Analysis of gene expression

Job Maathuis

9-3-2022

Contents

1. Exploratory Data Analysis

1.1 Loading in the data

The data consists of gene expression data which is available as raw count data in a single text file. For this reason the read.table() function can be used and no merging is needed. The column names were changed by using the 'patient_codes.csv'.

```
setwd('C:/Users/jobma/Documents/School/Bio-informatica/Jaar_2/Kwartaal_3/practicum/')

data <- read.table('counts.txt', header = T, row.names = 1)
patient_codes <- read.csv('patient_codes.csv', header=F)
# shortening patient names by deleting the middle part
patient_codes[,2] <- gsub('-.*-', '-', patient_codes[,2])
colnames(data) <- patient_codes[,2][patient_codes[,1] == colnames(data)]
```

1.2 Data structure

The dataset has the following structure

```
library(pander)
pander(head(data, 5))
```

Table 1: Table continues below

	m29-TR1	m42-TR1	m43-TR1	s10-TR1	s11-TR1	m29-TR2
A1BG	0	0	2	0	1	0
A2M	1	2	1	1	0	0
A2ML1	0	0	0	0	0	1
A4GALT	0	0	0	0	0	0
AAAS	8	6	4	9	8	3

Table 2: Table continues below

	m42-TR2	m43-TR2	s10-TR2	s11-TR2	m29-TR3	m42-TR3
A1BG	0	0	0	0	0	0
A2M	1	0	1	1	3	1
A2ML1	0	0	0	0	0	0
A4GALT	0	0	0	0	0	0
AAAS	12	10	10	19	16	15

	m43-TR3	s10-TR3	s11-TR3
A1BG	1	1	0
A2M	1	0	0
A2ML1	0	0	0
A4GALT	0	0	0
AAAS	15	9	12

```
print(dim(data))
```

```
## [1] 16282    15
```

```
print(str(data))
```

```
## 'data.frame':    16282 obs. of  15 variables:
##  $ m29-TR1: int  0 1 0 0 8 2 0 0 21 163 ...
##  $ m42-TR1: int  0 2 0 0 6 1 0 0 12 177 ...
##  $ m43-TR1: int  2 1 0 0 4 0 0 0 9 159 ...
##  $ s10-TR1: int  0 1 0 0 9 1 0 0 13 163 ...
##  $ s11-TR1: int  1 0 0 0 8 1 0 0 18 133 ...
##  $ m29-TR2: int  0 0 1 0 3 3 0 0 18 145 ...
##  $ m42-TR2: int  0 1 0 0 12 3 0 0 14 180 ...
##  $ m43-TR2: int  0 0 0 0 10 1 0 0 17 151 ...
##  $ s10-TR2: int  0 1 0 0 10 1 0 0 19 179 ...
##  $ s11-TR2: int  0 1 0 0 19 2 0 0 30 238 ...
##  $ m29-TR3: int  0 3 0 0 16 0 0 0 20 159 ...
##  $ m42-TR3: int  0 1 0 0 15 1 0 0 19 173 ...
##  $ m43-TR3: int  1 1 0 0 15 1 0 0 17 142 ...
##  $ s10-TR3: int  1 0 0 0 9 3 0 0 12 171 ...
##  $ s11-TR3: int  0 0 0 0 12 0 0 0 23 176 ...
## NULL
```

As can be seen above, the data consists out of 16,282 rows, corresponding to the genes, and 15 columns, which are the different samples. The structure of the samples is as follows:

- 3 patients with a mild disease severity (m29, m42, m43), each having 3 technical repeats
- 2 patients with a severe disease severity (s10, s11), each having 3 technical repeats

To distinguish between these groups, some variables are made.

```
# variable for dataset name, which can be used later
name_dataset <- 'GSE181032'

# mild patients
m29 <- c(1, 6, 11)
m42 <- c(2, 7, 12)
m43 <- c(3, 8, 13)
m.all <- c(m29, m42, m43)

# severe patients
s10 <- c(4, 9, 14)
s11 <- c(5, 10, 15)
s.all <- c(s10, s11)
```

To get a quick look at the data the summary() function is used. With this function the minimum, first quartile, median, mean, third quartile and the maximum value of each column are obtained.

```
summary(data)
```

```
##      m29-TR1            m42-TR1            m43-TR1            s10-TR1
##  Min.   :    0.00   Min.   :    0.00   Min.   :    0.00   Min.   :    0.00
##  1st Qu.:    0.00   1st Qu.:    0.00   1st Qu.:    0.00   1st Qu.:    0.00
##  Median :    4.00   Median :    4.00   Median :    3.00   Median :    3.00
##  Mean   :   34.36   Mean   :   35.55   Mean   :   30.43   Mean   :   32.75
##  3rd Qu.:   17.00   3rd Qu.:   18.00   3rd Qu.:   14.00   3rd Qu.:   16.00
##  Max.   :11538.00   Max.   :11394.00   Max.   :10072.00   Max.   :11121.00
##      s11-TR1            m29-TR2            m42-TR2            m43-TR2
##  Min.   :    0.0   Min.   :    0.00   Min.   :    0.00   Min.   :    0.00
##  1st Qu.:    0.0   1st Qu.:    0.00   1st Qu.:    0.00   1st Qu.:    0.00
##  Median :    4.0   Median :    4.00   Median :    4.00   Median :    3.00
##  Mean   :   35.5   Mean   :   36.68   Mean   :   37.52   Mean   :   34.99
##  3rd Qu.:   20.0   3rd Qu.:   18.00   3rd Qu.:   18.00   3rd Qu.:   16.00
##  Max.   :10663.0   Max.   :12369.00   Max.   :12465.00   Max.   :11447.00
##      s10-TR2            s11-TR2            m29-TR3            m42-TR3
##  Min.   :    0.00   Min.   :    0.00   Min.   :    0.0   Min.   :    0.00
##  1st Qu.:    0.00   1st Qu.:    0.00   1st Qu.:    0.0   1st Qu.:    0.00
##  Median :    4.00   Median :    6.00   Median :    4.0   Median :    4.00
##  Mean   :   36.32   Mean   :   47.34   Mean   :   40.7   Mean   :   37.62
##  3rd Qu.:   18.00   3rd Qu.:   28.00   3rd Qu.:   20.0   3rd Qu.:   19.00
##  Max.   :12091.00   Max.   :13122.00   Max.   :13664.0   Max.   :12547.00
##      m43-TR3            s10-TR3            s11-TR3
##  Min.   :    0.0   Min.   :    0.00   Min.   :    0.00
##  1st Qu.:    0.0   1st Qu.:    0.00   1st Qu.:    0.00
##  Median :    4.0   Median :    3.00   Median :    5.00
##  Mean   :   35.6   Mean   :   34.92   Mean   :   40.61
##  3rd Qu.:   17.0   3rd Qu.:   17.00   3rd Qu.:   22.00
##  Max.   :12015.0   Max.   :11651.00   Max.   :12333.00
```

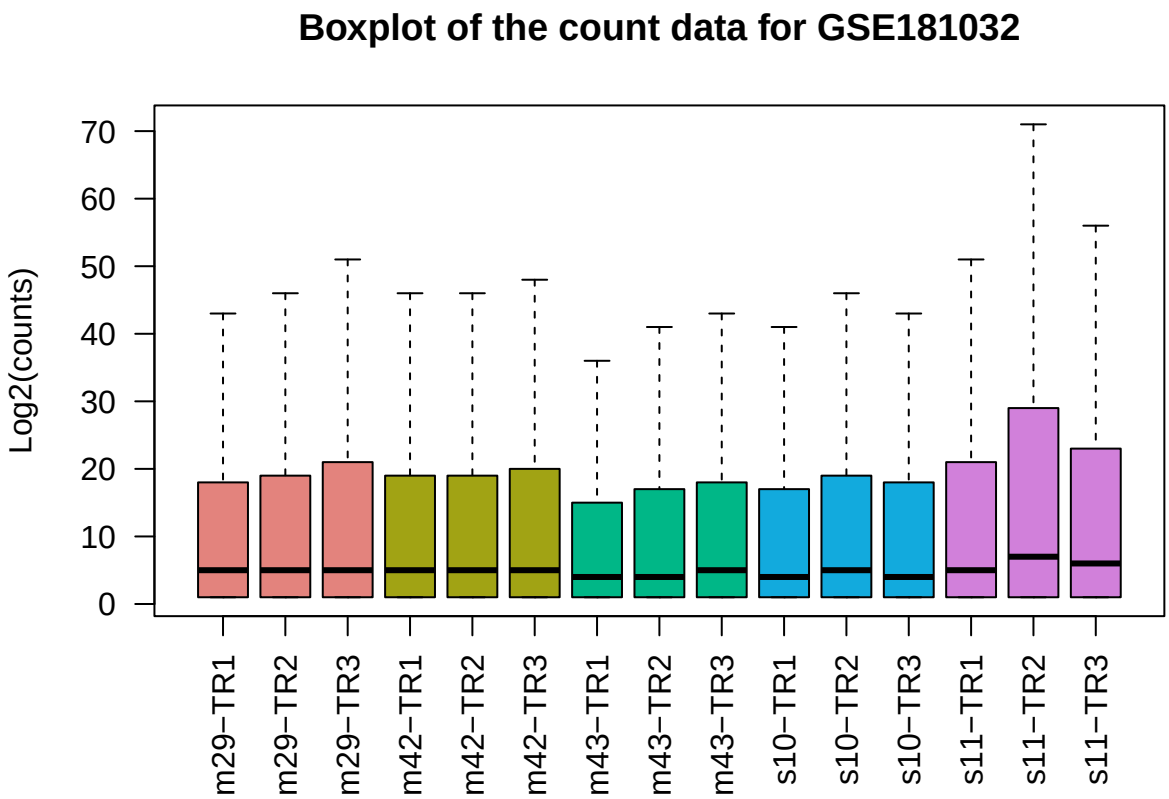
In the data above a large difference between the maximum values and the other values can be observed. This is probably due to the fact that the data contains a lot of zeros or low values. In a biological context this means that most of the genes that are analysed are 'off' and only a few genes are being expressed at time of the RNA-seq analysis.

