# Microarray Analysis

## Contents

Most expression studies try to identify the subset of genes that are differentially expressed between two or more conditions (differential gene expression (DGE) analysis).

# 1 Import data

In this example, we will use Affymetrix GeneChip bovine microarrays. It consists of five control slides and five treatment slides.

```
# Import data
library(affy)
filenames = c(paste("ctrl", 1:5, ".CEL", sep = ""),
              paste("treat", 1:5, ".CEL", sep = ""))
Names = c(paste("C", 1:5, sep = ""), paste("T", 1:5, sep = ""))
slides = ReadAffy(filenames = paste("Data/", filenames, sep = ""),
                  sampleNames = Names)
print(slides)
```

```
## AffyBatch object
## size of arrays=732x732 features (21 kb)
## cdf=Bovine (24128 affyids)
## number of samples=10
## number of genes=24128
## annotation=bovine
## notes=
```

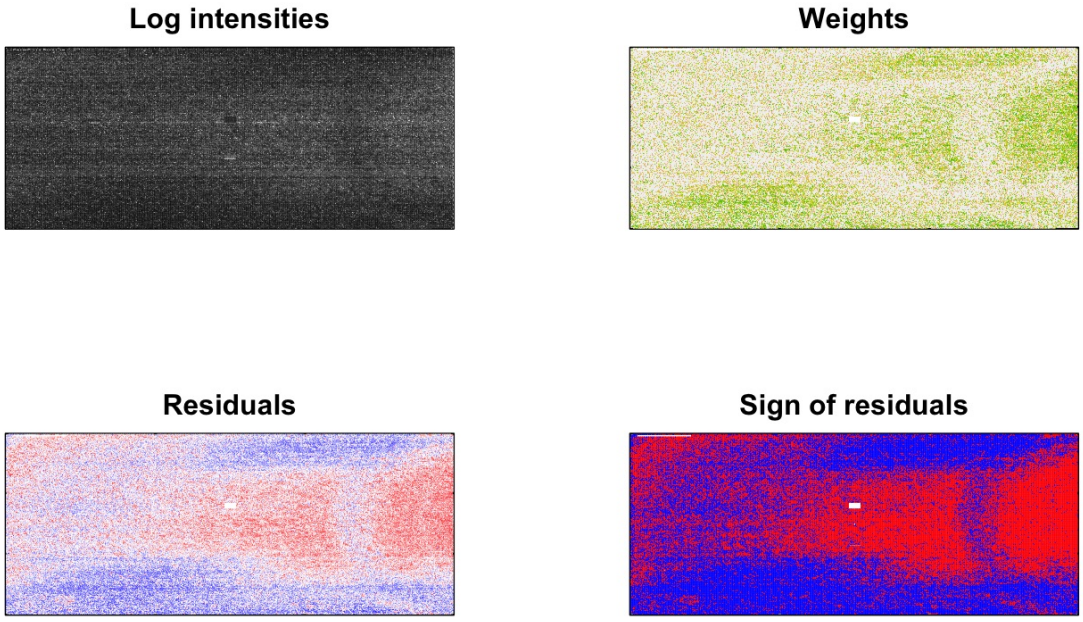
# 2 Quality control (QC)

## 2.1 Spatial effects

```
# Image of log intensities for the first slide
library(affyPLM)
PLM = fitPLM(slides)
```

```
par(mfrow = c(2, 2))
image(slides[, 1], main = "Log intensities")
image(
  PLM,
  type = "weights",
  which = 1,
  xlab = XLabel,
  main = "Weights"
)
image(
  PLM,
  type = "resids",
  which = 1,
  xlab = XLabel,
  main = "Residuals"
)
image(
  PLM,
  type = "sign.resids",
  which = 1,
  xlab = XLabel,
```

```
  main = "Sign of residuals"
)
```

