A. RNAseq: differential expression analysis

DESeq2

We used the DESeq2 program for differential expression analysis.

In [1]:
```
library("DESeq2")
library(scales)
options(repr.plot.width=4, repr.plot.height=3)
```

```
Loading required package: S4Vectors
Loading required package: stats4
Loading required package: BiocGenerics
Loading required package: parallel

Attaching package: ‘BiocGenerics’

The following objects are masked from ‘package:parallel’:

    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
    clusterExport, clusterMap, parApply, parCapply, parLapply,
    parLapplyLB, parRapply, parSapply, parSapplyLB

The following object is masked from ‘package:stats’:

    xtabs

The following objects are masked from ‘package:base’:

    anyDuplicated, append, as.data.frame, as.vector, cbind, colnames,
    do.call, duplicated, eval, evalq, Filter, Find, get, intersect,
    is.unsorted, lapply, Map, mapply, match, mget, order, paste, pmax,
    pmax.int, pmin, pmin.int, Position, rank, rbind, Reduce, rep.int,
    rownames, sapply, setdiff, sort, table, tapply, union, unique,
    unlist, unsplit

Creating a generic function for ‘nchar’ from package ‘base’ in package ‘S4Vecto
rs’
Loading required package: IRanges
Loading required package: GenomicRanges
Loading required package: GenomeInfoDb
Loading required package: Rcpp
Loading required package: RcppArmadillo
Warning message:
: replacing previous import ‘S4Vectors::Position’ by ‘ggplot2::Position’ when l
oading ‘DESeq2’
```

We ensured we're in the correct directory.

In [2]:
```
setwd(path.expand("/home/luna.kuleuven.be/r0307152/data/assignment/~/data/ncbi2/s
```

We defined the file containing the raw gene counts.

In [3]:
```
matrixFile = path.expand("/home/luna.kuleuven.be/r0307152/data/assignment/SRP0263
```

In [4]:
```
countData <- as.matrix(read.csv(matrixFile, sep="\t", row.names="Geneid"))
head(countData)
```

	SRR924146.bam	SRR924528.bam	SRR924529.bam	SRR924530.bam	SRR924
MIR1302-11	0	0	0	0	0
FAM138A	0	0	0	0	0
OR4F5	0	0	0	0	0
RP11-34P13.7	6	3	4	16	2
RP11-34P13.8	0	0	0	0	0
AL627309.1	165	158	194	155	100

We made the column names more informative.

In [5]:
```
colnames(countData) <- c("0h-1", "0h-2", "2h-1", "2h-2", "4h-1", "4h-2", "6h-1",
head(countData)
```

	0h-1	0h-2	2h-1	2h-2	4h-1	4h-2	6h-1	6h-2	24h-1	24h-2	72h-1	72h-2
MIR1302-11	0	0	0	0	0	0	0	0	0	0	0	0
FAM138A	0	0	0	0	0	0	0	0	0	0	0	0
OR4F5	0	0	0	0	0	0	0	0	0	0	0	0
RP11-34P13.7	6	3	4	16	2	4	4	2	2	0	0	0
RP11-34P13.8	0	0	0	0	0	0	0	0	0	0	0	0
AL627309.1	165	158	194	155	100	154	95	68	26	40	83	95

We defined the experimental conditions: all samples were stimulated with anti-CD3 and anti-CD28. RNA was isolated at a total of 6 time points, with two biological replicates per time point.

We ensured that the order is the same as the column order of the countData and that the order of the levels is correct.

In [6]:
```
condition <- factor(c("0h", "0h", "2h", "2h", "4h", "4h", "6h", "6h", "24h", "24h
print(condition)
```

```
 [1] 0h  0h  2h  2h  4h  4h  6h  6h  24h 24h 72h 72h
Levels: 0h 24h 2h 4h 6h 72h
```

We generated a dataframe with names and conditions and ensured that the order is correct.

In [7]:
```
colData <- data.frame(sampleName = colnames(countData),
                      condition = condition)
colData
```

sampleName	condition
0h-1	0h
0h-2	0h
2h-1	2h
2h-2	2h
4h-1	4h
4h-2	4h
6h-1	6h
6h-2	6h
24h-1	24h
24h-2	24h
72h-1	72h
72h-2	72h

We created a DESeq DESeqDataSet object for subsequent analysis. We ensured the column names are correct.

In [8]:
```
dds <- DESeqDataSetFromMatrix(countData = countData,
                              colData = colData,
                              design = ~ condition)

colnames(dds) <- colnames(countData)
```

We used the dds command to see some information.

In [9]:
```
dds
```

```
class: DESeqDataSet
dim: 41863 12
exptData(0):
assays(1): counts
rownames(41863): MIR1302-11 FAM138A ... AC007965.1 RNU6-1314P
rowRanges metadata column names(0):
colnames(12): 0h-1 0h-2 ... 72h-1 72h-2
colData names(2): sampleName condition
```

We confirmed that the order is still correct.

In [10]:
```
dds$condition
```

0h 0h 2h 2h 4h 4h 6h 6h 24h 24h 72h 72h

Running the differential expression analysis

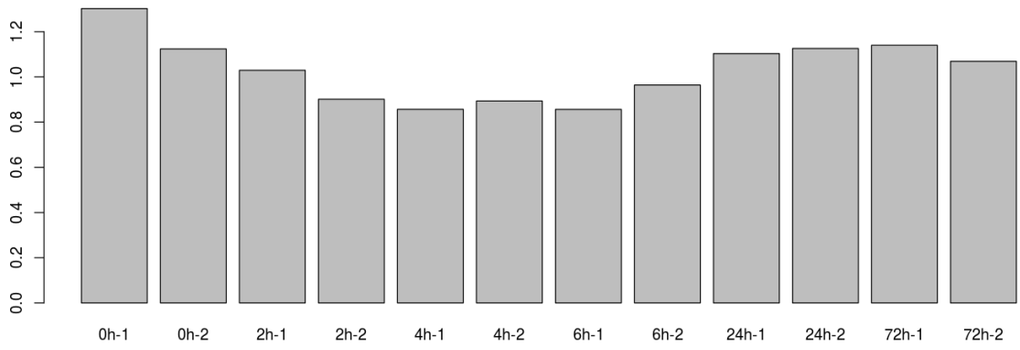
In [11]:
```
dds <- DESeq(dds)
```

```
estimating size factors
estimating dispersions
gene-wise dispersion estimates
mean-dispersion relationship
final dispersion estimates
fitting model and testing
```

Quality control

DESeq2 estimates a factor to correct for the library size.

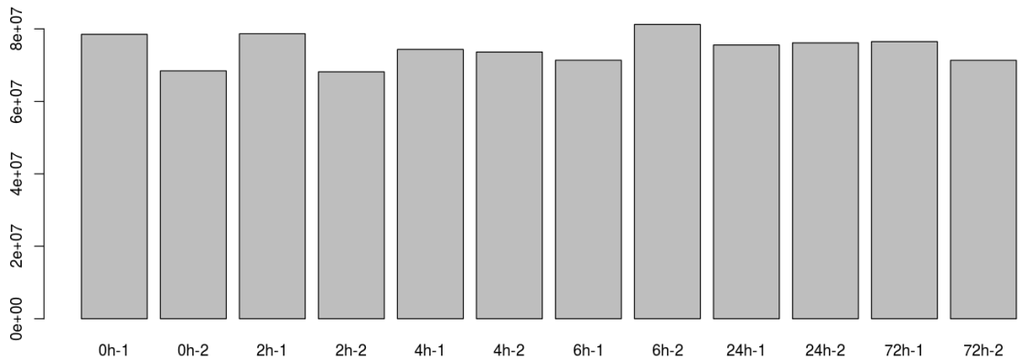
In [12]:
```
options(repr.plot.width=12, repr.plot.height=5)
barplot(sizeFactors(dds))
```