1.3 Data visualisation

1.3.1 Boxplot

Next a boxplot can be made to visualise the structure of the data.

```
library(scales)
colors <- hue_pal(c=70)(5)
boxplot((data[,c(m29, m42, m43, s10, s11)] + 1),
        main = 'Boxplot of the count data for GSE181032',
        ylab = 'Log2(counts)', outline = F, las = 2,
        col=rep(colors, each=3))
```

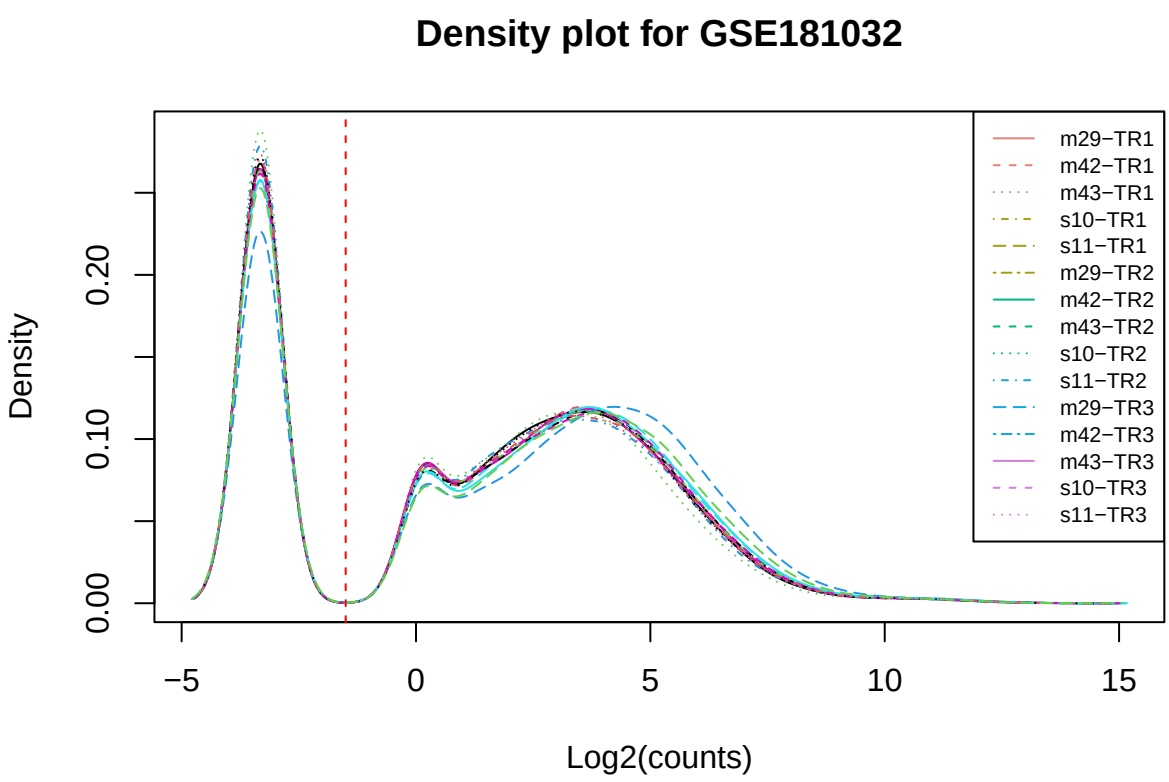


For every patient the repeats shows the same distrubution. This is also the case if the patients are compared. A somewhat larger deviation can be seen in s11-TR2 data. The large differences between the maximum values and the other values reflect the summary data in the previous chapter.

1.3.2 Density plot

The distrubtion of the count data can also be visualised using a density plot.

```
library(affy)
plotDensity(log2(data + 0.1),  main = 'Density plot for GSE181032',
       xlab = 'Log2(counts)', ylab = 'Density')
legend('topright', names(data), lty=c(1:ncol(data)),
       cex = 0.7, col=rep(colors, each=3))
abline(v=-1.5, lwd=1, col='red', lty=2)
```

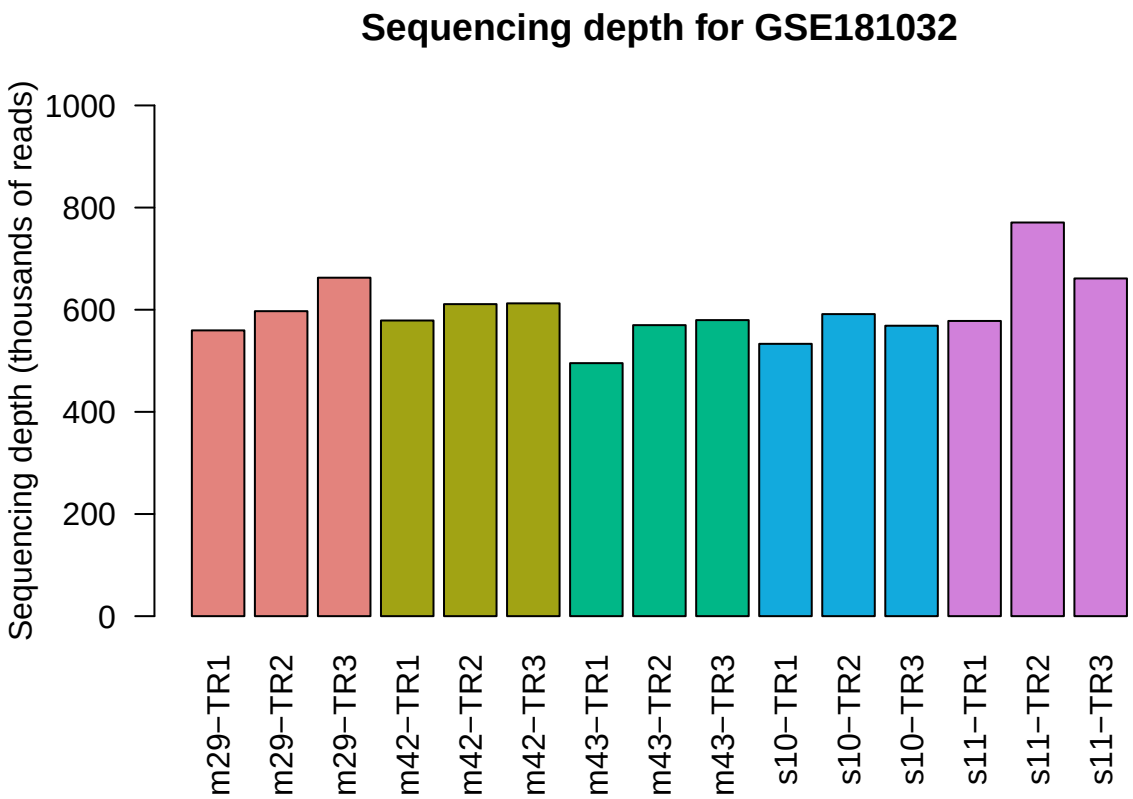


All of the lines follow the same distrubution, where only m29-TR3 is a bit variated from the other data. This variation is not much, so this will probably not cause any big problems. The large peak at the left of the red stippled line is caused by all of the zero's in the dataset.