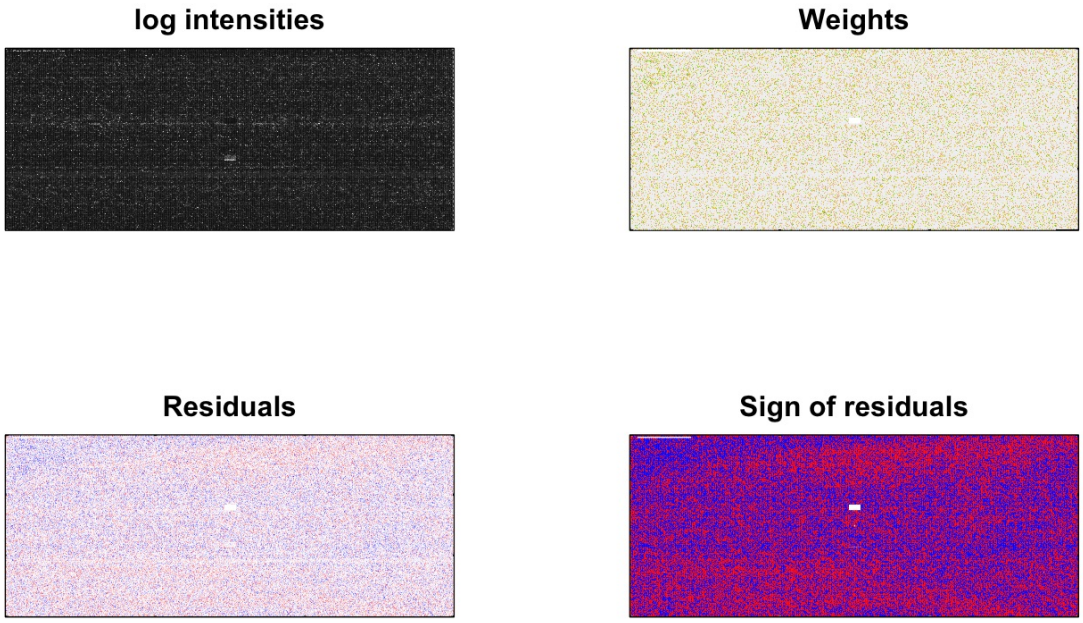



This slide is of bad quality due to its spatial effect. An example of a good slide is the following:

```
# Image of log intensities for the fifth slide
library(affyPLM)
PLM = fitPLM(slides)
```

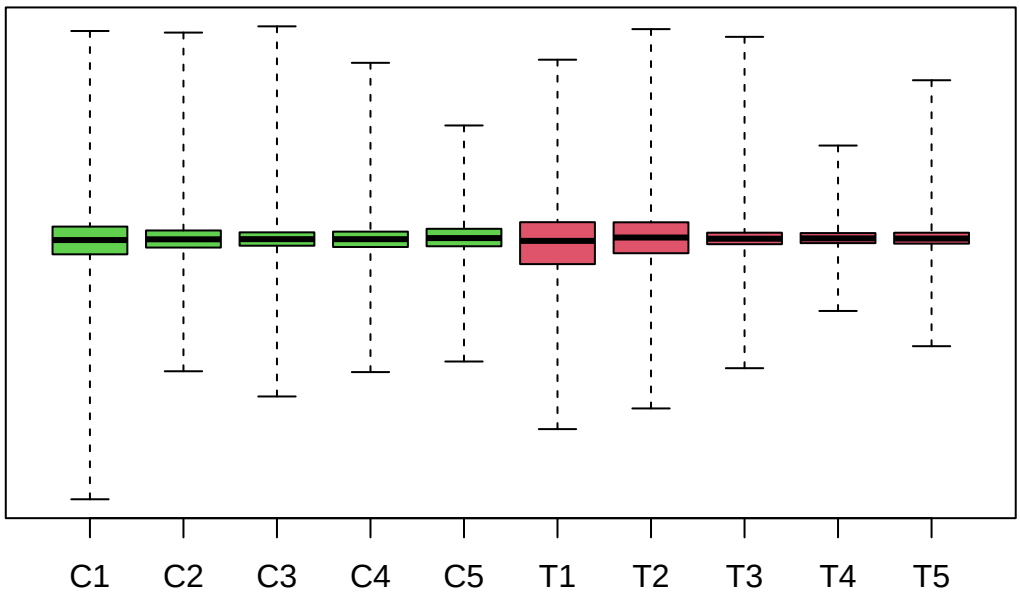
```
par(mfrow = c(2, 2))
image(slides[, 5], main = "log intensities")
image(
  PLM,
  type = "weights",
  which = 5,
  xlab = XLabel,
  main = "Weights"
)
image(
  PLM,
  type = "resids",
  which = 5,
  xlab = XLabel,
  main = "Residuals"
)
image(
  PLM,
  type = "sign.resids",
  which = 5,
```

```
  xlab = XLabel,
  main = "Sign of residuals"
)
```




