Macrophages' gene expression at myocardial infarction

Project for statistical assessment of a gene expression file with R studio

L T Stein

4/21/2023

Contents

Summary

The chosen article [3] compares their RNA-seq data set with that of another research. The genes are researched to see which experimental group is primarily involved in the reprogramming of macrofages behaviour towards a myocardial infarction.

The twelve provided samples with 24365 rows of genes and gene counts per sample are stored in a .tsv file and are divided into four groups; three experimental groups and one control group. The experimental groups are factors that stimulate the immune reaction of macrofages and mark the type of macrofage.

These stimulating factors are IL4, IFNg and PGE2. Groups IFNg and IL4 are cytokines. IFNg is used to mark and stimulate M1 macrofages, who are associated with a pro-inflammatory reaction and IL4 is used te indicate and stimulate M2 macrofages that are associated with an anti-inflammatory reaction.

On the contrary group PGE2 is a prostagladin, that in this context can stimulate the production of particular cytokines that either stimulate pro-inflammatory reaction or anti-inflammatory reaction of macrofages.

Furthermore the control group consists of data of untreated patients. De experimental data is obtained from fifty patients and the control data from twenty people.

The goal is to find Differentially Expressed Genes (DEGs), that are genes with a genuine significance and expression compared to the control group. While analysing the RNA-seq data, thus the gene counts different ways of normalizing the data are performed and explored. Also distinct plot methods are discussed and executed, such as the Volcano plot, MA plot, heatmap, normal distribution and a Venn diagram.

Introduction

In this project different approaches of handling gene count data are performed and explained. For example different methods for normalizing the data and different ways of plotting that data. The main goal in this project is finding genes from the chosen article that are marked as influencing or stimulating macrofages' immune response to myocardial infarction and see which experimental group contains most DEGs. This final step will be done using the Deseq2 library.

1. Loading data into R

The gene expression file in .tsv format is loaded with read.table(), since the data includes tab separated data *sep* = '\t' is specified. Also a header is included so that must be notified with *header* = *TRUE* . Finally three rows are printed using pander() and head().

```
count_data <- read.table(paste0("/homes/ltstein/Kwartaal_7/Thema_opdracht/GSE215801_exp_counts.tsv/",
                         "GSE215801_exp_counts.tsv"), sep = '\t', header = TRUE)
colnames(count_data) <- c("Untreated_1", "IL4-treated_1", "PGE2-treated_1",
                          "IFNg-treated_1", "Untreated_2", "IL4-treated_2",
                          "PGE2-treated_2", "IFNg-treated_2", "Untreated_3",
                          "IL4-treated_3", "PGE2-treated_3", "IFNg-treated_3")
```

Table 1: Table continues below

	Untreated_1	IL4-treated_1	PGE2-treated_1
ENSG00000223972.5	34	40	53

	IFNg-treated_1	Untreated_2
ENSG00000223972.5	97	1

Table 1.1: Expression data of gene counts from .tsv file

In the table above the expression data is displayed of the .tsv file *"GSE215801_exp_counts.tsv"* .
The x-axis is divided into twelve samples and three batches, test groups. Additionally on the y-axis all the gene names are displayed.

In the code block below dim() prints the dimensions of the dataset and str() the structure.

```
structure <- str(count_data)
```

```
## 'data.frame':    24365 obs. of  12 variables:
##  $ Untreated_1   : int  34 320 10 11 0 5 7 57 37 88 ...
##  $ IL4-treated_1 : int  40 788 16 17 7 1 12 59 65 119 ...
##  $ PGE2-treated_1: int  53 733 31 36 1 8 14 71 69 104 ...
##  $ IFNg-treated_1: int  97 1177 31 38 5 12 23 74 99 1051 ...
##  $ Untreated_2   : int  1 205 3 0 0 3 6 36 5 97 ...
##  $ IL4-treated_2 : int  4 506 1 0 0 4 0 76 6 95 ...
##  $ PGE2-treated_2: int  3 543 1 3 1 11 7 72 11 80 ...
##  $ IFNg-treated_2: int  2 328 0 0 0 0 3 32 16 139 ...
##  $ Untreated_3   : int  4 448 9 0 1 4 0 60 16 92 ...
##  $ IL4-treated_3 : int  36 2933 39 0 45 19 114 241 107 241 ...
##  $ PGE2-treated_3: int  42 496 15 38 1 1 11 59 20 107 ...
##  $ IFNg-treated_3: int  3 362 0 0 0 3 0 22 16 136 ...
```

```
dimension <- dim(count_data)
cat("dimensions =", dimension)
```

```
## dimensions = 24365 12
```

In the first code block above the structure and datatypes are displayed to confirm the data is correct . Also the second code block the dimensions of the x-as and y-axis are displayed from the dataset.