Different plots

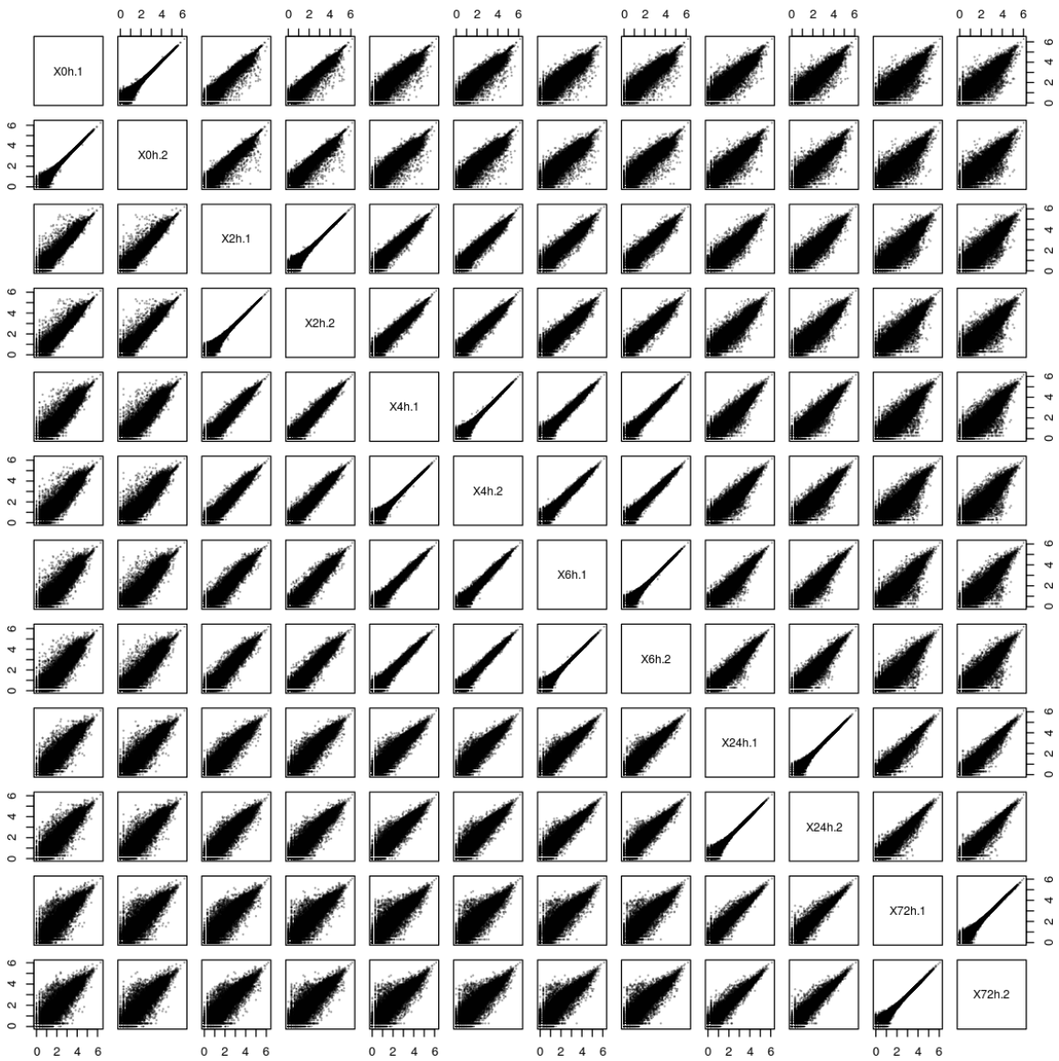
A barplot

In [13]:
```
barplot(colSums(countData))
```



A scatterplot of sample versus sample

In [14]:
```
 options(repr.plot.width=10, repr.plot.height=10)
plot(data.frame(log10(counts(dds))), cex=0.1, col=alpha("black", 0.4))
```

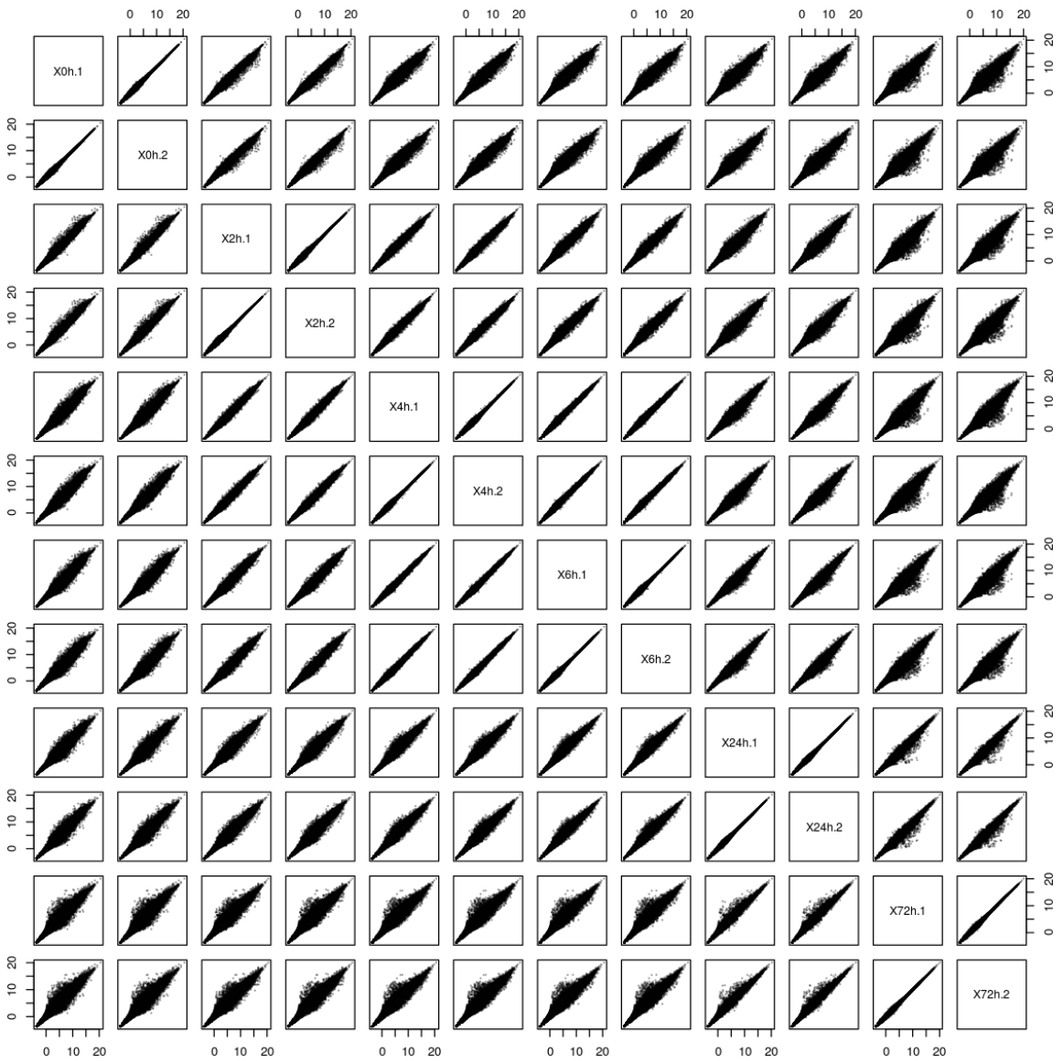


We can see that the replicates are more similar to each other and that the low expressed genes produce a lot of noise.

We therefore use a function of DESeq2 that minimalizes variation, normalizes for library size and log-transforms the data.