1.3.3 Barplot

The amount of reads, also known as the read depth, can differ widely between all of the measured samples. To get an overview of the differences in read depth a barplot is made.

```
barplot(colSums(data[,c(m29, m42, m43, s10, s11)]) / 1e3, las = 2,
        ylab = 'Sequencing depth (thousands of reads)',
        main = 'Sequencing depth for GSE181032',
        col=rep(colors, each=3), ylim = c(0, 1000))
```



In the barplot a not much deviation can be seen between samples or within the samples. Only patient s11 has some larger deviation between the minimum and maximum sequencing depth. For this reason the dataset is being normalized. Normalization is achieved by using a variance stabilizing transformation (vst) form the DESeq2 library. When the data is normalized sample distances can be calculated.

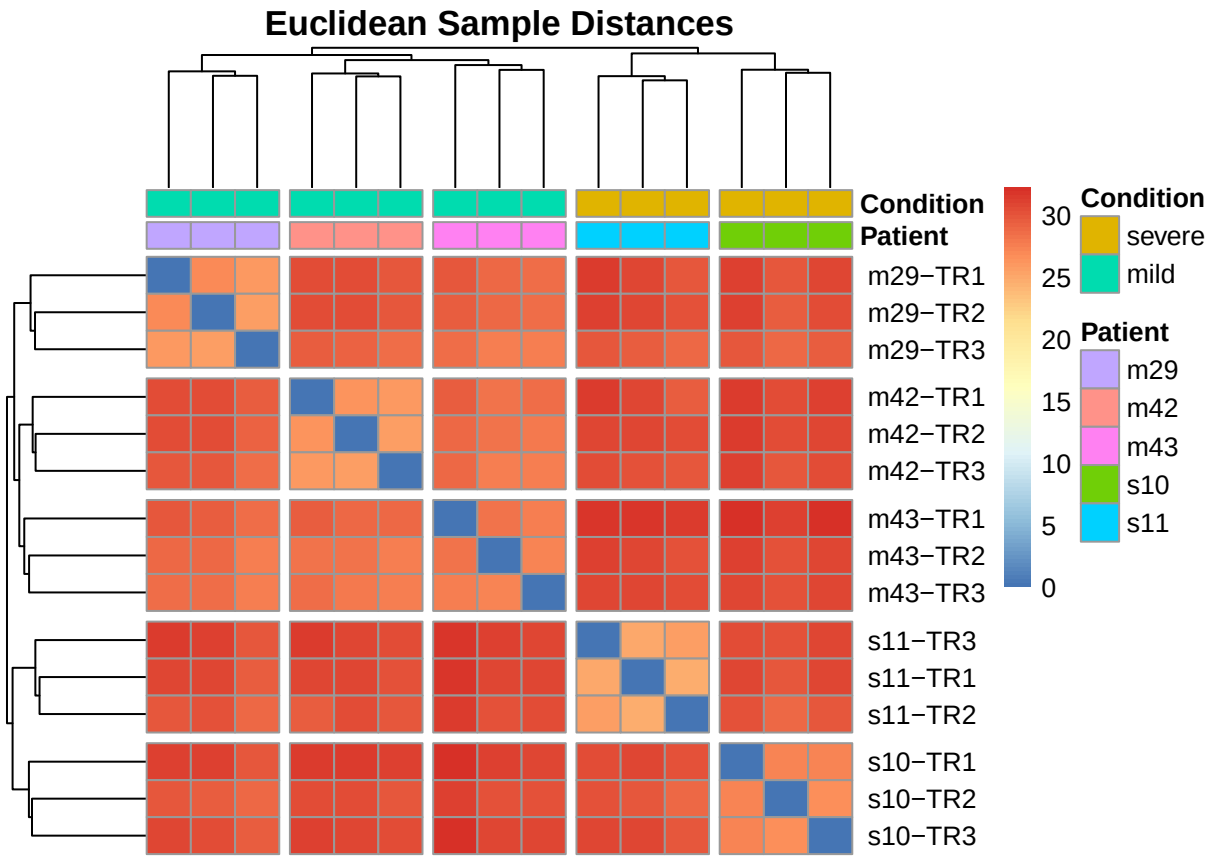
```
library("DESeq2")
# create a DESeq dataset
ddsMat <- DESeqDataSetFromMatrix(countData = data,
                                 colData = data.frame(samples = names(data)),
                                 design = ~ 1)
# normalization
rld.dds <- vst(ddsMat)
# obtain normalized data
rld <- assay(rld.dds)
# transposes and calculate distances
sampledists <- dist(t(rld))
```

1.3.4 Heatmap

Using the calculated sample distances a heatmap can be created. The clustering is based on condition and patient.

```
library(pheatmap)

# Convert the sample distances into a matrix for creating a heatmap
sampleDistMatrix <- as.matrix(sampledists)
```

```
# create annotation, which represents the design of the data
annotation <- data.frame(Patient = factor(rep(1:5, times = 3),
                                          labels = c("m29", "m42", "m43", "s10", "s11")),
                         Condition = factor(rep(c(rep("mild", times = 3),
                                                  rep("severe", times = 2)), times = 3),
                                            levels = c('severe', 'mild')),
                         Repeat = factor(rep(rep(1:3, times = 1), each = 5),
                                         labels = c("R1", "R2", "R3")))

# Set the rownames of the annotation dataframe to the sample names (
rownames(annotation) <- names(data)

# Create heatmap
pheatmap(sampleDistMatrix, show_colnames = FALSE,
         annotation_col = annotation[,c(1,2)],
         clustering_distance_rows = sampledists,
         clustering_distance_cols = sampledists,
         main = "Euclidean Sample Distances",
         cutree_cols = 5,
         cutree_rows = 5)
```

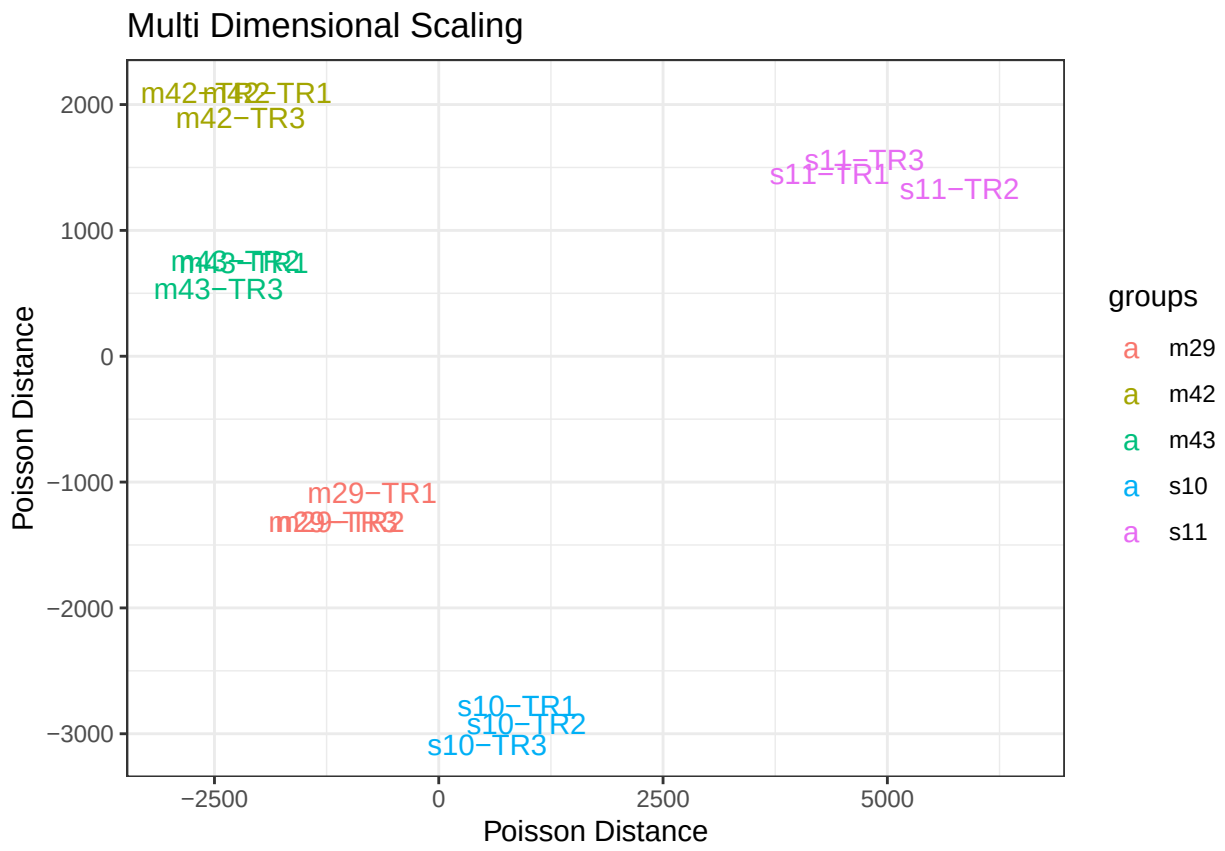


In the heatmap a clustering can be seen. The mild and severe patients are divided. The mild patients are orange colored and the severe patients show a dark red color. This color difference, and therefor the overal expression, is not very large. Furthermore, each patient except for m43 can be seen as a cluster, because is shows a light ...