Case and control is assigned using indexing. This assigning makes a clearer difference between the experimental group and test group

```
case1 <- c(2,6,10)
case2 <- c(3,7,11)
case3 <- c(4,8,12)
control <- c(1,5,9)
```

In the code above case is assigned to the tested groups and control is assigned to the control group

2. Discovering Data Sets

2.1 Boxplot

Two boxplots are made of the data using the 'boxplot()' function. Since the raw data of the first boxplot shows a range which is too wide, it doesn't offer a clear view of the data. In the second boxplot a log-transformation is performed on the data using the 'log2()' function, that narrows down the range.

The replication groups are differentiated using the 'rep()' function. A vector with three colors is passed to rep, and rep is told that per four items the colors switch.

```
boxplot(count_data, las = 2, par(mar=c(8,8,4,2)), main = "Gene expression of twelve samples",
        xlab = "", ylab = "")
mtext("Samples", line = 0.7)
title(ylab = "Gene counts (n)", line = 5)
```

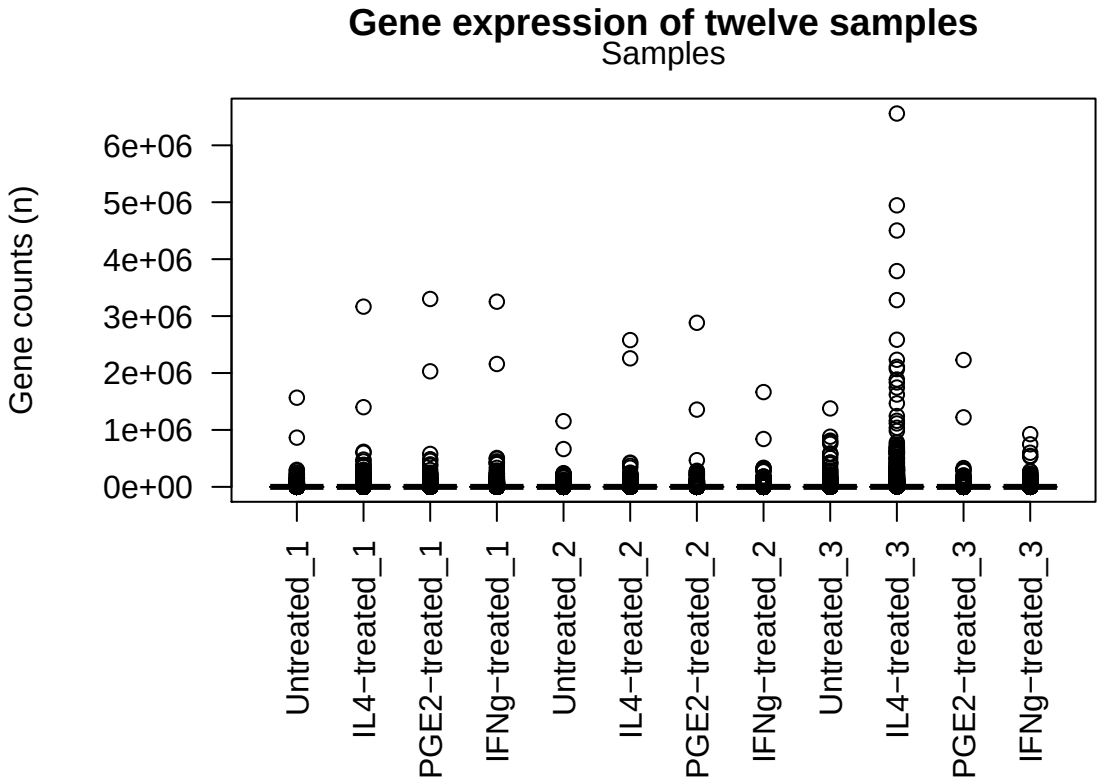

Figure 1: Boxplot of gene expression of twelve samples

In the figure above a boxplot is performed on the raw data. On the x-axis lie all the samples and on the y-axis are the gene counts.

```r
color_place_holder <- rep(c("orange", "cyan", "purple"), each = 4)


boxplot(log2(count_data + 1), las = 2, par(mar=c(8,8,4,2)),
        main = "Gene expression of twelve samples that are log-transformed",
        xlab = "", ylab = "", col= color_place_holder)
mtext("Samples", line = 0.7)
title(ylab = "Gene counts (n)", line = 5)
```