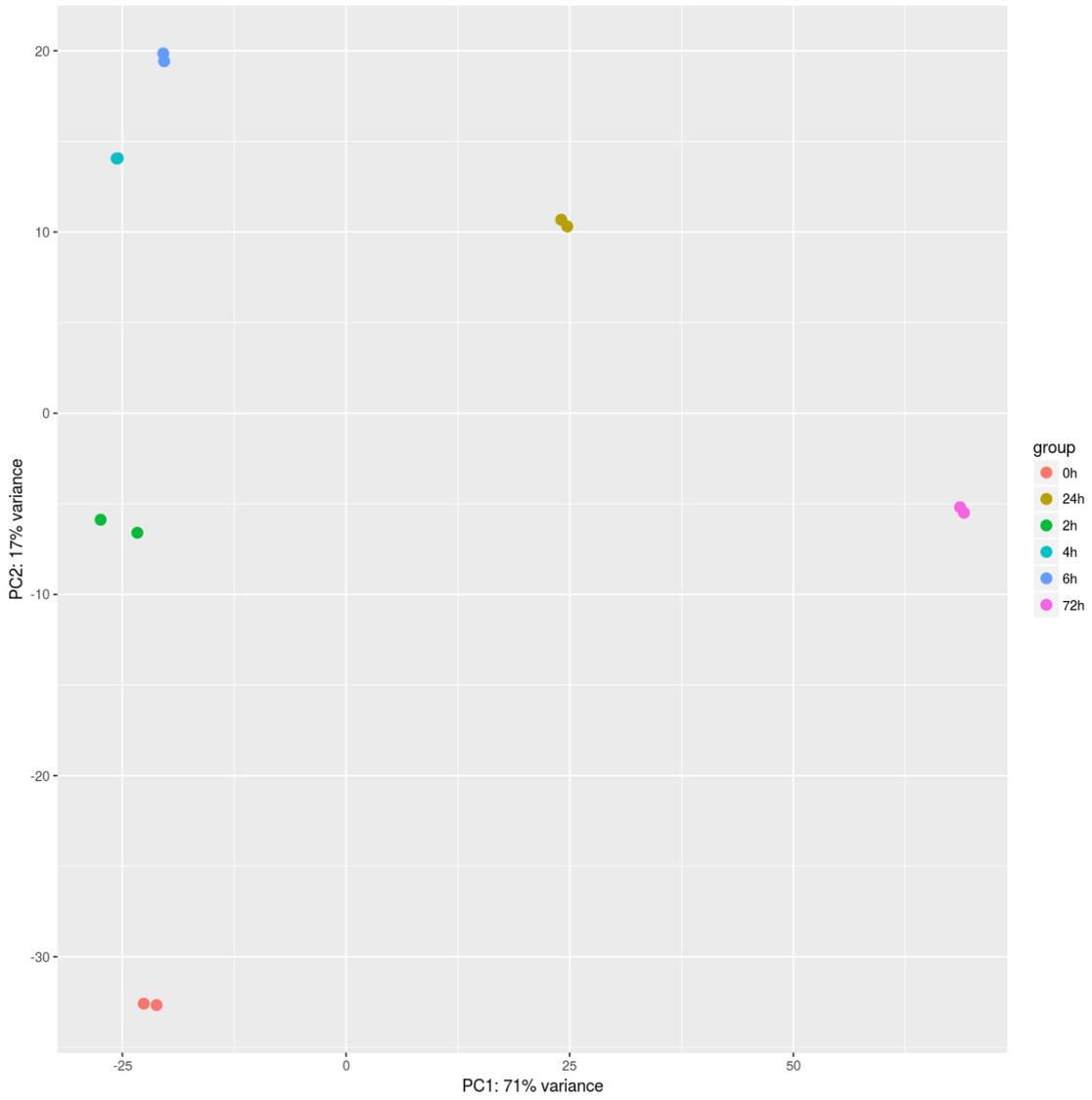
In [15]:
```
rld <- rlogTransformation(dds, blind=TRUE)
```

In [16]:
```
options(repr.plot.width=10, repr.plot.height=10)
plot(data.frame(assay(rld)), cex=0.1, col=alpha("black", 0.4))
```



A PCA plot

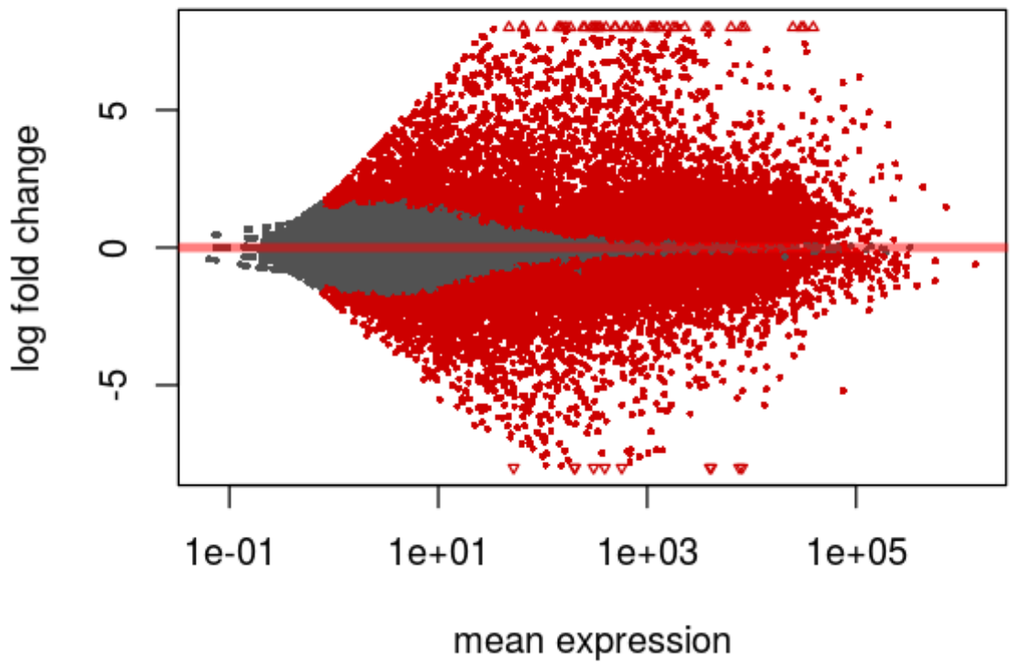
In [17]:
```
plotPCA(rld, intgroup=c("condition"))
```



This plot is a way of displaying intersample variance. It shows that the biological replicates here cluster nicely together. The first two principle components explain for 88% of variance.

A MA plot

In [18]:
```
options(repr.plot.width=5, repr.plot.height=4)
plotMA(dds,ylim=c(-8,8))
```



The MA plot shows the mean expression versus the log fold change. Significantly differentially expressed genes are colored red.

Exporting the results

Comparison to 0h group

We extracted the results from DESeq analysis comparing **72h group** with 0h group.

In [19]:
```
res <- results(dds)
res <- res[order(res$padj),]
head(res)
```

```
log2 fold change (MAP): condition 72h vs 0h
Wald test p-value: condition 72h vs 0h
DataFrame with 6 rows and 6 columns
          baseMean log2FoldChange      lfcSE      stat    pvalue      padj
         <numeric>      <numeric>  <numeric> <numeric> <numeric> <numeric>
ACOT7     4235.094       4.576920 0.08053354  56.83247         0         0
TNFRSF9   9757.640       4.845202 0.10475215  46.25396         0         0
ENO1    238717.202       3.038870 0.05236099  58.03691         0         0
TNFRSF8   7990.467       8.581231 0.12082482  71.02208         0         0
MRTO4    10341.440       2.582855 0.05900467  43.77374         0         0
LDLRAP1   2853.626      -3.855108 0.08392241 -45.93657         0         0
```

Selection criteria for differential expression required genes to have fold-change greater than 2.0, FDR-adjusted p less than 0.05.

Number of genes that are differentially upregulated:

In [23]:
```
up_genes <- res[which((res$log2FoldChange) >  2) ,]
dim(up_genes)
```

```
2073  6
```

In [25]:
```
up.sign <- up_genes[(!is.na(up_genes$padj)) & (up_genes$padj < 0.05),]
dim(res.sign)
```

2073 6

Number of genes that are differentially downregulated:

In [29]:
```
down_genes <- res[which((res$log2FoldChange) < - 2 ),]
dim(down_genes)
```

1725 6

In [32]:
```
down.sign <- down_genes[(!is.na(down_genes$padj)) & (down_genes$padj < 0.05),]
dim(down.sign)
```

1725 6

In [21]:
```
write.table(res, "deseq-72h-vs-0h.results.tsv", sep="\t", col.names=NA, quote=FAL
```

We extracted the results from DESeq analysis comparing **24h group** with 0h group.

In [50]:
```
res <- results(dds, contrast=c("condition","24h","0h"))
head(res)
```

```
log2 fold change (MAP): condition 24h vs 0h
Wald test p-value: condition 24h vs 0h
DataFrame with 6 rows and 6 columns
               baseMean log2FoldChange     lfcSE      stat       pvalue
              <numeric>      <numeric> <numeric> <numeric>    <numeric>
MIR1302-11     0.000000             NA        NA        NA           NA
FAM138A        0.000000             NA        NA        NA           NA
OR4F5          0.000000             NA        NA        NA           NA
RP11-34P13.7   3.690667     -0.9995906 0.8979967 -1.113134 2.656509e-01
RP11-34P13.8   0.000000             NA        NA        NA           NA
AL627309.1   109.927627     -2.0942925 0.3222139 -6.499695 8.048284e-11
                     padj
                <numeric>
MIR1302-11             NA
FAM138A                NA
OR4F5                  NA
RP11-34P13.7 3.453696e-01
RP11-34P13.8           NA
AL627309.1   2.317100e-10
```

Number of genes that are differentially upregulated:

In [51]:
```
up_genes1 <- res[which((res$log2FoldChange) >  2) ,]
dim(up_genes1)
```

1386 6

In [55]:
```
up.sign <- up_genes1[(!is.na(up_genes1$padj)) & (up_genes1$padj < 0.05),]
dim(up.sign)
```

1386 6

Number of genes that are differentially downregulated:

In [37]:
```
down_genes1 <- res[which((res$log2FoldChange) < - 2 ),]
dim(down_genes1)
```

1908 6

In [56]:
```
down.sign1 <- down_genes[(!is.na(down_genes1$padj)) & (down_genes1$padj < 0.05),]
dim(down.sign1)
```

1903 6

In [23]:
```
write.table(res, "deseq-24h-vs-0h.results.tsv", sep="\t", col.names=NA, quote=FAL
```

We extracted results from DESeq analysis comparing **6h group** with 0h group.