1.3.5 Multi-dimensional scaling

With the use of multidimensional scaling a visual representation can be made of the dissimilarities of the samples. These dissimilarities are based upon Poission distances. It is expected that the repeats of each patient are close to each other because they show less dissimilarities when compared to other samples. However, if this is not the case the data may need to be cleaned or outliers needs to be removed.

```
library('PoiClaClu')
library('ggplot2')

dds <- assay(ddsMat)
# Calculate distances using Poisson distances
poisd <- PoissonDistance(t(dds))
# Extract the matrix with distances
samplePoisDistMatrix <- as.matrix(poisd$dd)
# Calculate the MDS and get the X- and Y-coordinates
mdsPoisData <- data.frame(cmdscale(samplePoisDistMatrix))

# Rename col names
names(mdsPoisData) <- c('x_coord', 'y_coord')

# Separate the annotation factor (as the variable name is used as label)
groups <- factor(rep(1:5, times=3),
                 labels = c("m29", "m42", "m43", "s10", "s11"))
coldata <- names(data)

# Create the plot using ggplot
ggplot(mdsPoisData, aes(x_coord, y_coord, color = groups, label = coldata)) +
  geom_text(size = 4) +
  ggtitle('Multi Dimensional Scaling') +
  labs(x = "Poisson Distance", y = "Poisson Distance") +
  xlim(-3000, 6500) +
  theme_bw()
```



All of the technical repeats from each patient are clustered together. Only s11-TR2 is somewhat seperated from the other repeats of the s11 patient, but this distance is still small. For these reasons, the data does not have to be cleaned since there are no clear outliers visible. Furthermore, all of the mild patients are on the left, while the severe patience are a bit shifted to the right (all positive x-axis values)

2 Discovering differentialy expressed genes (DEGs)

2.1 Manual pre-processing

Later on we will use libraries in order to normalize the data and filter out low count genes. This is essential because these low count genes may disturb the statistical test that these libraries apply. However, in this pragraph the normalization and filtering will be done manually in order to get some insight into what is going during these steps.

First the data is normalized using log2 transformation on the fragments per million mapped fragments (FPM) data. Low count genes are then filtered out based on the criterium that the total FPM of a gene is higher or equal to 3. If this is note the case the whole gene will be left out.

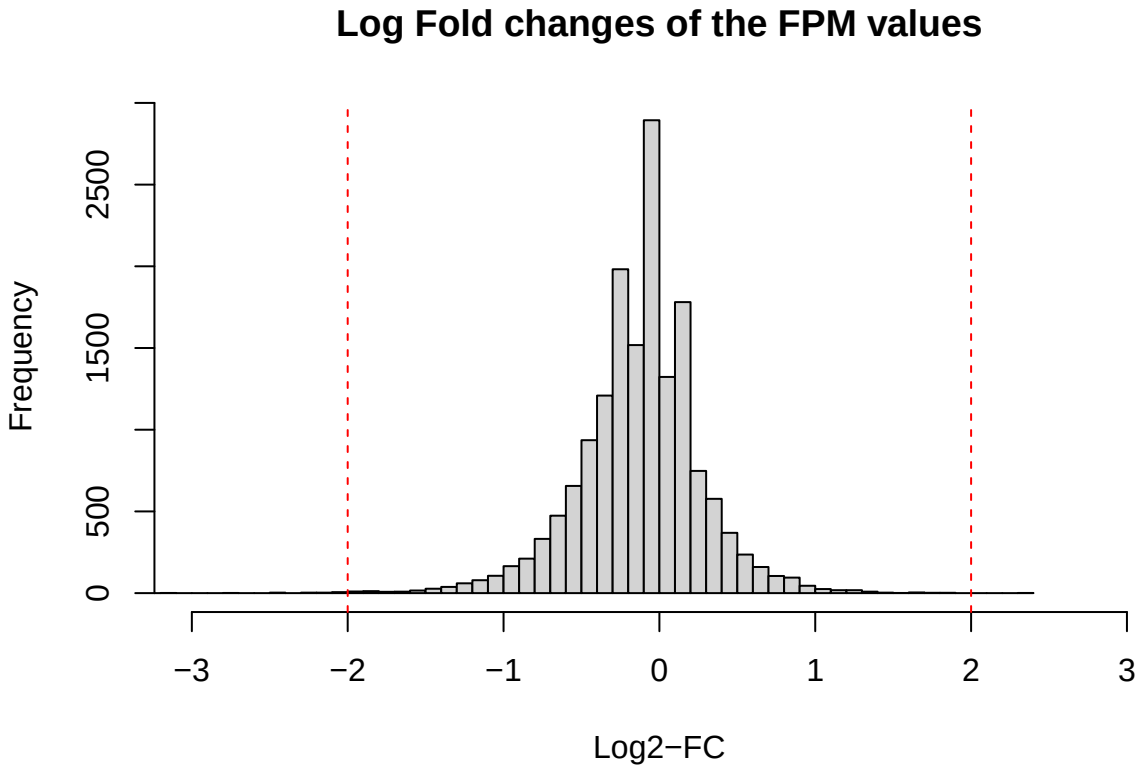
```
# Perform a naive FPM4 normalization
# Note: log transformation includes a pseudocount of 11
data.fpm <- log2( (data / (colSums(data) / 1e6)) + 1 )


data.fpm.filtered <- data.fpm[rowSums(data.fpm) >= 3,]
cat(nrow(data.fpm) - nrow(data.fpm.filtered), "genes have been filtered out")
```

```
## 3326 genes have been filtered out
```

This process results in the filtering out of 3326 genes which is 20.4274659 % of the total genes

Using these FPM values the log fold changes can be calculated between the mild and severe groups. With the log fold changes a histogram can be made to see the distribution.

```
data.fpm$m.mean <- rowMeans(data.fpm[m.all])
data.fpm$s.mean <- rowMeans(data.fpm[s.all])
data.fpm$lfc <- data.fpm$m.mean - data.fpm$s.mean

hist(data.fpm$lfc, breaks=80, xlab = "Log2-FC",
     main = "Log Fold changes of the FPM values",
     xlim = c(-3, 3))
abline(v=-2, col = 'red', lty=2)
abline(v=2, col = 'red', lty=2)
```



As can be seen in the histogram, most of the Log Fold changes are within the -2 and 2 boundaries. These are the boundaries used later on to determine the differential expressed genes (DEG's). So seeing this histogram it is not expected that there are a lot of DEG's when comparing the mild and severe patients.