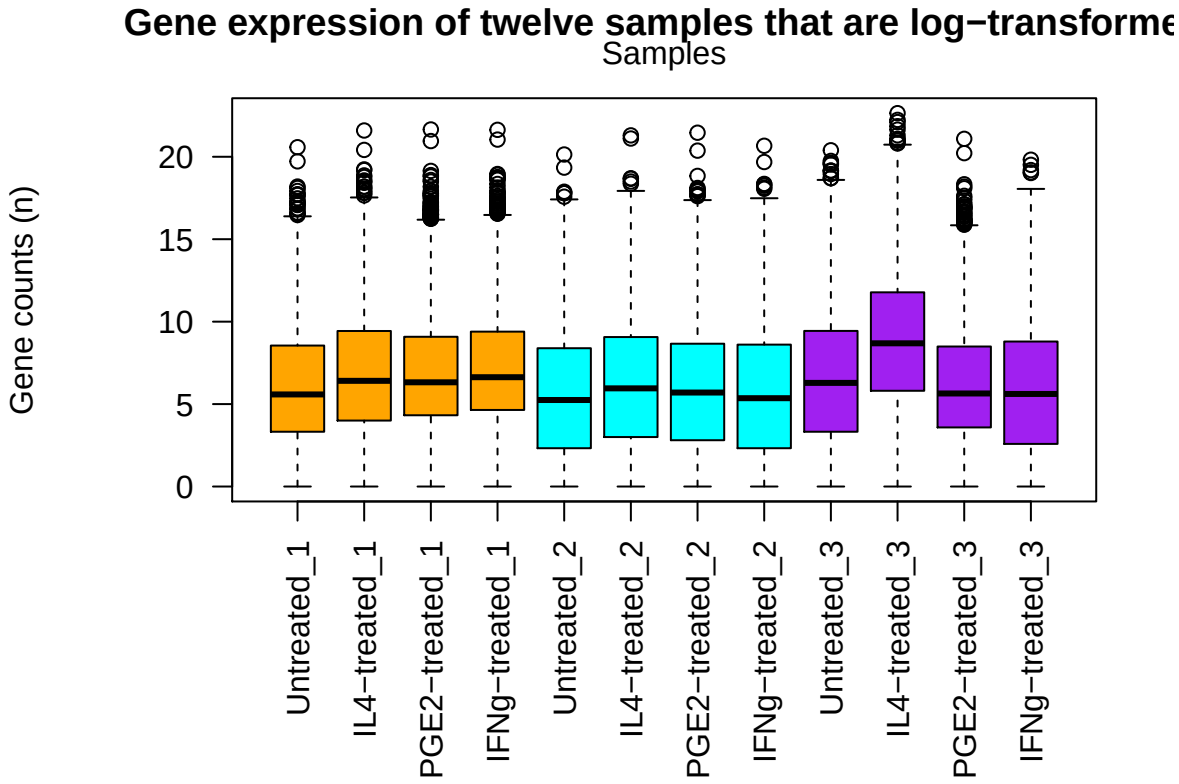

Figure 2: Boxplot of log-transformed gene expression of twelve samples

In the figure above a boxplot is performed on the raw data, using a log-transformation. On the x-axis lie all the samples and on the y-axis are the gene counts.

2.2 Density plot

The figure below is plotted using the 'plotDensity()' function from the ´affy' library. The legend is made with the 'legend()' function for all columns and the colors are specified using the same method as used in the boxplots above.

