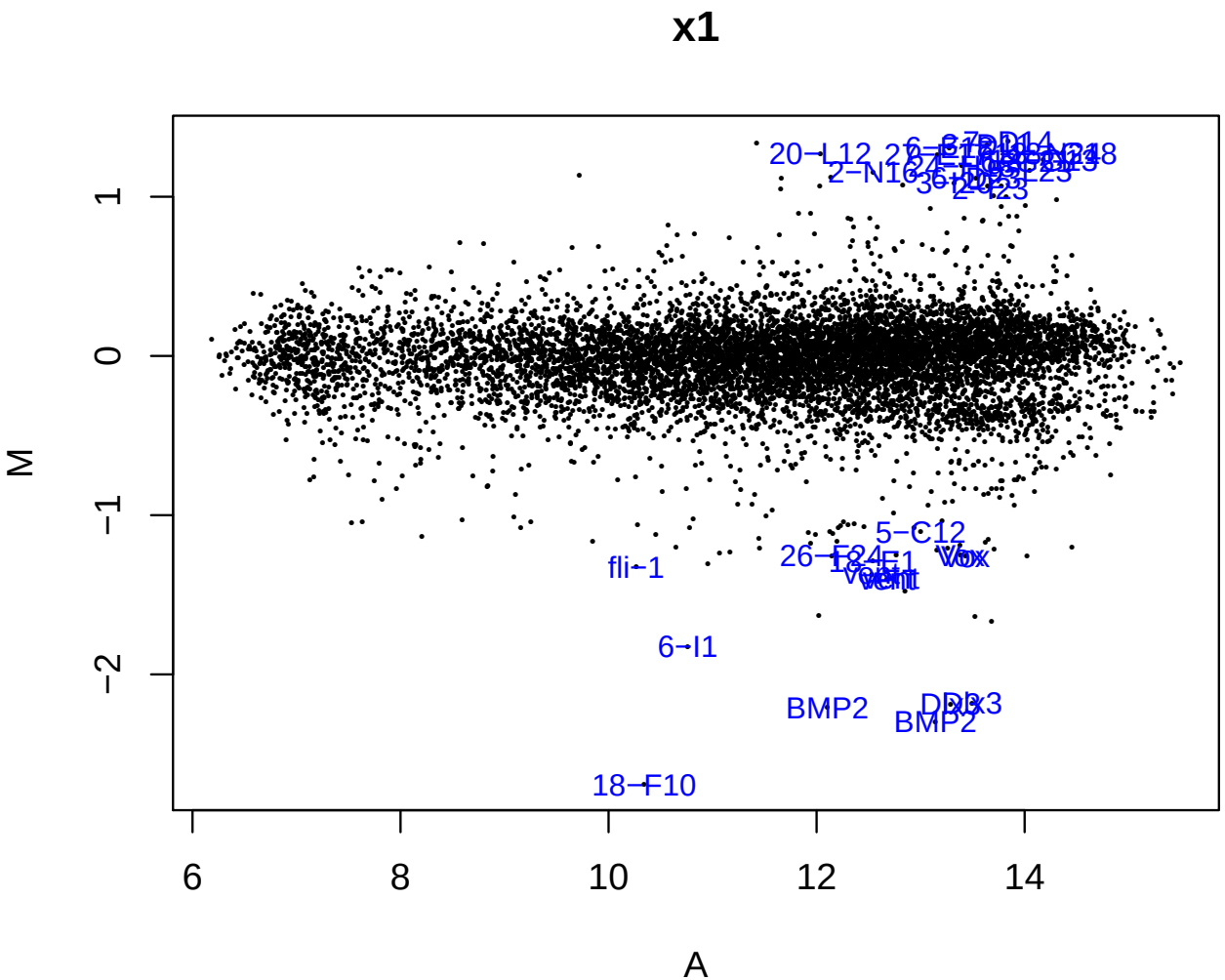
- As it was expected, our gene with the largest difference is BMP2. Remember that its knocked-out on the Swirl line.
- We can look at the original p values, the adjusted ones, the *log odds* from the empirtcal Bayes statistics, ...

Finishing with limma

- To finish our limma practical session, lets mark the top 30 genes in our previous plotMA plot.
- We'll use order and text from basic R to do this.

```
> plotMA(fit)
> ord <- order(fit$lods, decreasing = TRUE)
> top30 <- ord[1:30]
> text(fit$Amean[top30], fit$coef[top30],
+      labels = fit$genes[top30, "Name"],
+      cex = 0.8, col = "blue")
```

Finishing with `limma`



Homework

- With a data set of your choice (but new!) that has two variables, make a plot with the linear regression.
- Calculate the Spearman and Pearson correlation coefficients.
- Add your own conclusions.

SessionInfo

```
> sessionInfo()

R version 2.10.0 Under development (unstable) (2009-07-21 r48968)
i386-pc-mingw32

locale:
[1] LC_COLLATE=English_United States.1252
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] stats     graphics  grDevices
[4] utils     datasets  methods
[7] base

other attached packages:
[1] limma_2.19.2  UsingR_0.1-12
```