2.2 DESeq2

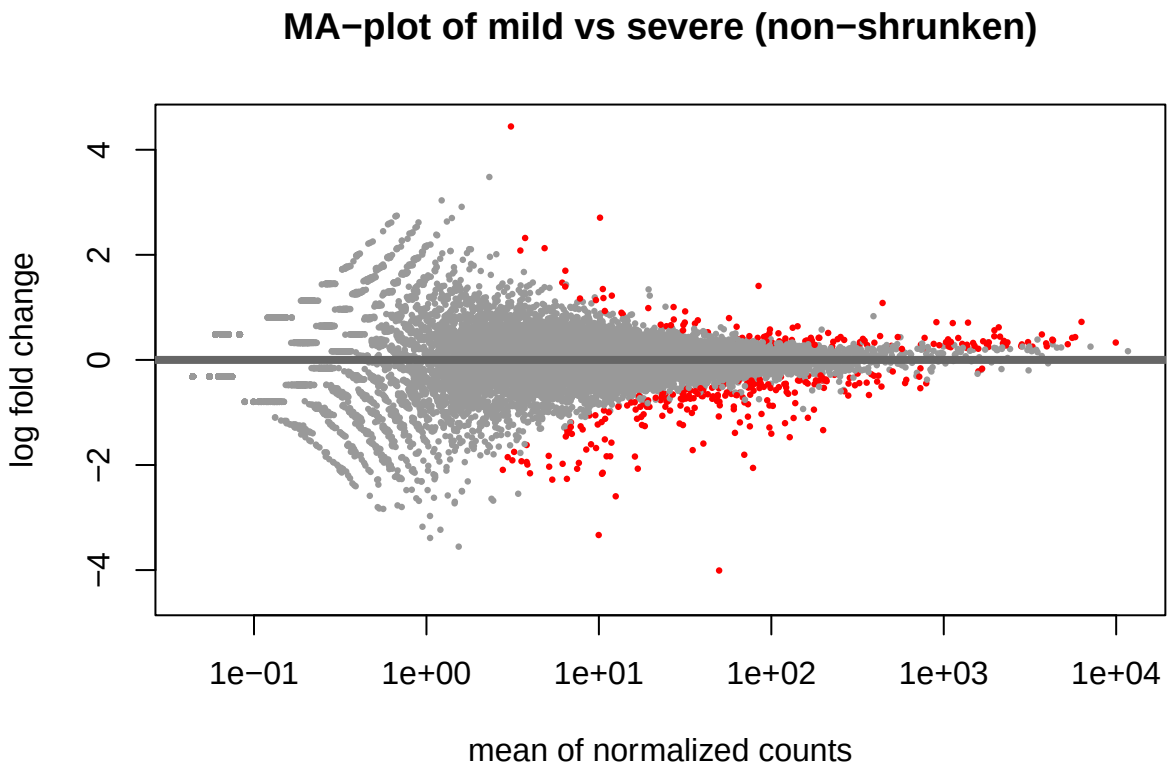
The DESeq2 package can be used for differential analysis of high-dimensional count data. It will apply a normalization (which was done manually in the previous paragraph) on the data and pefrom a differential expression analysis. The technical repeats are also combined using the collapseReplicates function.

```
dds <- DESeqDataSetFromMatrix(countData = data,
                              colData = annotation,
                              design = ~ Condition)
dds <- DESeq(dds, betaPrior = FALSE)
dds.collapse <- collapseReplicates(dds, groupby = annotation$Patient)
res <- results(dds.collapse)
pander(as.data.frame(head(res)))
```

	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
A1BG	0.3677	0.1646	1.521	0.1082	0.9138	NA
A2M	0.8517	1.319	1.077	1.225	0.2207	NA
A2ML1	0.06701	0.4852	3.135	0.1548	0.877	NA
A4GALT	0	NA	NA	NA	NA	NA
AAAS	9.928	0.0104	0.3084	0.03373	0.9731	0.9948
AACS	1.311	0.1606	0.8086	0.1986	0.8426	NA

The results that are obtained from this DESeqDataSet object and plotted in an MA-plot. This plot visualizes the differences between the log transformed data of the mild and severe groups, better known as the log fold change.

```
DESeq2::plotMA(res, ylim = c(-4.5, 4.5),
       main = "MA-plot of mild vs severe (non-shrunken)", alpha = 0.05, colSig = "red")
```



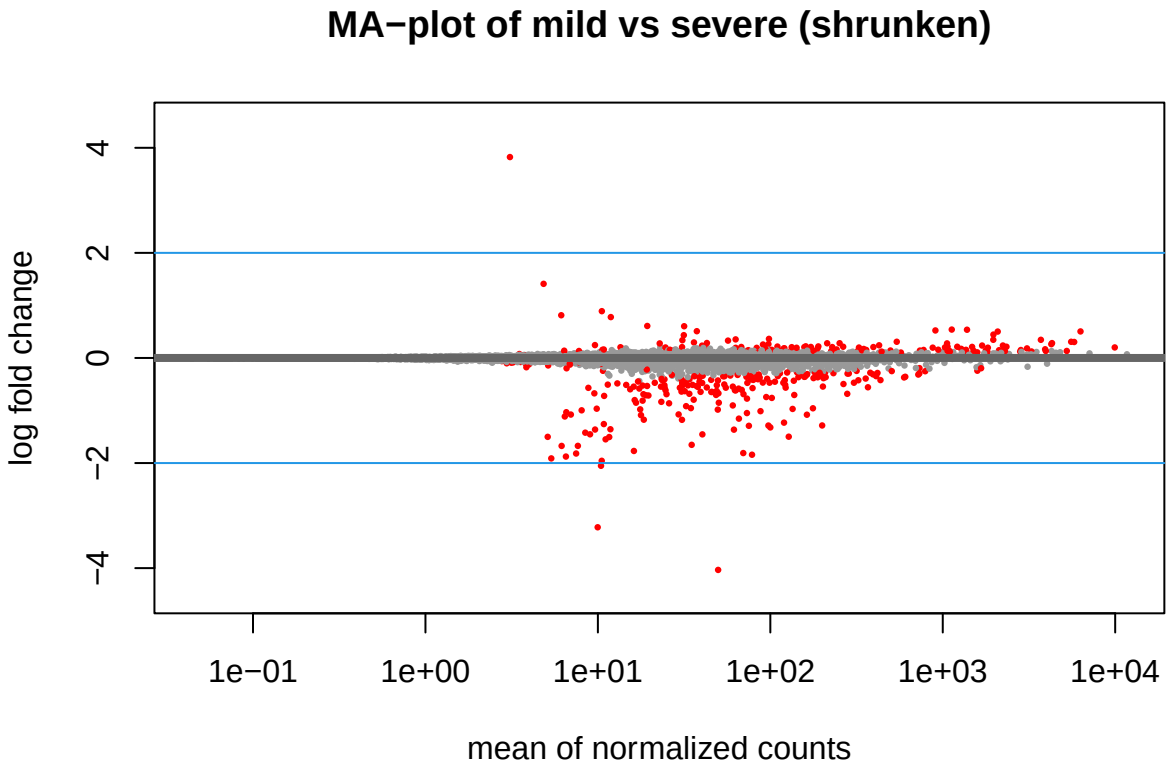
The red-coloured dots are significant genes (p.value < 0.05) and the grey dots are insignificant genes. On

the left side of the MA-plot a lot of noise can be seen. This is due to the low count genes. In order to remove this noise the log fold changes can be shrunken. This is done by the following code-block.

```
res <- lfcShrink(dds.collapse, coef =  "Condition_mild_vs_severe", type = "apeglm")
```

```
## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##     sequence count data: removing the noise and preserving large differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895
```

```
DESeq2::plotMA(res,  main = "MA-plot of mild vs severe (shrunken)",
               ylim = c(-4.5, 4.5), alpha = 0.05, colSig = "red")
abline(h=c(-2, 2), col=4)
```