Finally a vertical line is added using 'abline()' to indicate that the values from the peak on its left are all inactive genes.

```
plotDensity(log2(count_data + 0.1), col = color_place_holder,
            main = "Density plot of gene expression data using log-transformation",
            xlab = "log2(count_data)")
legend('topright', names(count_data), lty=c(1:ncol(count_data)),
       col= color_place_holder, cex = 0.95)
abline(v=-1.5, lwd=1, col='red', lty=2)
```

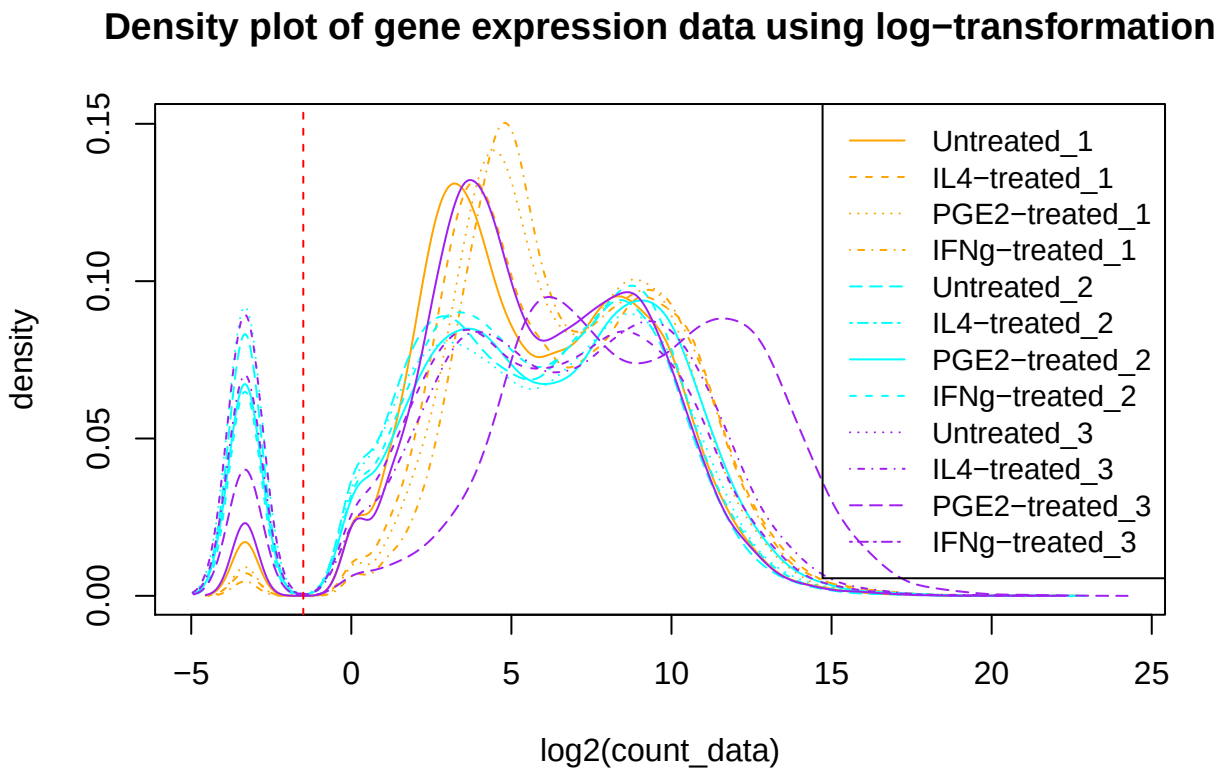

Figure 3: Density plot of gene expression data using log-transformation

In the figure above a density plot is made from the expression data. On the right side of the red line are all active genes and on its left side are therefore all inactive genes.

The replication group are divided into the same 3 colors as in the boxplots, and each sample received an individual line.

3. Exploratory Data Analysis

3.1 Normalization

```
par(mar=c(7,3,2,2))
barplot((colSums(count_data) / 1e6),
        ylab = "Sequencing depth (times million)",
        xlab  = "",
        main = "Expression read counts for GSE215801",
        col = color_place_holder,
        las = 2)
mtext("Samples", line = -0.5)
legend('topleft', names(count_data), lty=c(1:ncol(count_data)),
       col= color_place_holder, cex = 0.80)
```