We can also use relative log expression as another QC measure:

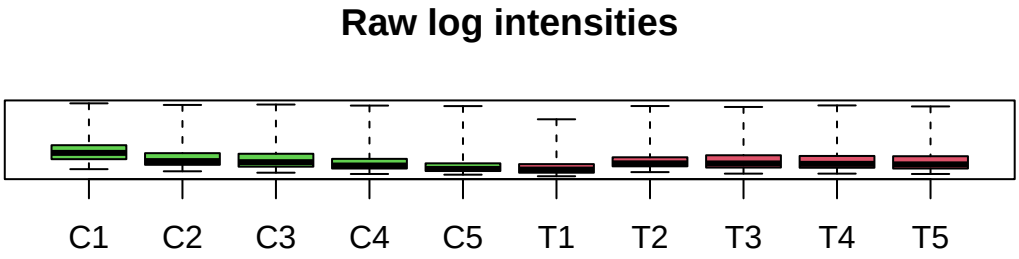
```
treatcol = c(3, 3, 3, 3, 3, 2, 2, 2, 2, 2)
Mbox(PLM,
     col = treatcol,
     main = "Relative log expression",
     yaxt = 'n',
     show.names = TRUE)
```

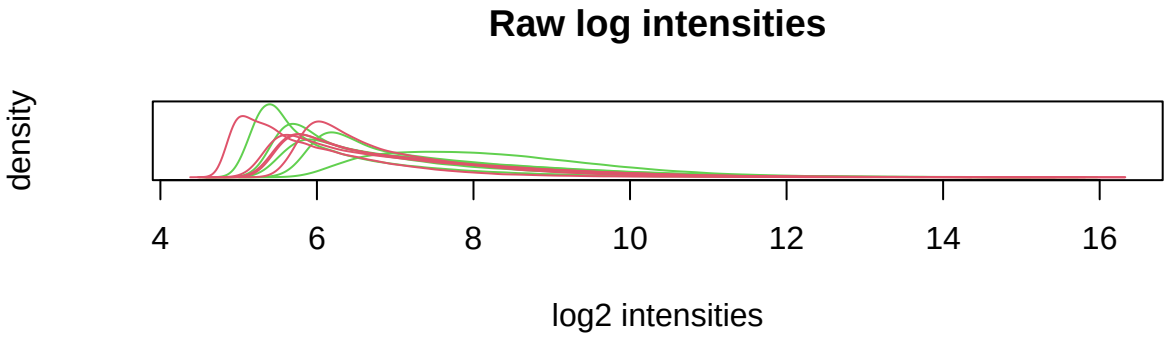



We would hope the medians are close to zero and the spread across arrays is similar. Slides C1, T1, and T2 look somewhat worrisome.

We could also make boxplots or histograms of the raw log intensities:

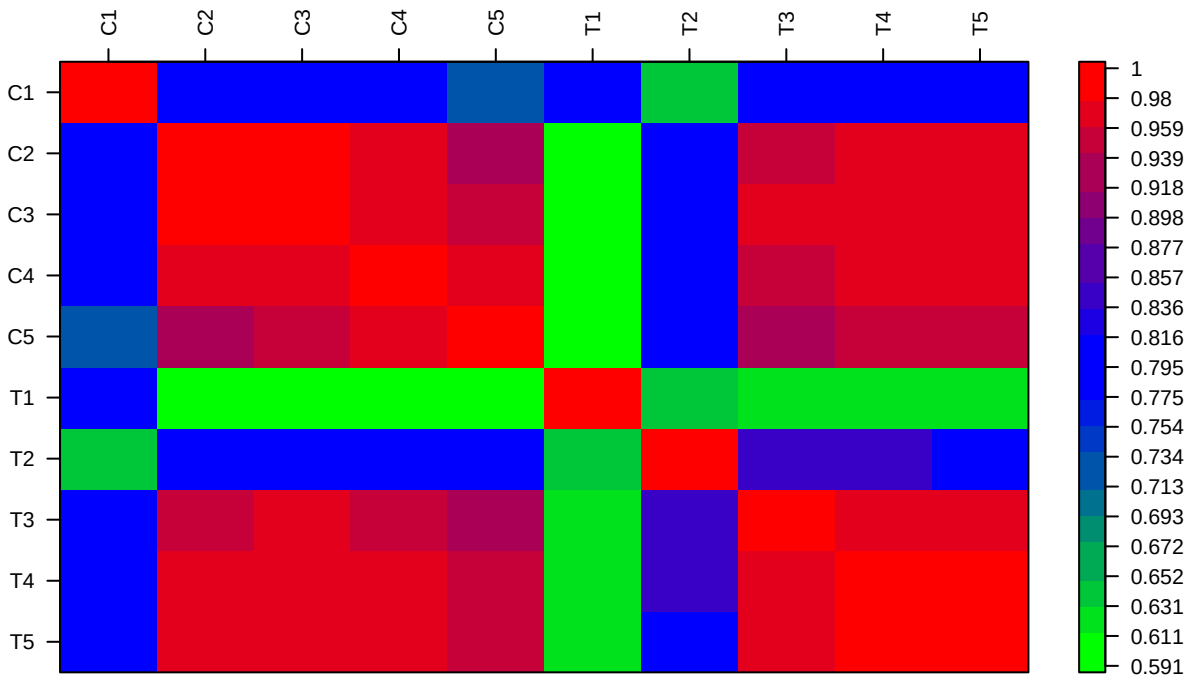
```
par(mfrow = c(2, 1))
boxplot(slides,
        col = treatcol,
        main = "Raw log intensities",
        yaxt = 'n',
        show.names = TRUE)
hist(
  slides,
  col = treatcol,
  lty = 1,
  xlab = "log2 intensities",
  yaxt = 'n',
  main = "Raw log intensities"
)
```




It is also useful to plot the correlations between slides:

```
library(ABarray)
Cor = pm(slides)
Cor = cor(Cor)
matrixPlot(Cor,
           nrgcols = 21,
           k = 21,
           title = "Perfect match probes (PM)")
```




Overall, it seems that slides C1 and T1 could be excluded due to low quality.

## 3 Preprocessing

Steps required include:

- Background correction to remove intensity measures that are not due to the target.
- Normalization which is necessary for across array comparisons.
- A summarization step which is more specific to Affymetrix GeneChips since these are unique in the use a set of short oligos to target a transcript. Various summarization algorithms have been proposed. The main ones are: MAS 5.0, RMA, GCRMA, PLIER, VSN, and MBEI. RMA is the most widely used.

RMA, GCRMA, PLIER, and VSN already return intensities in log 2 scale. MAS does not.

```
# MAS
library(affy)
MAS = mas5(slides, sc = 200)
```

```
## background correction: mas
## PM/MM correction : mas
## expression values: mas
## background correcting...done.
## 24128 ids to be processed
## |                    |
## |####################|
```

```
MAS = exprs(MAS)
MAS = log2(MAS)

# MASCalls
MASCalls = mas5calls(slides)
```

```
## Getting probe level data...
## Computing p-values
## Making P/M/A Calls
```

```
MASCalls = exprs(MASCalls)

# RMA
RMA = rma(slides)
```

```
## Background correcting
## Normalizing
## Calculating Expression
```

```
RMA = exprs(RMA)

# GCRMA
GCRMA = gcrma(slides)
```

```
## Adjusting for optical effect.........Done.
## Computing affinities.Done.
## Adjusting for non-specific binding.........Done.
## Normalizing
## Calculating Expression
```

```
GCRMA = exprs(GCRMA)

# PLIER
library(plier)
PLIER = justPlier(slides, normalize = TRUE)
```

```
## Quantile normalizing...Done.
```

```
PLIER = exprs(PLIER)

# VSN
library(vsn)
VSN = vsnrma(slides)
```

```
## Calculating Expression
```

```
VSN = exprs(VSN)

# MBEI
MBEI = expresso(
  slides,
  normalize.method = "invariantset",
  bg.correct = FALSE,
  pmcorrect.method = "pmonly",
  summary.method = "liwong"
)
```

```
## normalization: invariantset
## PM/MM correction : pmonly
## expression values: liwong
## normalizing...done.
## 24128 ids to be processed
## |                    |
## |####################|
```

```
MBEI = exprs(MBEI)
MBEI = log2(MBEI)
```