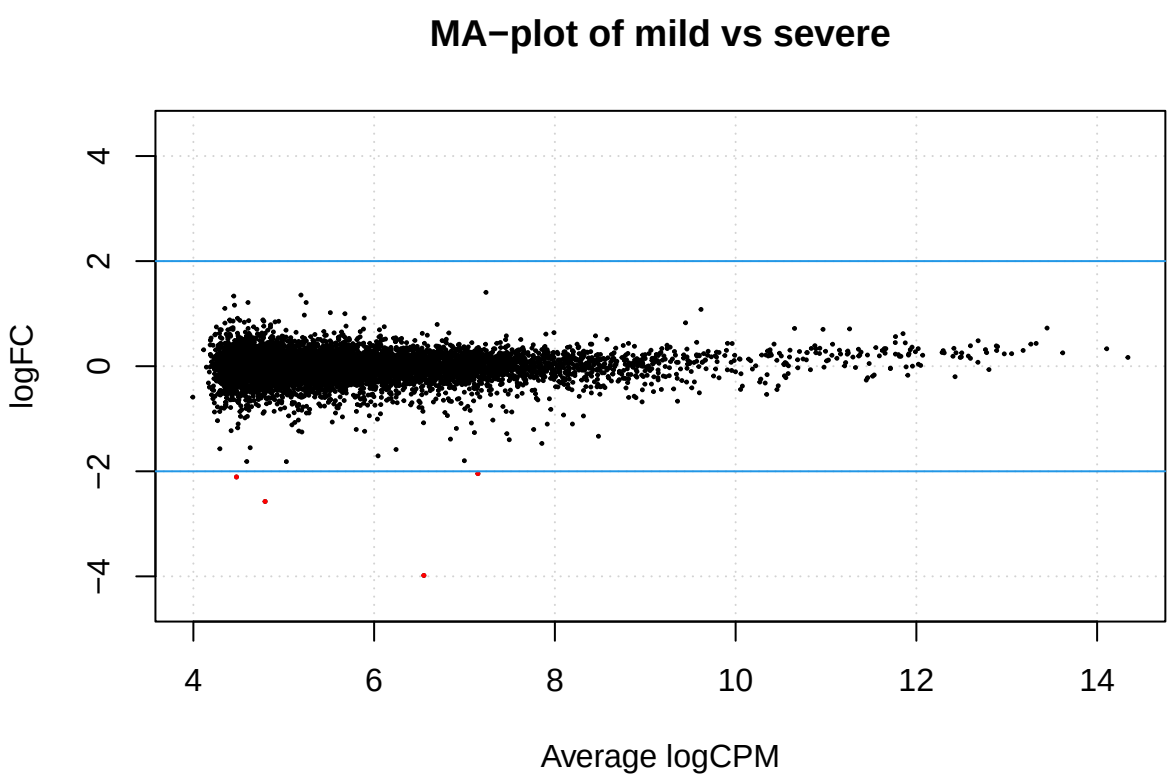


The noise has now successfully been removed, which results in a cleaner dataset. The structure and the amount of the significant points (red- coloured) is the same, only closer to 0. The amount of significant genes is fairly large, even at low log fold changes. In contrast, there only a few of significant genes having a log fold change above 2 or below -2. The log fold changes will be the biggest discriminating factor in selecting DEG's.

2.3 EdgeR

EdgeR is a similar technique as DESeq2 and will also be used to discover DEG's. The results of each technique will be compared.

```r
library(edgeR)
dge <- DGEList(counts = data, group = annotation$Condition)


design <- model.matrix(~ annotation$Condition)
keep <- filterByExpr(dge, design)
dge <- dge[keep, , keep.lib.sizes=FALSE]

dge <- calcNormFactors(dge)
dge <- estimateDisp(dge, design)
et <- exactTest(dge)

# variable which is later used to create the volcano plot
res.edger <- topTags(et, n = Inf)

deGenes <- decideTestsDGE(et, p=0.05, lfc=2)
deGenes <- rownames(et)[as.logical(deGenes)]
plotSmear(et, de.tags=deGenes, ylim = c(-4.5, 4.5), main = "MA-plot of mild vs severe")
abline(h=c(-2, 2), col=4)
```

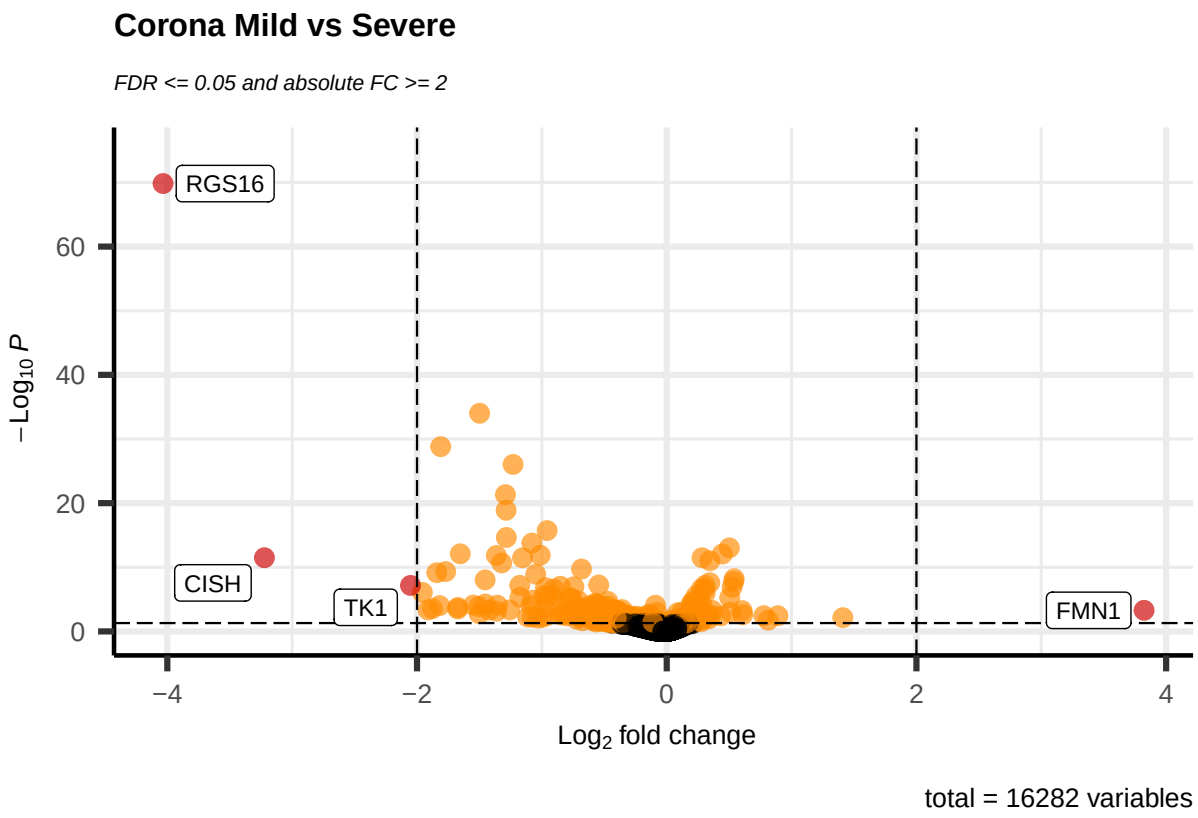


3 Data Analysis and Visualization

3.1 Volcano plot

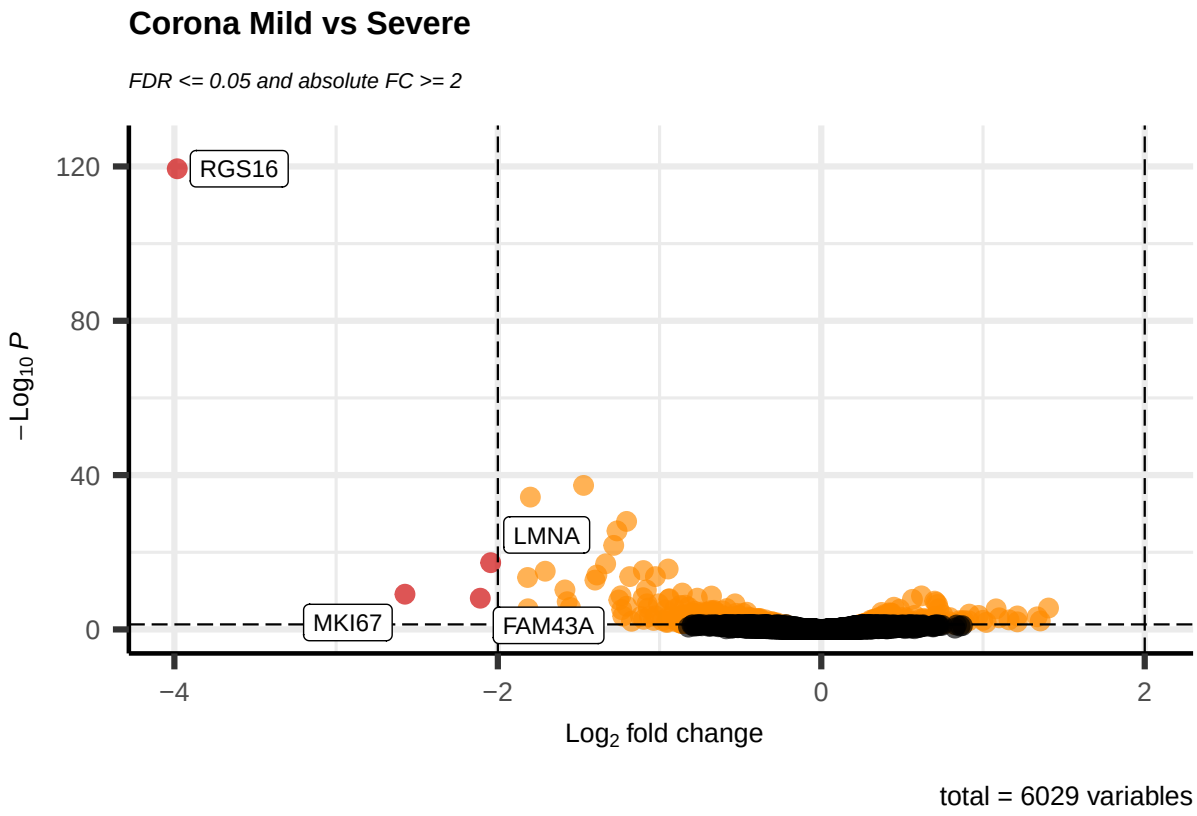
3.1.1 DESeq2

```
library(EnhancedVolcano)

## Simple function for plotting a Volcano plot, returns a ggplot object
EnhancedVolcano(res, x = 'log2FoldChange', y = 'padj',
                lab=rownames(res),
                title = paste("Corona", "Mild vs Severe"),
                subtitle = bquote(italic('FDR <= 0.05 and absolute FC >= 2')),
                # Change text and icon sizes
                labSize = 3, pointSize = 3, axisLabSize=10, titleLabSize=12,
                subtitleLabSize=8, captionLabSize=10,
                xlim = c(min(res$log2FoldChange) + 0.5, max(res$log2FoldChange) + 0.5, na.rm = T),
                # Disable legend
                legendPosition = "none",
                # Set cutoffs
                pCutoff = 0.05, FCcutoff = 2,
                boxedLabels = T,
                drawConnectors = T,
                widthConnectors = 1.0,
                min.segment.length =  1,
                col = c("black", "green", "darkorange", "red3"),
                colAlpha = 2/3)
```