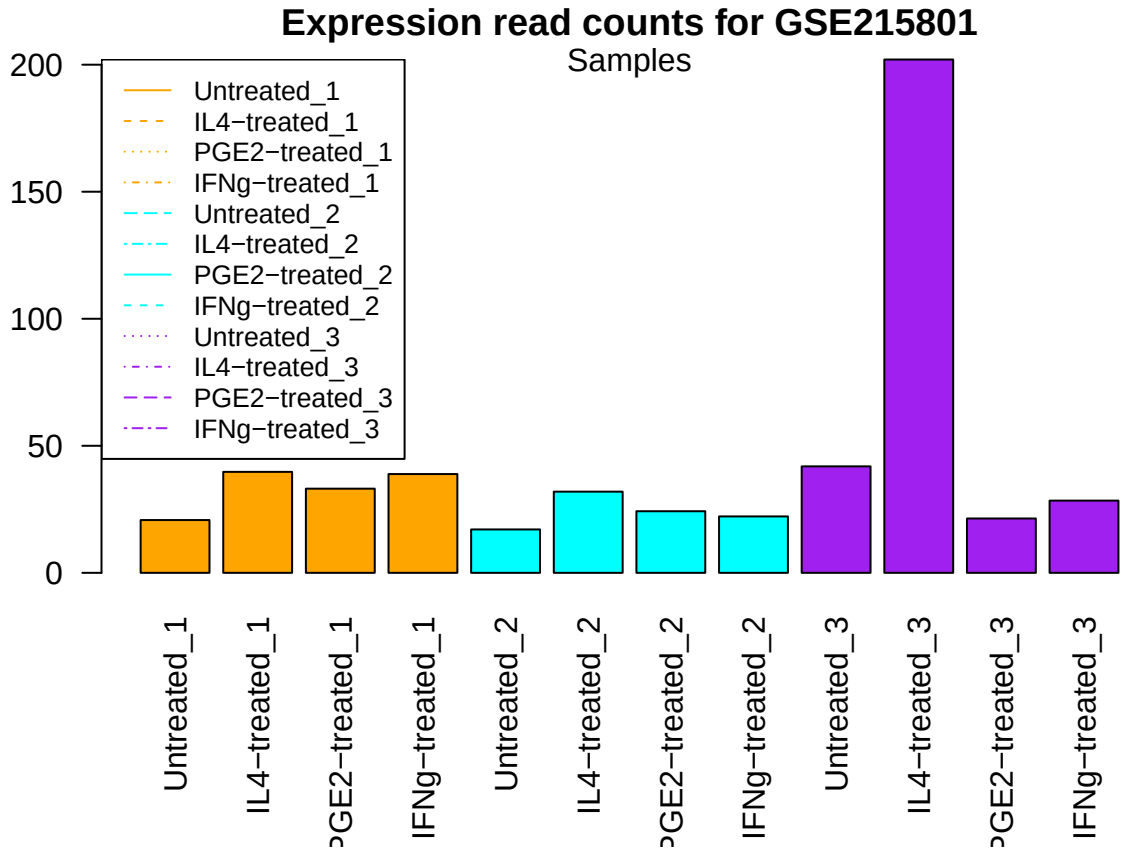

Figure 4: Expression read counts for all samples

In the figure above a barplot is made of the sequencing depth for each sample. On the x-axis are all samples with divided into three replication groups as shown with the colors orange

In the code block below a DESeqDataSet is made with the DESeqDataSetFromMatrix() function from the 'DESeq2' library

```
ddsMat <- DESeqDataSetFromMatrix(countData = count_data,
                                 colData = data.frame(samples = names(count_data)),
                                 design = ~ 1)
```

The data is normalized using the *variance stabilizing transformation* method. In R this method takes shape

in the vst() function with the *DESeqDataSetFromMatrix* as parameter. Later all the normalized values are extracted using the assay() function.

Table 3: Table continues below

	Untreated_1	IL4-treated_1	PGE2-treated_1
ENSG00000223972.5	6.436	6.087	6.478

Table 4: Table continues below

	IFNg-treated_1	Untreated_2	IL4-treated_2
ENSG00000223972.5	6.85	4.837	5.078

Table 5: Table continues below

	PGE2-treated_2	IFNg-treated_2	Untreated_3
ENSG00000223972.5	5.067	4.971	4.975

	IL4-treated_3	PGE2-treated_3	IFNg-treated_3
ENSG00000223972.5	5.149	6.676	5.023

Table 2: Normalized data of gene counts from .tsv file

In table 2 normalized count data is presented for all samples in one gene

3.2 Distance calculation

Distances between samples are calculated using the *dist()* function. The *t (transpose)* function within flips the data to a matrix format. After calculating all distances between samples that data format is converted to an actual matrix object using function *as.matrix()* . Finally the first row is tabelized.

```
# Calculating distances between samples
distance_data <- dist( t ( normalized.data  ))

# convert distance matrix data to an actual matrix object
DistMatrix <- as.matrix(distance_data)

# Tabelize group data
pander(DistMatrix[1:3, 1:4])
```

Table 7: Table continues below

	Untreated_1	IL4-treated_1	PGE2-treated_1
Untreated_1	0	72.51	89.01
IL4-treated_1	72.51	0	103.2
PGE2-treated_1	89.01	103.2	0

	IFNg-treated_1
Untreated_1	93.91
IL4-treated_1	96.5
PGE2-treated_1	99.87

Table 2.2: Normalized distance data between samples from .tsv file

In table 2.2 is a matrix shown for distances between all samples for one gene.