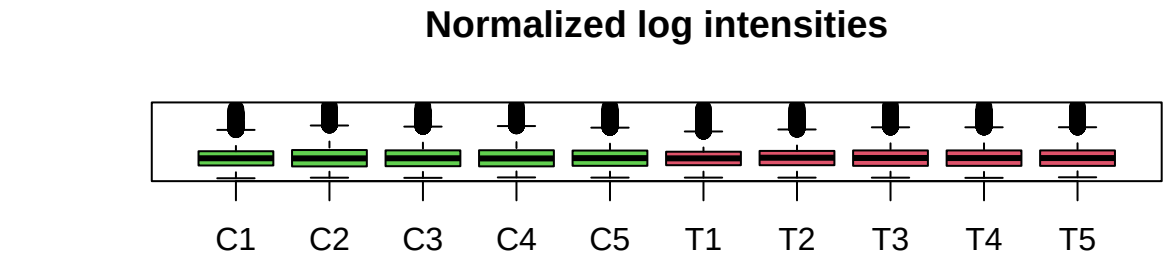
The **MASCalls** function returns a matrix with flag calls for the expression of each probe in each sample. The flags are P—present, M—marginal, and A—absent. It's best to remove all probes that are flagged as M or A in all arrays.

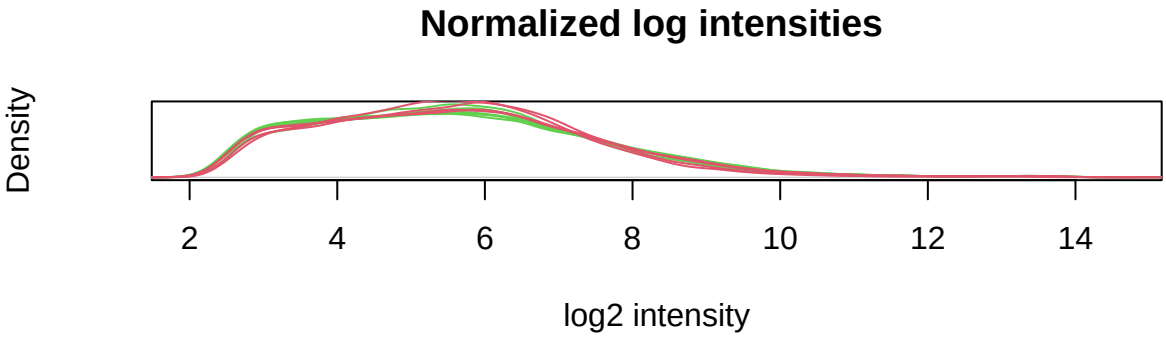
```
print(MASCalls[1:5, 1:3])
```

```
##                C1  C2  C3
## AFFX-BioB-3_at "M" "P" "P"
## AFFX-BioB-5_at "P" "P" "P"
## AFFX-BioB-M_at "P" "P" "P"
## AFFX-BioC-3_at "P" "P" "P"
## AFFX-BioC-5_at "P" "P" "P"
```

Now let's recheck the data for RMA:

```
par(mfrow = c(2, 1))
boxplot(RMA,
        col = treatcol,
        main = "Normalized log intensities",
        yaxt = 'n',
        show.names = TRUE)
plot(
  density(RMA[, 1]),
  col = treatcol[1],
  main = "Normalized log intensities ",
  yaxt = 'n',
  xlim = c(min(RMA), max(RMA)),
  xlab = "log2 intensity"
)
for (j in 2:length(treatcol)) {
  lines(density(RMA[, j]), col = treatcol[j])
}
```