In [44]:
```
res <- results(dds, contrast=c("condition","6h","0h"))
head(res)
```

```
log2 fold change (MAP): condition 6h vs 0h
Wald test p-value: condition 6h vs 0h
DataFrame with 6 rows and 6 columns
               baseMean log2FoldChange     lfcSE        stat     pvalue
              <numeric>      <numeric> <numeric>   <numeric>  <numeric>
MIR1302-11     0.000000             NA        NA          NA         NA
FAM138A        0.000000             NA        NA          NA         NA
OR4F5          0.000000             NA        NA          NA         NA
RP11-34P13.7   3.690667     -0.0853025 0.8940839 -0.09540772 0.92399099
RP11-34P13.8   0.000000             NA        NA          NA         NA
AL627309.1   109.927627     -0.5480306 0.2976275 -1.84133055 0.06557313
                     padj
                <numeric>
MIR1302-11           NA
FAM138A              NA
OR4F5                NA
RP11-34P13.7 0.95067416
RP11-34P13.8         NA
AL627309.1   0.09867431
```

Number of genes that are differentially upregulated:

In [45]:
```
up_genes2 <- res[which((res$log2FoldChange) >  2) ,]
dim(up_genes2)
```

```
1596  6
```

In [57]:
```
up.sign <- up_genes2[(!is.na(up_genes2$padj)) & (up_genes2$padj < 0.05),]
dim(up.sign)
```

```
1596  6
```

Number of genes that are differentially downregulated:

In [46]:
```
down_genes2 <- res[which((res$log2FoldChange) < - 2 ),]
dim(down_genes2)
```

```
2258  6
```

In [47]:
```
down_genes2 <- res[which((res$log2FoldChange) < - 2 ),]
dim(down_genes2)
```

```
2258  6
```

In [26]:
```
write.table(res, "deseq-6h-vs-0h.results.tsv", sep="\t", col.names=NA, quote=FALS
```

We extracted results from DESeq analysis comparing **4h group** with 0h group.

In [59]:
```
res <- results(dds, contrast=c("condition","4h","0h"))
head(res)
```

```
log2 fold change (MAP): condition 4h vs 0h
Wald test p-value: condition 4h vs 0h
DataFrame with 6 rows and 6 columns
               baseMean log2FoldChange     lfcSE       stat    pvalue      padj
              <numeric>      <numeric> <numeric>  <numeric> <numeric> <numeric>
MIR1302-11     0.000000             NA        NA         NA        NA        NA
FAM138A        0.000000             NA        NA         NA        NA        NA
OR4F5          0.000000             NA        NA         NA        NA        NA
RP11-34P13.7   3.690667     -0.0699330 0.8943480 -0.0781944 0.9376734 0.9624078
RP11-34P13.8   0.000000             NA        NA         NA        NA        NA
AL627309.1   109.927627      0.1113377 0.2910515  0.3825361 0.7020638 0.7737167
```

Number of genes that are differentially upregulated:

In [60]:
```
up_genes3 <- res[which((res$log2FoldChange) >  2) ,]
dim(up_genes3)
```

```
1491  6
```

In [61]:
```
up.sign <- up_genes3[(!is.na(up_genes3$padj)) & (up_genes3$padj < 0.05),]
dim(up.sign)
```

```
1490  6
```

Number of genes that are differentially downregulated:

In [64]:
```
down_genes <- res[which((res$log2FoldChange) < - 2 ),]
dim(down_genes)
```

```
1707  6
```

In [63]:
```
down_genes <- res[which((res$log2FoldChange) < - 2 ),]
dim(down_genes)
```

```
1707  6
```

In [29]:
```
write.table(res, "deseq-4h-vs-0h.results.tsv", sep="\t", col.names=NA, quote=FALS
```

We extracted results from DESeq analysis comparing **2h group** with 0h group.

In [65]:
```
res <- results(dds, contrast=c("condition","2h","0h"))
head(res)
```

```
log2 fold change (MAP): condition 2h vs 0h
Wald test p-value: condition 2h vs 0h
DataFrame with 6 rows and 6 columns
               baseMean log2FoldChange     lfcSE      stat    pvalue      padj
              <numeric>      <numeric> <numeric> <numeric> <numeric> <numeric>
MIR1302-11     0.000000             NA        NA        NA        NA        NA
FAM138A        0.000000             NA        NA        NA        NA        NA
OR4F5          0.000000             NA        NA        NA        NA        NA
RP11-34P13.7   3.690667      1.0715449 0.8863394  1.208956 0.2266799 0.3498564
RP11-34P13.8   0.000000             NA        NA        NA        NA        NA
AL627309.1   109.927627      0.4219418 0.2876122  1.467051 0.1423622 0.2383835
```

Number of genes that are differentially upregulated:

In [66]:
```
up_genes <- res[which((res$log2FoldChange) >  2) ,]
dim(up_genes)
```

```
968  6
```

In [67]:
```
up.sign <- up_genes[(!is.na(up_genes$padj)) & (up_genes$padj < 0.05),]
dim(up.sign)
```

```
965  6
```

Number of genes that are differentially downregulated:

In [68]:
```
down_genes <- res[which((res$log2FoldChange) < - 2 ),]
dim(down_genes)
```

```
341  6
```

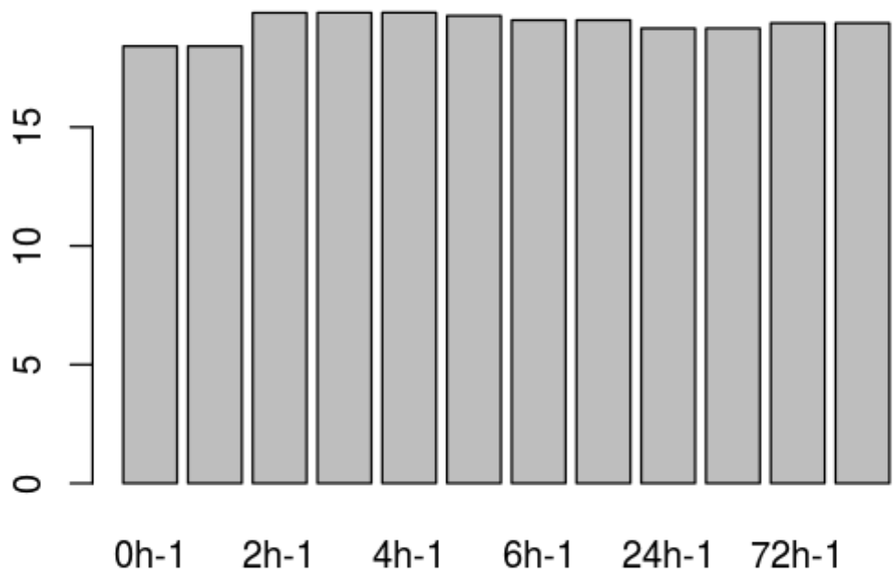
In [69]:
```
down_genes <- res[which((res$log2FoldChange) < - 2 ),]
dim(down_genes)
```

```
341  6
```

In [32]:
```
write.table(res, "deseq-2h-vs-0h.results.tsv", sep="\t", col.names=NA, quote=FALS
```

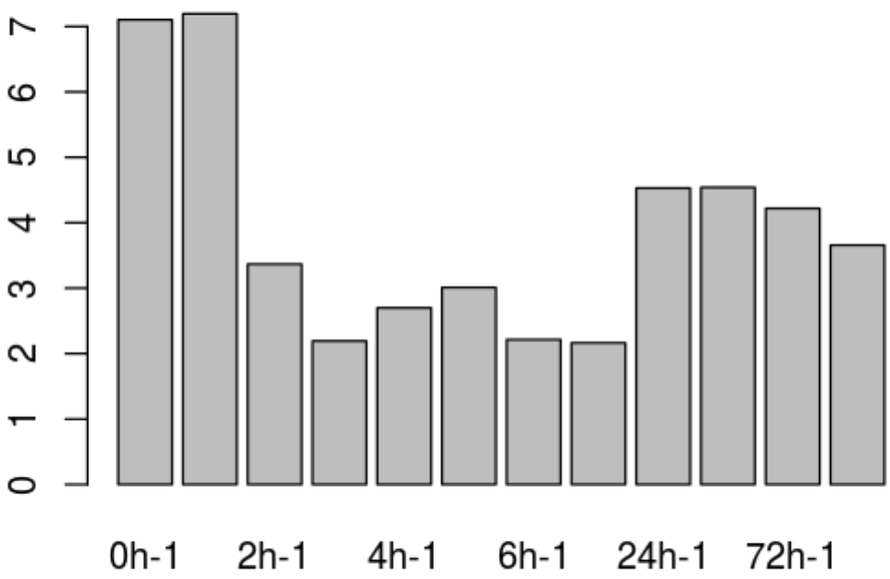
One example of a gene that was **not** differentially expressed at different timepoints after stimulation with anti-CD3 and anti-CD28 was *ACTB*.