3.3 Sample distances using a Heatmap

```
# Get colnames for untreated an treated
Untreated <- c(1,5,9)
Treated <- c(2:4, 6:8, 10:12)
Treatment <- c(1:12)

Treatment[Untreated] <- 0
Treatment[Treated] <- 1
Treatment <- factor(Treatment, labels = c("Untreated", "Treated"))
Treated_genes <- factor(rep(c(0,1,2,3), 3),
                        labels = c("control", "IL4", "PGE2", "IFNg"))

annotation <- data.frame(Treatment, Treated_genes)
pander(annotation)
```

Treatment	Treated_genes
Untreated	control
Treated	IL4
Treated	PGE2
Treated	IFNg
Untreated	control
Treated	IL4
Treated	PGE2
Treated	IFNg
Untreated	control
Treated	IL4
Treated	PGE2
Treated	IFNg

```
rownames(annotation) <- names(count_data)

pheatmap(DistMatrix, show_colnames = FALSE,
         annotation_col = annotation,
         clustering_distance_rows = distance_data,
         clustering_distance_cols = distance_data,
         main = "Euclidean Sample Distances")
```

The figure below shows a heatmap of the annotated data as shown in the legend. On the x and y-axis dendogram show the distances between each sample. For example IL4-treated_1 all at the bottom doesn't seem to be similar to its group members IL4-treated_2 and IL4-treated_3