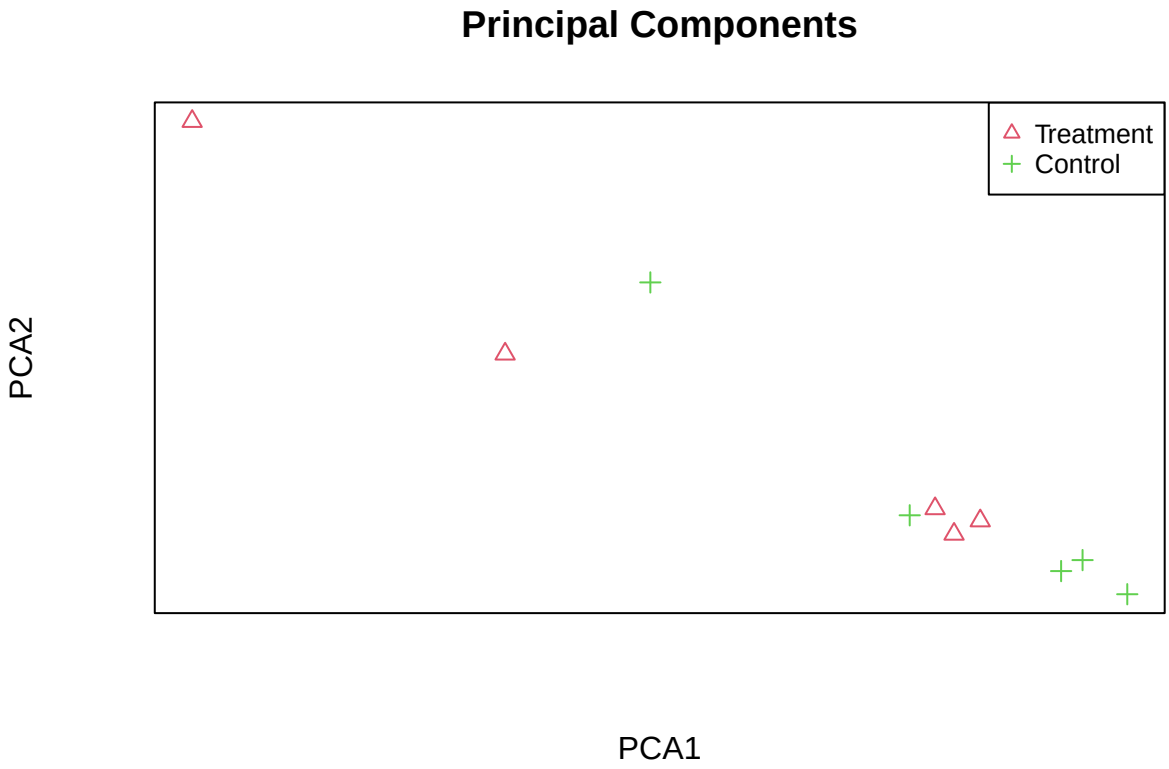

Now let's filter out the control probes and the probes that were flagged as A or M:

```
# Identify the control probes (Affymetrix uses AFFY in the probe names)
index1 = grep("AFFX", row.names(RMA), ignore.case = TRUE)

# Count the number of P calls for each probe
Pcounts = apply(MASCalls, 1, function(x) {
  length(which(x == "P"))
})

# Get the indices of those with 4 or less Ps per probe (6 will be A or M)
index2 = which(Pcounts <= 4)

# Filter them out
fRMA = RMA[-unique(c(index1, index2)), ]
```

Another interesting graph is PCA of the data. We would expect that the samples from the same group would be more similar to each other and cluster together while the distances between groups would be greater. If you notice that the samples group into, e.g., dates of running the hybridization or slide batches (could fit these effects as a color code), then you have a concern. Slides that are very far apart from others (irrespective of group) also should be looked at more closely.

```
# PCA
PCA = princomp(RMA)
PCA = loadings(PCA)[, 1:2]
plot(
  PCA,
  col = treatcol,
  main = "Principal Components",
  pch = treatcol,
  xlab = "PCA1",
```

```
  ylab = "PCA2",
  xaxt = 'n',
  yaxt = 'n',
  labels = FALSE
)
legend(
  "topright",
  c("Treatment", "Control"),
  pch = c(2, 3),
  col = c(2, 3),
  cex = 0.8
)
```




As we see, three slides are quite different from the others. We can find these slides using:

```
which(PCA[, 1] < 0.31)
```

```
## C1 T1 T2
##  1  6  7
```

# 4 Analysis of differential expression (DE)

There are many ways of testing for DE ranging from simple mean fold change differences between treatments to mixed models with various effects. We will use linear model.