In [33]:
```
barplot(assay(rld)["ACTB",])
```



One example of a gene that was differentially expressed at different time points after stimulation with anti-CD3 and anti-CD28 was *MYCL*.

In [34]:
```
barplot(assay(rld)["MYCL",])
```



B. Microarray analysis

Analysis of the Role of CD8+ T Cells in Severe Asthma

The discription of the study can be found in the main notebook section 2.

Libraries

The first step would be to load the required libraries. The data provided are Affymetrix microarray data, therefore we will load the affy package, and the limma package for the differentail expression analysis.

In [1]:
```
library(limma)
library(affy)
```

```
Loading required package: BiocGenerics
Loading required package: parallel

Attaching package: ‘BiocGenerics’

The following objects are masked from ‘package:parallel’:

    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
    clusterExport, clusterMap, parApply, parCapply, parLapply,
    parLapplyLB, parRapply, parSapply, parSapplyLB

The following object is masked from ‘package:limma’:

    plotMA

The following object is masked from ‘package:stats’:

    xtabs

The following objects are masked from ‘package:base’:

    anyDuplicated, append, as.data.frame, as.vector, cbind, colnames,
    do.call, duplicated, eval, evalq, Filter, Find, get, intersect,
    is.unsorted, lapply, Map, mapply, match, mget, order, paste, pmax,
    pmax.int, pmin, pmin.int, Position, rank, rbind, Reduce, rep.int,
    rownames, sapply, setdiff, sort, table, tapply, union, unique,
    unlist, unsplit

Loading required package: Biobase
Welcome to Bioconductor

    Vignettes contain introductory material; view with
    'browseVignettes()'. To cite Bioconductor, see
    'citation("Biobase")', and for packages 'citation("pkgname")'.
```

Loading the raw probe data

The dataset has been downloaded from GEO database and all the cel files can be found in the 'assignment' folder, on the path shown below:

In [2]:
```
celfiles = Sys.glob("/home/luna.kuleuven.be/r0603823/data/assignment/*.gz")
```

Read the data

In [3]:
```
celfiles
```

'/home/luna.kuleuven.be/r0603823/data/assignment/AAHC (1).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AAHC (2).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AAHC (3).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AAHC (4).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AAHC (5).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AAHC (6).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AAHC (7).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AAHC (8).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AS (1).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AS (2).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AS (3).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AS (4).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AS (5).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AS (6).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AS (7).gz'
'/home/luna.kuleuven.be/r0603823/data/assignment/AS (8).gz'

Next, we will load the raw probe data by using the ReadAffy() command

In [4]:
```
rawcel <- ReadAffy(filenames=celfiles)
```

Next step is to annotate the samples

In [5]:
```
sampleNames(rawcel)
```

'AAHC (1).gz' 'AAHC (2).gz' 'AAHC (3).gz' 'AAHC (4).gz' 'AAHC (5).gz'
'AAHC (6).gz' 'AAHC (7).gz' 'AAHC (8).gz' 'AS (1).gz' 'AS (2).gz' 'AS (3).gz'
'AS (4).gz' 'AS (5).gz' 'AS (6).gz' 'AS (7).gz' 'AS (8).gz'

As we see the rawcel object only knows the file names, therefore we need to make a list of the sample IDs (asthmatic patients or healthy donors), so that we let R know which sample is which. To do this we will make a list called 'samples' and put the sample IDs inside:

In [6]:
```
samples = c("healthy_donor_1",
            "healthy_donor_2",
              "healthy_donor_3",
              "healthy_donor_4",
              "healthy_donor_5",
              "healthy_donor_6",
              "healthy_donor_7",
              "healthy_donor_8",
             "asthmatic_1",
              "asthmatic_2",
             "asthmatic_3",
             "asthmatic_4",
              "asthmatic_5",
              "asthmatic_6",
             "asthmatic_7",
             "asthmatic_8")
```

After this we will merge the sample names and IDs into one data frame

In [7]:
```
pd <- data.frame(samples = samples)
rownames(pd) <- sampleNames(rawcel)
pd
```

	samples
AAHC (1).gz	healthy_donor_1
AAHC (2).gz	healthy_donor_2
AAHC (3).gz	healthy_donor_3
AAHC (4).gz	healthy_donor_4
AAHC (5).gz	healthy_donor_5
AAHC (6).gz	healthy_donor_6
AAHC (7).gz	healthy_donor_7
AAHC (8).gz	healthy_donor_8
AS (1).gz	asthmatic_1
AS (2).gz	asthmatic_2
AS (3).gz	asthmatic_3
AS (4).gz	asthmatic_4
AS (5).gz	asthmatic_5
AS (6).gz	asthmatic_6
AS (7).gz	asthmatic_7
AS (8).gz	asthmatic_8

This dataframe needs to be converted into an AnnotatedDataFrame, and in order to do that we will first load the Biobase library. What the Annotation procedure does is associating the phenotype data with the rawcel data (raw data). That way we associate the raw data we have with the phenotype of our samples (HC or SA).

In [8]:
```
library(Biobase)
phenoData(rawcel) <- AnnotatedDataFrame(data=pd)
```

Once we have the AnnotatedDataFrame we will use frma to normalise. But first we need to load the frma library.

In [9]:
```
library(frma)
```

```
Creating a generic function for ‘nchar’ from package ‘base’ in package ‘S4Vecto
rs’
```

Database normalization is used in order to organize the contents of our table, making the dataset more accurate, fast, efficient and trustworthy. The normalised file is saved in the pathway shown below:

In [10]:
```
eset <- frma(rawcel, verbose=TRUE)
save(eset, file="/home/luna.kuleuven.be/r0603823/data/assignment.eset")
```

```
Background Correcting ...


Normalizing ...

Summarizing ...
```

Now, we have an expression set loaded as an ExpressionSet. ExpressionSet is a container for high-throughput assays and experimental metadata. By printing the object we can get some basic data on the expression set.

In [11]:
```
eset
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 54675 features, 16 samples
  element names: exprs, se.exprs
protocolData: none
phenoData
  sampleNames: AAHC (1).gz AAHC (2).gz ... AS (8).gz (16 total)
  varLabels: samples
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation: hgu133plus2
```

The eset object contains the normalised data per probe and some experimental annotation. From what is printed we can see that:

> Our dataset includes a total of 54675 probes and 16 samples The names of each sample are presented (sampleNames) The platform used for annotation was:hgu133plus2

Gene names

The names of the samples are not very informative:

In [12]:
```
sampleNames(eset)
```

'AAHC (1).gz' 'AAHC (2).gz' 'AAHC (3).gz' 'AAHC (4).gz' 'AAHC (5).gz' 'AAHC (6).gz' 'AAHC (7).gz' 'AAHC (8).gz' 'AS (1).gz' 'AS (2).gz' 'AS (3).gz' 'AS (4).gz' 'AS (5).gz' 'AS (6).gz' 'AS (7).gz' 'AS (8).gz'

We are changing that by assigning sample names manually

In [13]:
```
eset.labels <- c("HC_1",
                 "HC_2",
                 "HC_3",
                 "HC_4",
                 "HC_5",
                 "HC_6",
                 "HC_7",
                 "HC_8",
                "SA_1",
                 "SA_2",
                 "SA_3",
                 "SA_4",
                 "SA_5",
                 "SA_6",
                 "SA_7",
                 "SA_8")

sampleNames(eset) <- eset.labels
```