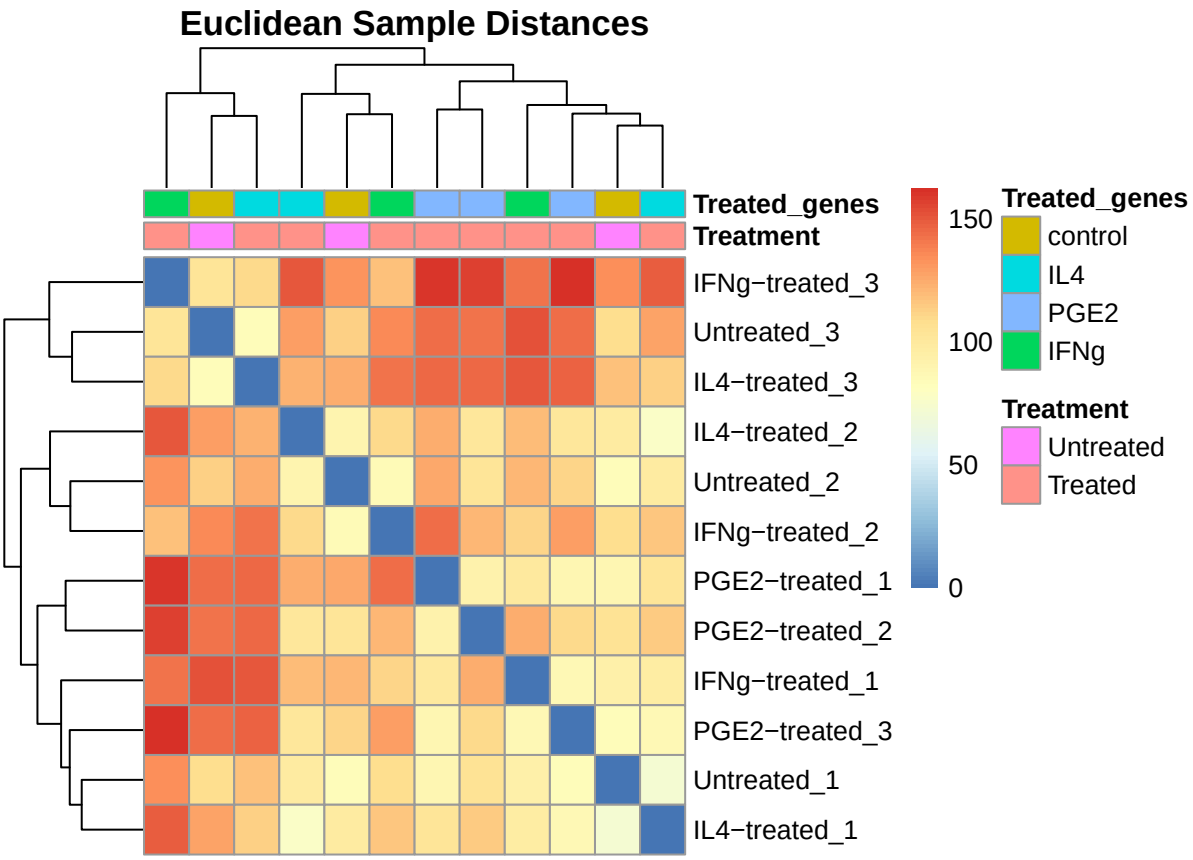

Figure 5: Euclidean distances between samples per group

3.4 Multi-Dimensional Scaling

In the code block below several steps are performed for normalizing the raw count data and calculating preferable distances between the samples called the Poisson Distance.

```
dds <- assay(ddsMat)
poisd <- PoissonDistance( t(dds), type = "deseq")
normalized.samplePoisDistMatrix <- as.matrix(poisd$dd)
mdsPoisData <- data.frame( cmdscale(normalized.samplePoisDistMatrix) )
names(mdsPoisData) <- c('x_coord', 'y_coord')
pander(mdsPoisData)
```

x_coord	y_coord
-924.3	1272
-2870	3710
-7857	2411
-9394	-4689
977	-16.49
-993.7	2650
-3556	1439
-260.4	-5532
8507	180.9
14457	4494
-5742	2102
7656	-8020

Table 3: X and Y coordinates of Poisson Distance.

In the figure below a graph is drawn showing the Poisson Distance between the samples. First a factor is made *factor()* for the three replication groups assigning unique names to show in the legend. Furthermore the names of the samples *names()* are assigned to coldata, as the names for the samples will show up instead of dots.

The plotting is done with *ggplot()* . The function offers a prettier and clearer graph and is constructed using more parameters. Such as the colnames from the raw data, the colors chosen arbitrarily based on the length of groups and the graph theme is adjusted by *theme_bw* .

```
groups <- factor(paste(annotation$Treatment, annotation$Treated_genes, sep = "_"))

coldata <- names(count_data)


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") +
  theme_bw()
```

In the figure below the samples of each group cluster. Though sample IL4-treated 3 deviates from this pattern. This might be due to large count data, that differs to much from its group samples neighbors.