We need three things:

- Our data
- A design matrix: simply states which arrays belong to which treatment
- A contrasts matrix: defines which treatment combinations we want to test

```r
# Design matrix
Design = cbind(c(rep(1, 5), rep(0, 5)), c(rep(0, 5), rep(1, 5)))
colnames(Design) = c("ctrl", "treat")
print(Design)
```

```
##       ctrl treat
##  [1,]    1     0
##  [2,]    1     0
##  [3,]    1     0
##  [4,]    1     0
##  [5,]    1     0
##  [6,]    0     1
##  [7,]    0     1
##  [8,]    0     1
##  [9,]    0     1
## [10,]    0     1
```

```r
# Contrasts matrix
Contrasts = matrix(c(1, -1), byrow = F)
colnames(Contrasts) = "ctrl-treat"
rownames(Contrasts) = c("ctrl", "treat")
print(Contrasts)
```

```
##       ctrl-treat
## ctrl           1
## treat         -1
```

Let's fit the model:

```r
library(limma)
Fit = lmFit(fRMA, Design)
Fitc = contrasts.fit(Fit, Contrasts)
## to test for DE we use an empirical Bayes shrinkage of SEs
Fitb = eBayes(Fitc)

# Coefficients from model fitted to data and design matrix
head(Fit$coefficients)
```

```
##                       ctrl    treat
## Bt.1.1.S1_at     8.310481 8.323970
## Bt.10.2.S1_a_at  7.478839 7.488885
## Bt.100.1.S1_at   6.716818 6.464648
## Bt.10005.1.S1_at 6.664079 6.308243
## Bt.10006.1.A1_at 3.947718 4.076727
## Bt.10006.2.S1_at 6.711188 6.376521
```

```r
# Next we tested our contrast
head(Fitc$coefficients)
```

```
##                   ctrl-treat
## Bt.1.1.S1_at     -0.01348861
## Bt.10.2.S1_a_at  -0.01004659
## Bt.100.1.S1_at    0.25216948
## Bt.10005.1.S1_at  0.35583584
## Bt.10006.1.A1_at -0.12900870
## Bt.10006.2.S1_at  0.33466694
```

In our case we know that the contrast is control– treatment, hence any positive difference in expression means that the control is more expressed than the treatment and any negative difference means the treatment is more expressed than the control.

```
# Retrieve the results from the moderated t-test and corresponding p-values
## t score
head(Fitb$t)
```

```
##                   ctrl-treat
## Bt.1.1.S1_at     -0.07052703
## Bt.10.2.S1_a_at  -0.06823025
## Bt.100.1.S1_at    1.30687042
## Bt.10005.1.S1_at  1.95689931
## Bt.10006.1.A1_at -0.67790699
## Bt.10006.2.S1_at  1.42722551
```

```
## p-value
head(Fitb$p.value)
```

```
##                   ctrl-treat
## Bt.1.1.S1_at     0.94475870
## Bt.10.2.S1_a_at  0.94655465
## Bt.100.1.S1_at   0.21204381
## Bt.10005.1.S1_at 0.07033442
## Bt.10006.1.A1_at 0.50873645
## Bt.10006.2.S1_at 0.17515314
```

If there were more contrasts we could also retrieve the F-statistics and their p-values.

After all, which are the differentially expressed probes?

```
res = data.frame(
  FoldChange = Fitb$coefficients,
  p.value = Fitb$p.value,
  Amean = Fitb$Amean
)
names(res) = c("FoldChange", "p.value", "Avalue")
res = res[order(res$p.value), ]
DE = res[res$p.value < 0.01, ]
head(DE)
```

```
##                    FoldChange      p.value   Avalue
## Bt.28515.1.A1_at    1.3280633 8.867785e-05 6.019431
## Bt.3289.1.S1_at     1.0095877 1.043538e-04 5.679625
## Bt.22867.2.A1_at    0.9772835 4.177071e-04 7.796835
## Bt.4751.1.S1_a_at   1.0429455 8.267074e-04 9.469517
## Bt.25089.2.A1_at    0.8267011 1.027947e-03 6.323292
## Bt.23123.1.S1_at    0.9263304 1.250711e-03 6.575189
```

For the case of multiple testing, Bonferroni correction is probably too stringent for most cases. A reasonable compromise option is the Benjamin and Hochberg method:

```
adjusted = p.adjust(res$p.value, method = "BH")
min(adjusted)
```

```
## [1] 0.4667224
```

Not a single probe survived multiple testing correction. Given the bad quality of three slides and the fact that we did not have a real contrast, it's probably a good thing.