Now the sample names represent if the sample comes from a control or a patient, and the which one

In [14]:
```
sampleNames(eset)
```

'HC_1' 'HC_2' 'HC_3' 'HC_4' 'HC_5' 'HC_6' 'HC_7' 'HC_8' 'SA_1' 'SA_2' 'SA_3' 'SA_4' 'SA_5' 'SA_6' 'SA_7' 'SA_8'

In [15]:
```
pData(phenoData(eset))
```

	samples
HC_1	healthy_donor_1
HC_2	healthy_donor_2
HC_3	healthy_donor_3
HC_4	healthy_donor_4
HC_5	healthy_donor_5
HC_6	healthy_donor_6
HC_7	healthy_donor_7
HC_8	healthy_donor_8
SA_1	asthmatic_1
SA_2	asthmatic_2
SA_3	asthmatic_3
SA_4	asthmatic_4
SA_5	asthmatic_5
SA_6	asthmatic_6
SA_7	asthmatic_7
SA_8	asthmatic_8

We can check the normalized expression level on a probe by probe basis (we choose to present only the data for the first 4 samples and the 5 first probes)

In [16]:
```
probe.counts <- exprs(eset)
probe.counts[1:5,1:4]
```

	HC_1	HC_2	HC_3	HC_4
1007_s_at	7.858796	7.973577	7.823374	7.815018
1053_at	4.743307	4.870033	4.969293	4.800928
117_at	6.238942	5.830911	5.979301	5.833766
121_at	8.310146	8.551032	8.408467	8.551527
1255_g_at	3.914073	3.677508	3.637221	3.449558

The data of all samples would look something like this

In [17]:
```
head(probe.counts)
```

	HC_1	HC_2	HC_3	HC_4	HC_5	HC_6	HC_7	HC_8
1007_s_at	7.858796	7.973577	7.823374	7.815018	7.797765	7.127558	7.705339	6.94821
1053_at	4.743307	4.870033	4.969293	4.800928	5.007958	6.297594	5.749584	6.12664
117_at	6.238942	5.830911	5.979301	5.833766	5.951767	6.257923	6.672200	6.21873
121_at	8.310146	8.551032	8.408467	8.551527	8.483914	7.863853	8.130164	7.82592
1255_g_at	3.914073	3.677508	3.637221	3.449558	3.707343	3.721185	3.551437	3.63645
1294_at	8.762248	8.533612	8.619347	8.617739	8.434989	8.670508	8.697701	8.47482

The probe names are not very informative. So, we will assign gene names to each probe. For that we need to load two extra libraries. The latter contains the annotation of our microarray platform (hg133+2)

In [18]:
```
library(annotate)
library(hgu133plus2.db)
```

```
Loading required package: AnnotationDbi
Loading required package: stats4
Loading required package: GenomeInfoDb
Loading required package: S4Vectors
Loading required package: IRanges
Loading required package: XML
Loading required package: org.Hs.eg.db
```

No we will create a mapping from probe to gene and by printing the table you can actually see the corresponding genes for the some probes

In [19]:
```
probe.names <- rownames(probe.counts)
gene.symbols <- getSYMBOL(probe.names, "hgu133plus2")
head(gene.symbols)
```

1007_s_at	'NA'
1053_at	'RFC2'
117_at	'HSPA6'
121_at	'PAX8'
1255_g_at	'GUCA1A'
1294_at	'NA'

How many genes do we have?

In [20]:
```
length(gene.symbols)
```

54675

That although is the number of probes and it does not really represent the number of genes since a single gene can have multiple probes.

Next we are trying to fix the raw expression table. The first step is to convert the raw expression to a data.frame with the gene symbols addes as an extra column at the end.

In [21]:
```
probe.counts.genesymbol <- as.data.frame(cbind(as.data.frame(probe.counts), gene.
head(probe.counts.genesymbol)
```

	HC_1	HC_2	HC_3	HC_4	HC_5	HC_6	HC_7	HC_8
1007_s_at	7.858796	7.973577	7.823374	7.815018	7.797765	7.127558	7.705339	6.94821
1053_at	4.743307	4.870033	4.969293	4.800928	5.007958	6.297594	5.749584	6.12664
117_at	6.238942	5.830911	5.979301	5.833766	5.951767	6.257923	6.672200	6.21873
121_at	8.310146	8.551032	8.408467	8.551527	8.483914	7.863853	8.130164	7.82592
1255_g_at	3.914073	3.677508	3.637221	3.449558	3.707343	3.721185	3.551437	3.63645
1294_at	8.762248	8.533612	8.619347	8.617739	8.434989	8.670508	8.697701	8.47482

Then we will remove all probes for which we do not have a gene symbol, and ensure that the gene symbol column contains only strings.

In [22]:
```
probe.counts.genesymbol = probe.counts.genesymbol[complete.cases(probe.counts.gen
probe.counts.genesymbol$gene.symbols <- as.character(probe.counts.genesymbol$gene
```

Now, we will have multiple probes per gene. To group the probes per gene, we will use a function from the excellent reshape2 package.

In [23]:
```
library(reshape2)
```

We will now melt and dcast the probe expression table into a gene count table.

In [24]:
```
m <- melt(probe.counts.genesymbol, id.vars = c("gene.symbols"))
```

Let's check the contents of m, not how the data is now in "long" form:

In [25]:
```
head(m)
```

gene.symbols	variable	value
RFC2	HC_1	4.743307
HSPA6	HC_1	6.238942
PAX8	HC_1	8.310146
GUCA1A	HC_1	3.914073
THRA	HC_1	6.526577
PTPN21	HC_1	3.950251

And now we aggregate into a counts table:

In [26]:
```
counts<-dcast(m, gene.symbols ~ variable, fun.aggregate=mean)
```

In [27]:
```
head(counts)
```

gene.symbols	HC_1	HC_2	HC_3	HC_4	HC_5	HC_6	HC_7	HC_
A1BG	6.783135	6.005207	6.409351	7.101967	6.772818	6.704954	6.929161	6.80
A1BG-AS1	5.333544	4.785200	5.758632	5.177936	5.229441	5.439741	5.304764	5.87
A1CF	4.335090	4.555904	4.573801	4.574014	4.563831	4.439237	4.528347	4.45
A2M	5.335486	5.357158	5.312197	5.423714	5.362729	4.893165	5.049978	4.80
A2M-AS1	6.171490	6.304766	6.812521	6.088466	6.296502	7.432015	7.575485	6.88
A2ML1	4.258969	4.056858	4.267614	4.013507	4.034405	4.195844	4.184892	4.17

As we can see there is a column called `gene.symbols`. We're turning these into the row names now:

In [28]:
```
rownames(counts) <- counts$gene.symbols
counts$gene.symbols <- NULL
```

In [29]:
```
head(counts)
```

	HC_1	HC_2	HC_3	HC_4	HC_5	HC_6	HC_7	HC_8	S
A1BG	6.783135	6.005207	6.409351	7.101967	6.772818	6.704954	6.929161	6.806240	6
A1BG-AS1	5.333544	4.785200	5.758632	5.177936	5.229441	5.439741	5.304764	5.877617	4
A1CF	4.335090	4.555904	4.573801	4.574014	4.563831	4.439237	4.528347	4.456341	4
A2M	5.335486	5.357158	5.312197	5.423714	5.362729	4.893165	5.049978	4.807754	4
A2M-AS1	6.171490	6.304766	6.812521	6.088466	6.296502	7.432015	7.575485	6.886836	7
A2ML1	4.258969	4.056858	4.267614	4.013507	4.034405	4.195844	4.184892	4.170089	3

The expression profile data are going through log2 transformation before going any further. This is done in order to normalize the data achieving an approximate normal distribution.