3.1.2 EdgeR

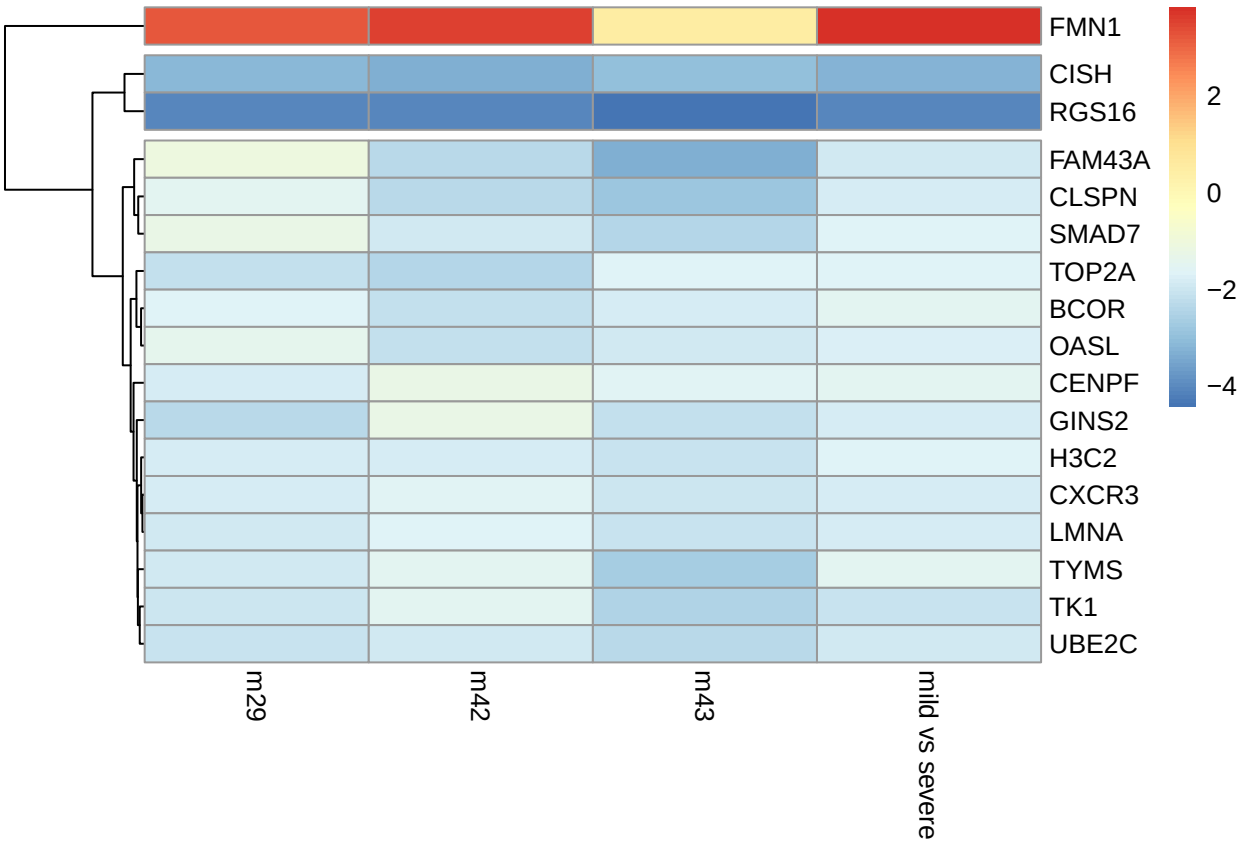
```
EnhancedVolcano(res.edger$table, x = 'logFC', y = 'FDR',
                lab=rownames(res.edger$table),
                title = paste("Corona", "Mild vs Severe"),
                subtitle = bquote(italic('FDR <= 0.05 and absolute FC >= 2')),
                # Change text and icon sizes
                labSize = 3, pointSize = 3, axisLabSize=10, titleLabSize=12,
                subtitleLabSize=8, captionLabSize=10,
                xlim = c(min(res$log2FoldChange) + 0.5, max(res$log2FoldChange) + 0.5, na.rm = T),
                # Disable legend
                legendPosition = "none",
                # Set cutoffs
                pCutoff = 0.05, FCcutoff = 2,
                boxedLabels = T,
                drawConnectors = T,
                widthConnectors = 1.0,
                min.segment.length =  1,
                col = c("black", "green", "darkorange", "red3"),
                colAlpha = 2/3)
```



3.2 Clustering

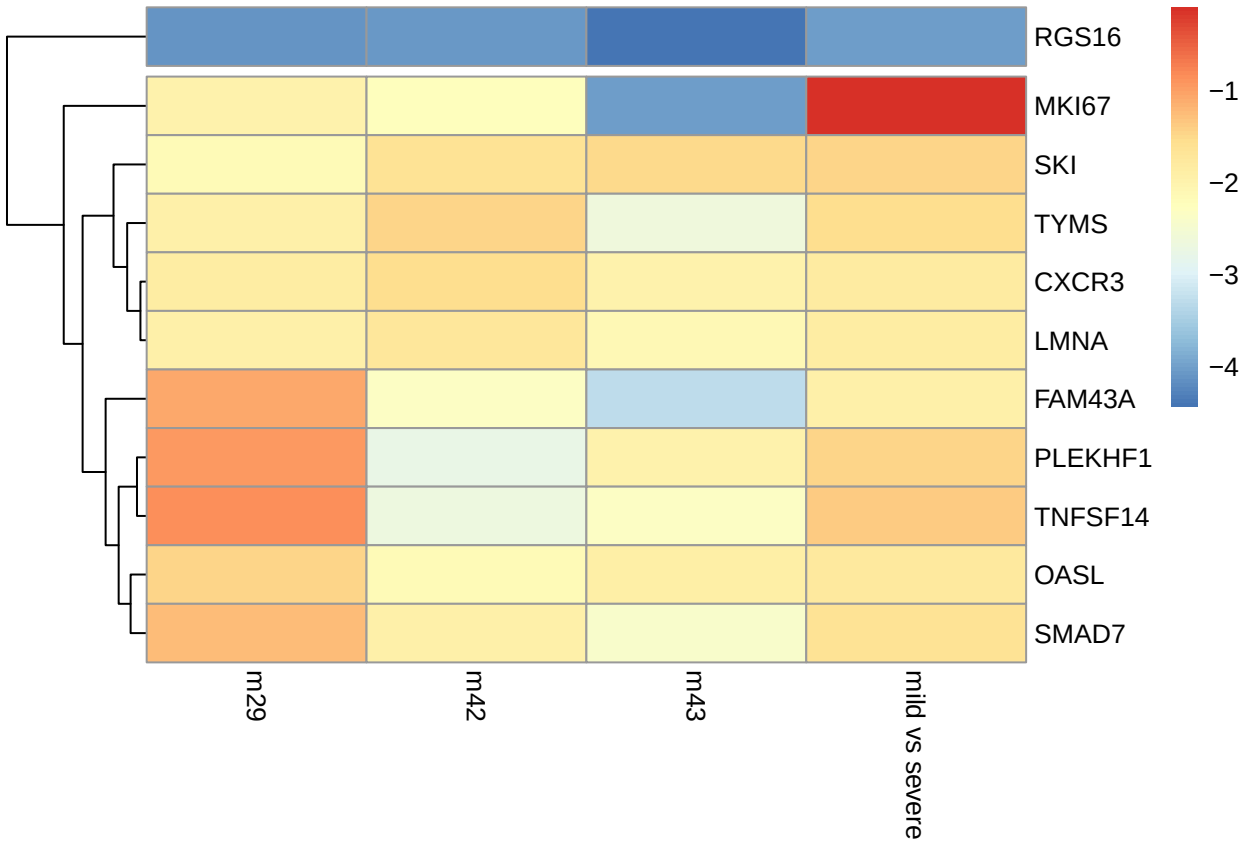
3.2.1 DESeq2

```
res.omit <- na.omit(res)
sig.genes <- res.omit[res.omit$padj <= 0.05 & abs(res.omit$log2FoldChange) > 1.5,]

expr <- data.frame(rowMeans(data.fpm[m29]) - data.fpm$s.mean,
                   rowMeans(data.fpm[m42]) - data.fpm$s.mean,
                   rowMeans(data.fpm[m43]) - data.fpm$s.mean,
                   res$log2FoldChange)
colnames(expr) <- c("m29", "m42", "m43", "mild vs severe")

pheatmap(expr[rownames(expr) %in% rownames(sig.genes),],
         cluster_cols = F,
         cutree_rows = 3)
```