In [30]:
```
#counts.log2 <- log2(counts[1:7468, 1:16])
#head(counts.log2)
```

	HC_1	HC_2	HC_3	HC_4	HC_5	HC_6	HC_7	HC_8	S
A1BG	2.761952	2.586214	2.680178	2.828219	2.759756	2.745227	2.792681	2.766858	2
A1BG-AS1	2.415094	2.258579	2.525726	2.372377	2.386657	2.443538	2.407289	2.555231	2
A1CF	2.116062	2.187737	2.193394	2.193461	2.190245	2.150312	2.178984	2.155860	2
A2M	2.415620	2.421468	2.409309	2.439281	2.422967	2.290768	2.336277	2.265363	2
A2M-AS1	2.625619	2.656443	2.768189	2.606079	2.654550	2.893753	2.921338	2.783841	2
A2ML1	2.090504	2.020363	2.093430	2.004863	2.012356	2.068961	2.065190	2.060078	1

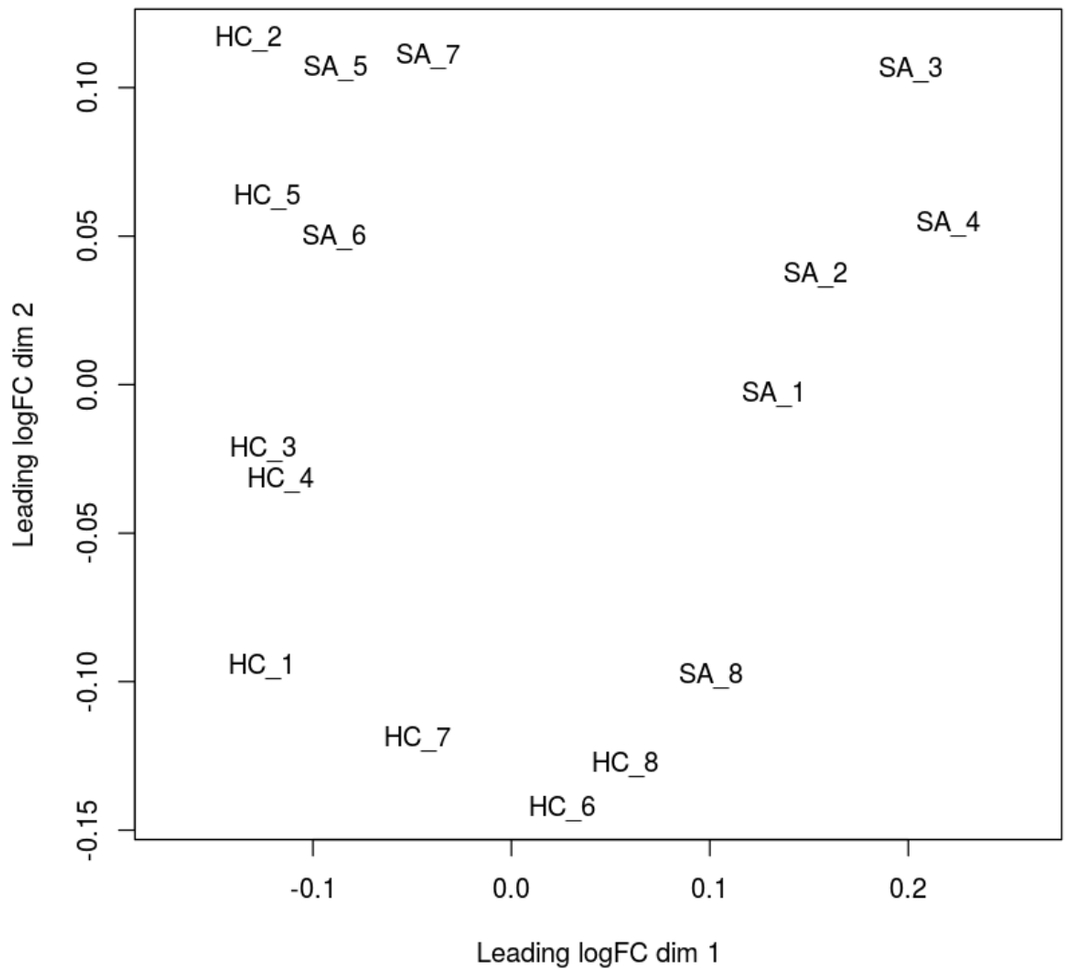
Now we will save this count table to disk for later use.

In [31]:
```
write.table(counts, 'GSE31773.counts.tsv', sep="\t", quote=FALSE, col.names=NA)
```

Quality Control

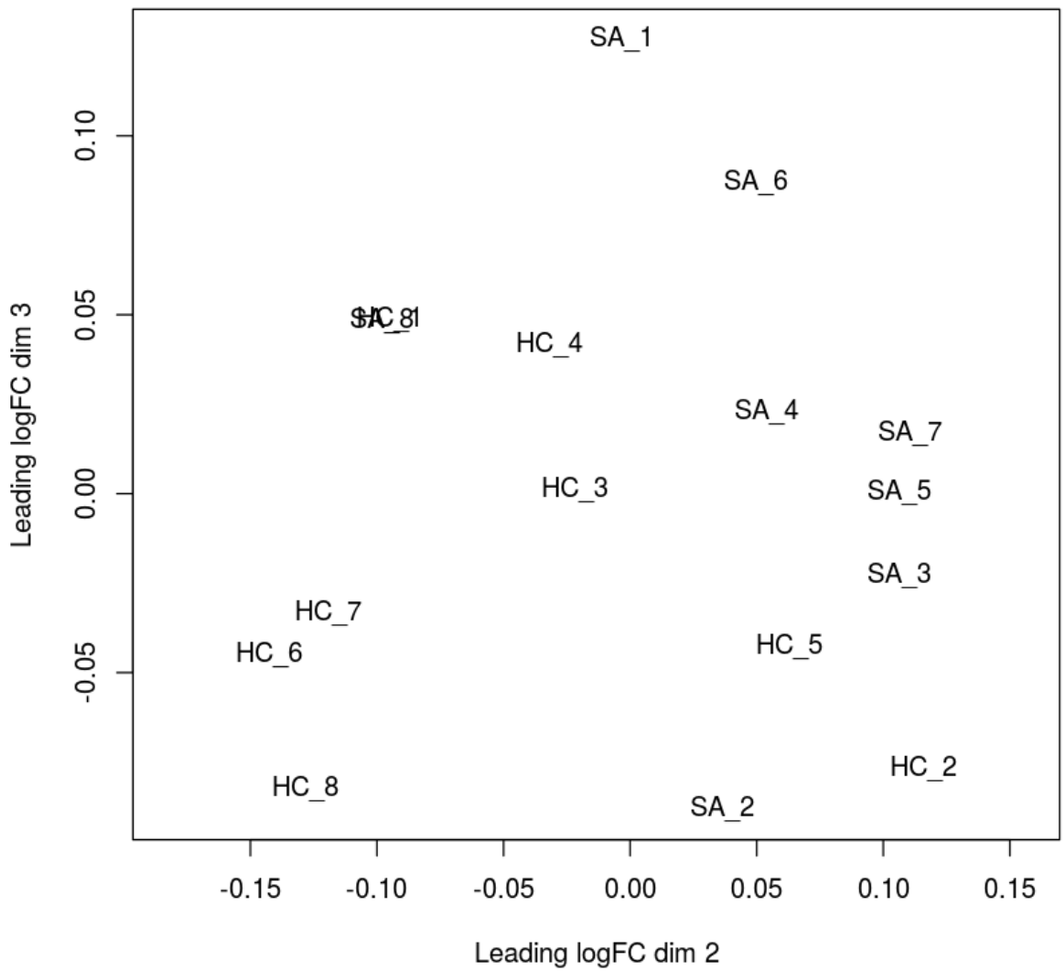
Checking the quality of the data can be done with the use of MDS tools. With the use of those tools basically, if two samples are similar, they are close together. In the graph below we can some distinction between the HC and SA samples, showing that there is a difference in gene expression between them.

In [32]:
```
plotMDS(counts)
```



MDS in dimensions 2 and 3: in this case the dinstinction is a bit clearer with the majority of SA samples being on the right half of the plot, whereas the majority of HC on the left.

In [33]:
```
plotMDS(counts, ndim=3, dim.plot=c(2,3))
```



As we can see our initial hypothesis for good quality data is barely fullfiled, since there is not a very clear distinction between AS patients and controls.

Differential Expression Analysis

Differentially expressed genes (DEGs) are screened out by using the limma package. Theferore we load the library:

In [34]:
```
library(limma)
```

We have to define the experimental conditions, so that the software knows which groups and samples it needs to compare.

In [35]:
```
expression <- factor(c( "control","control","control","control","control","contro
                       "SA", "SA", "SA", "SA", "SA", "SA", "SA"),
                 levels=c("control", "SA"))
expression
```

control control control control control control control control SA SA SA SA SA SA SA SA SA

Limma uses a linear model, to this end we'll build a model matrix:

In [36]:
```
design <- model.matrix(~expression)
design
```

	(Intercept)	expressionSA
1	1	0
2	1	0
3	1	0
4	1	0
5	1	0
6	1	0
7	1	0
8	1	0
9	1	1
10	1	1
11	1	1
12	1	1
13	1	1
14	1	1
15	1	1
16	1	1

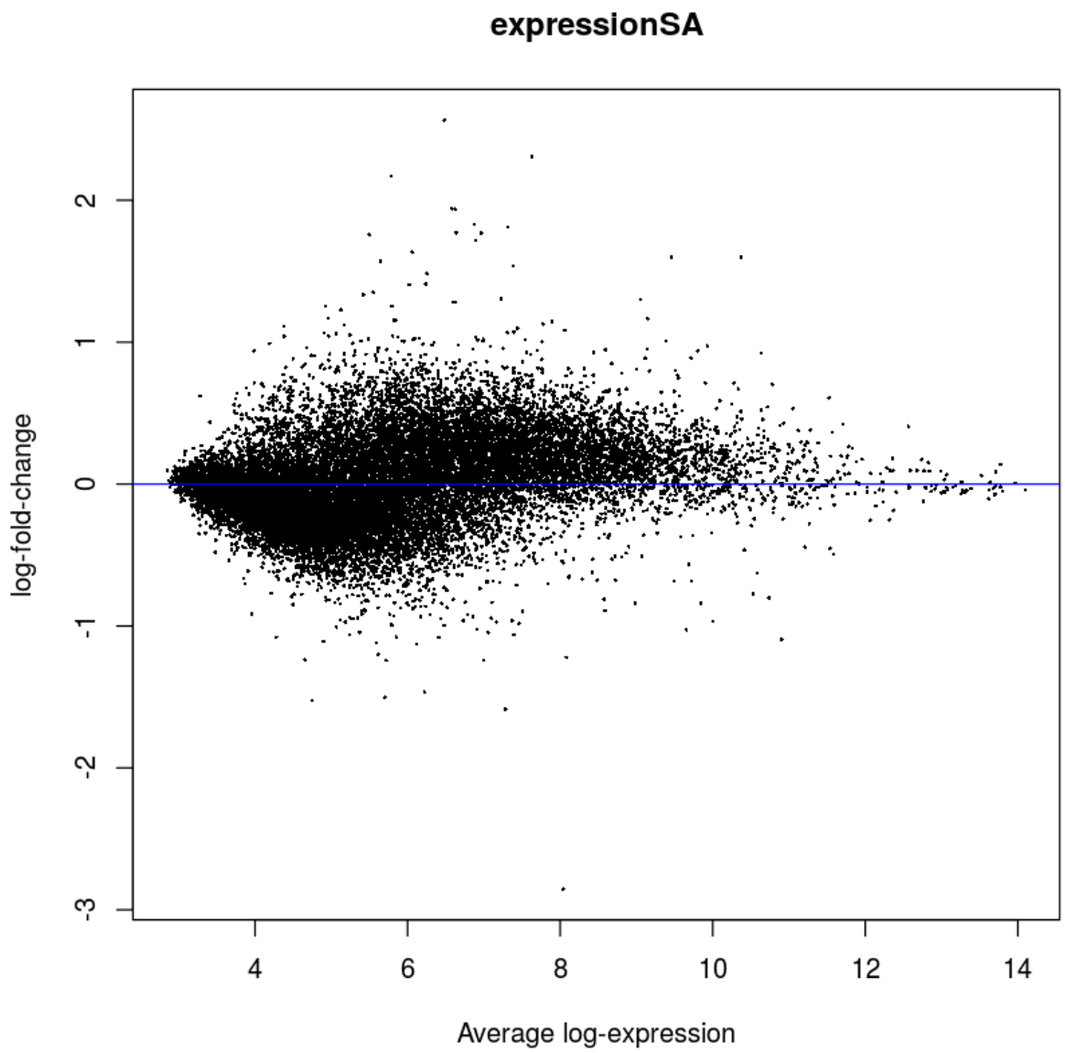
We will use the log2 transformed count table as the dataset to which a linear model will be fitted and the probabilities will be calculated.

In [49]:
```
fit <- eBayes(lmFit(counts, design))
```

MA plot

We make a plot for the differential expression analysis. It show the log ratio change of expression (M) vs the average expression (A). It gives you insight into the spread of differential expression. Based on the plot dots possitively upregulated or downregulated are shown as outliners of the main line.

In [50]:
```
limma::plotMA(fit, 2)
abline(0,0,col='blue')
```



Based on the plot above we can comment that genes seem to be both upregulated and dowregulated in AS patients. If we look at the log-fold changes, most genes are within the area of logFC -1 and 1 (logFC=1 means that the gene is upregulated by a 2-fold change, whereas logFC=-1, means down-regulation of the gene by a 2-fold change).

Extract a table of the top-ranked genes from a linear model fit. I want to visualise all my genes in an order from the most to the less DE ones so i put as number=7469(total number of genes I have in my dataset).

In [67]:
```
topTable(fit, coef=2)
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
ABCG1	0.6156932	5.895772	7.949421	3.351290e-07	0.003617068	6.692755
FXYD6	-0.9900161	5.784054	-7.916715	3.548058e-07	0.003617068	6.642436
RPL32P3	0.5077287	6.563759	7.381365	9.210764e-07	0.005052095	5.794732
CLDN12	0.7512713	4.762427	7.174582	1.345179e-06	0.005052095	5.455044
LINC01121	-0.6050225	5.338010	-7.070145	1.632332e-06	0.005052095	5.280868
PAPLN	-0.5644426	4.902225	-7.005011	1.843064e-06	0.005052095	5.171349
DMRTA2	-0.5805955	5.082582	-6.957764	2.013482e-06	0.005052095	5.091478
CTAGE5	0.4836920	5.877416	6.930922	2.117512e-06	0.005052095	5.045943
FAM83C	-0.7201035	4.552351	-6.769241	2.874166e-06	0.005052095	4.769203
CCDC60	-0.6516908	6.160207	-6.746165	3.003144e-06	0.005052095	4.729361

Correct for multiple testing

The cutoff used for significance is an FDR of less than 0.05. This means that there is a 5% chance each gene is not differentially expressed.

In [55]:
```
select = p.adjust(fit[,2]$p.value, method="BH") < 0.05
```

In [56]:
```
summary(select)
```

```
   Mode   FALSE    TRUE    NA's
logical   17222    3167       0
```

The above is an overview of how many genes are signficantly differentially expressed: 3167 genes

Next we are going to create an output file of our data

In [74]:
```
t = topTable(fit, coef=2, number=100000)
```

In [75]:
```
write.table(t, "GSE31773.tophits.tsv", sep="\t", quote=FALSE, col.names=NA)
```

In [78]:
```
head(t)
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
ABCG1	0.6156932	5.895772	7.949421	3.351290e-07	0.003617068	6.692755
FXYD6	-0.9900161	5.784054	-7.916715	3.548058e-07	0.003617068	6.642436
RPL32P3	0.5077287	6.563759	7.381365	9.210764e-07	0.005052095	5.794732
CLDN12	0.7512713	4.762427	7.174582	1.345179e-06	0.005052095	5.455044
LINC01121	-0.6050225	5.338010	-7.070145	1.632332e-06	0.005052095	5.280868
PAPLN	-0.5644426	4.902225	-7.005011	1.843064e-06	0.005052095	5.171349

In [79]:
```
summary (t)
```

```
     logFC             AveExpr             t              P.Value
 Min.   :-2.853805   Min.   : 2.849   Min.   :-7.91671   Min.   :0.0000003
 1st Qu.:-0.181630   1st Qu.: 4.482   1st Qu.:-1.65637   1st Qu.:0.0212264
 Median :-0.005008   Median : 5.504   Median :-0.05235   Median :0.1201868
 Mean   : 0.013874   Mean   : 5.792   Mean   :-0.06973   Mean   :0.2496103
 3rd Qu.: 0.202736   3rd Qu.: 6.828   3rd Qu.: 1.61391   3rd Qu.:0.4135215
 Max.   : 2.564774   Max.   :14.098   Max.   : 7.94942   Max.   :0.9999739
   adj.P.Val              B
 Min.   :0.003617   Min.   :-6.229
 1st Qu.:0.084893   1st Qu.:-5.880
 Median :0.240320   Median :-4.970
 Mean   :0.337438   Mean   :-4.412
 3rd Qu.:0.551315   3rd Qu.:-3.472
 Max.   :0.999974   Max.   : 6.693
```

The generated .tsv file is opened in Excel and the first 3167 genes are the ones that are significantly differentially expressed.