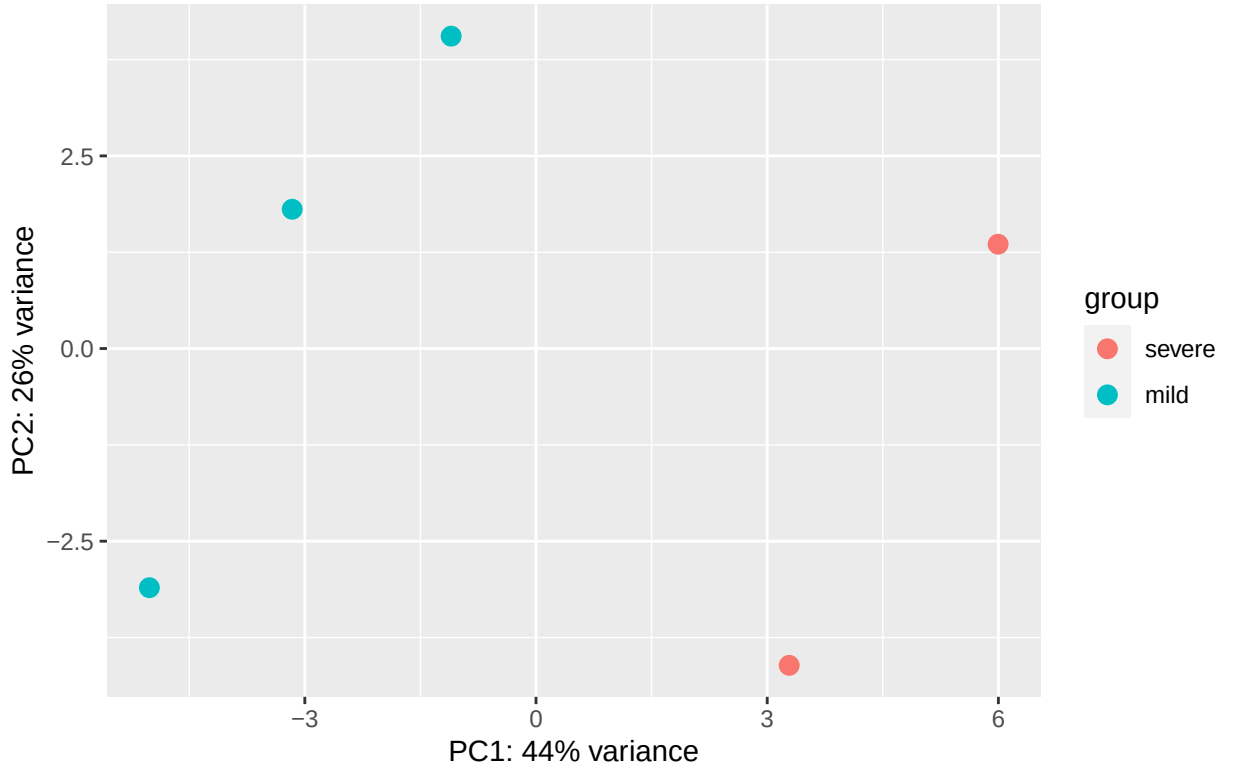
3.2.2 EdgeR

```
sig.genes <- res.edger[res.edger$table$FDR <= 0.05 & abs(res.edger$table$logFC) > 1.5,]

expr <- data.frame(rowMeans(data.fpm[m29]) - data.fpm$s.mean,
                   rowMeans(data.fpm[m42]) - data.fpm$s.mean,
                   rowMeans(data.fpm[m43]) - data.fpm$s.mean,
                   res$log2FoldChange)
colnames(expr) <- c("m29", "m42", "m43", "mild vs severe")

pheatmap(expr[rownames(expr) %in% rownames(sig.genes),],
         cluster_cols = F,
         cutree_rows = 2)
```

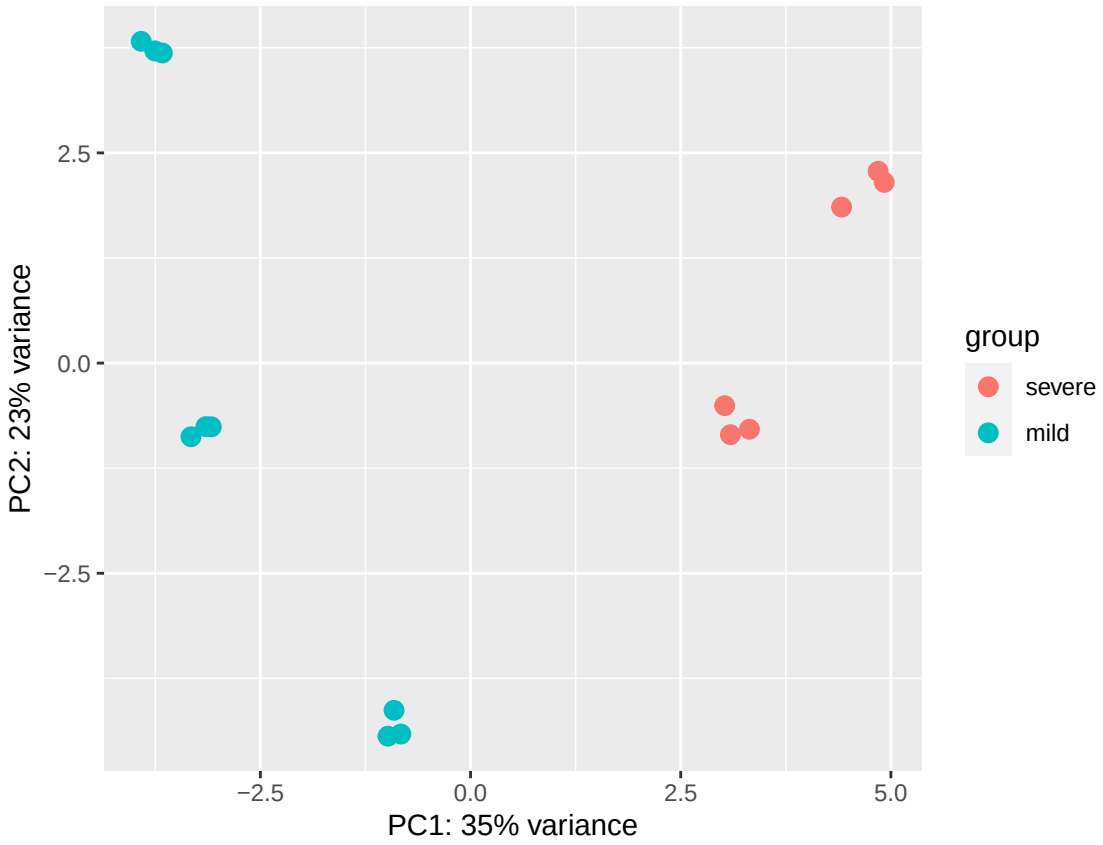


PCA

```
rlog_deseq <- rlog(dds.collapse)
plotPCA(rlog_deseq, intgroup="Condition")
```




```
edger.dds <- DESeqDataSetFromMatrix(dge$counts, colData = annotation, design = ~ Condition)
rlog_edger <- rlog(edger.dds)
plotPCA(rlog_edger, intgroup="Condition")
```


2.5
0.0
−2.5
PC2: 23% variance
−2.5
0.0
2.5
5.0
PC1: 35% variance
group
severe
mild