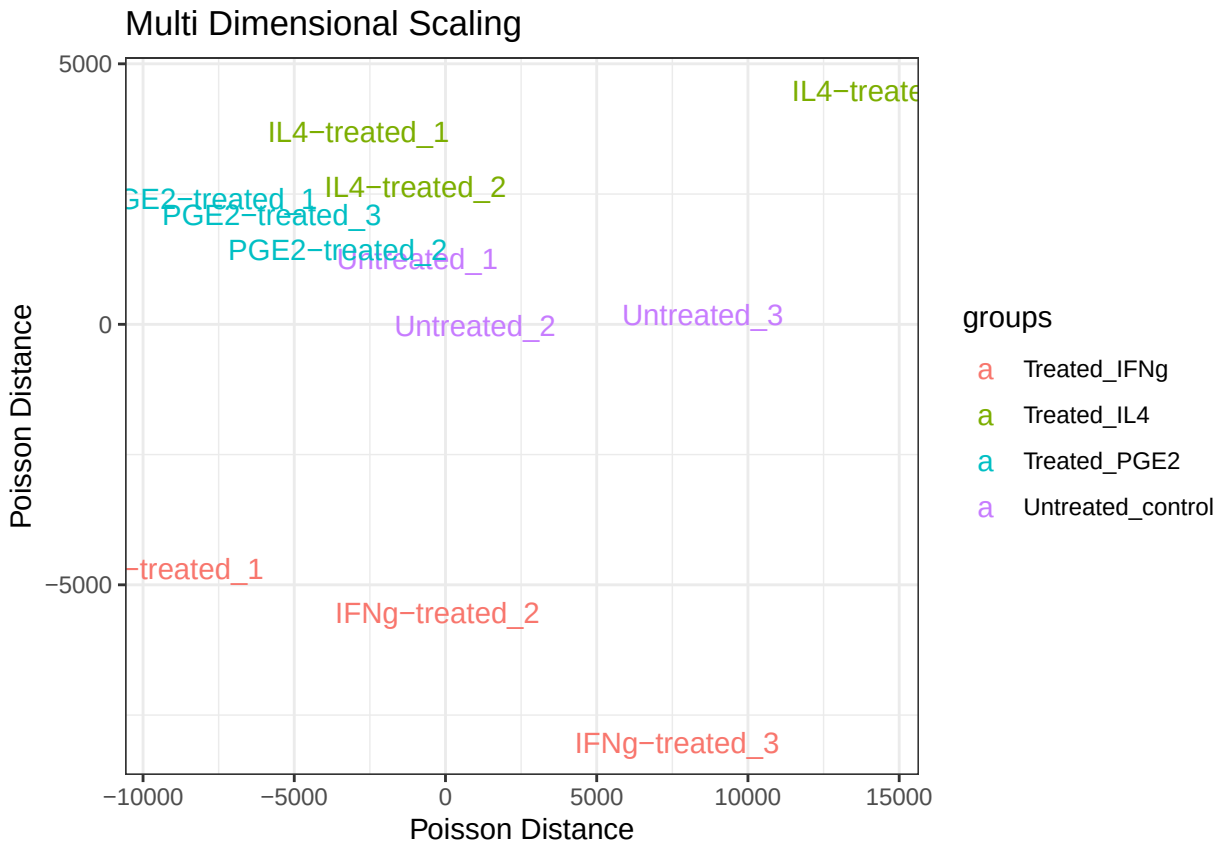

Figure 6: MDS graph showing the Poisson distance between samples

4. Discovering Differentialy Expressed Genes (DEGs)

4.1 Pre-processing

A different way of normalizing the data using the fragments per million mapped fragments (FPM) formula. As shown below the count data of the samples is divided by the sum of the count data divided by one million. The log2 is taken from the results plus one, as to remove zero values while normalization.

```
counts.fpm <- log2( (count_data / (colSums(count_data) / 1e6)) + 1 )
pander(counts.fpm[1:3, 1:5])
```

Table 11: Table continues below

	Untreated_1	IL4-treated_1	PGE2-treated_1
ENSG00000223972.5	1.399	1.171	1.798
ENSG00000227232.5	3.178	5.066	4.744
ENSG00000243485.5	0.3806	0.783	1.318

	IFNg-treated_1	Untreated_2
ENSG00000223972.5	1.805	0.03403
ENSG00000227232.5	6.128	1.011
ENSG00000243485.5	0.9783	0.1893

Table 4.1: The log2 of the gene count data as a pre-processing step for normalization

4.1 Assignment

In the code block below genes are kept if they hold a count higher than five for at least six samples. If this requirement is not met, then the gene (or row) will be filtered from the count data. In this case half of the samples (6) must meet the requirements.

```
# function that receives three parameters, the count_data object,
# a minimum count for a sample as threshold
# and n_samples for amount of samples that satisfy the threshold
normalize_data <- function(count_data, min_count, n_samples){
  keep <- rowSums(count_data >= min_count) >= n_samples
  kept_counts <- count_data[keep,]
  return(kept_counts)
}

kept_counts <- normalize_data(count_data, 5, 6)
filtered_data <- nrow(kept_counts)
original_gene_count <- nrow(count_data)
difference_filtered_count <- filtered_data - original_gene_count
cat("After the filtering step with minimum count 5 for at least 6 samples the new
filtered count is substracted from the original count:", original_gene_count, "-",
filtered_data, "=", difference_filtered_count, "genes")
```

```
## After the filtering step with minimum count 5 for at least 6 samples the new
## filtered count is substracted from the original count: 24365 - 21527 = -2838 genes
```

In the graph below a density plot is made with the same method as the density plot in figure three. Though a filtered dataset is passed from the function above.

```
plotDensity(log2(kept_counts + 0.1), col = groups,
            main = "Density plot of gene expression data using log-transformation",
            xlab = "log2(count_data)")
legend('topright', names(kept_counts), lty=c(1:ncol(kept_counts)),
       col= groups, cex = 0.95)
abline(v=-1.5, lwd=1, col='red', lty=2)
```

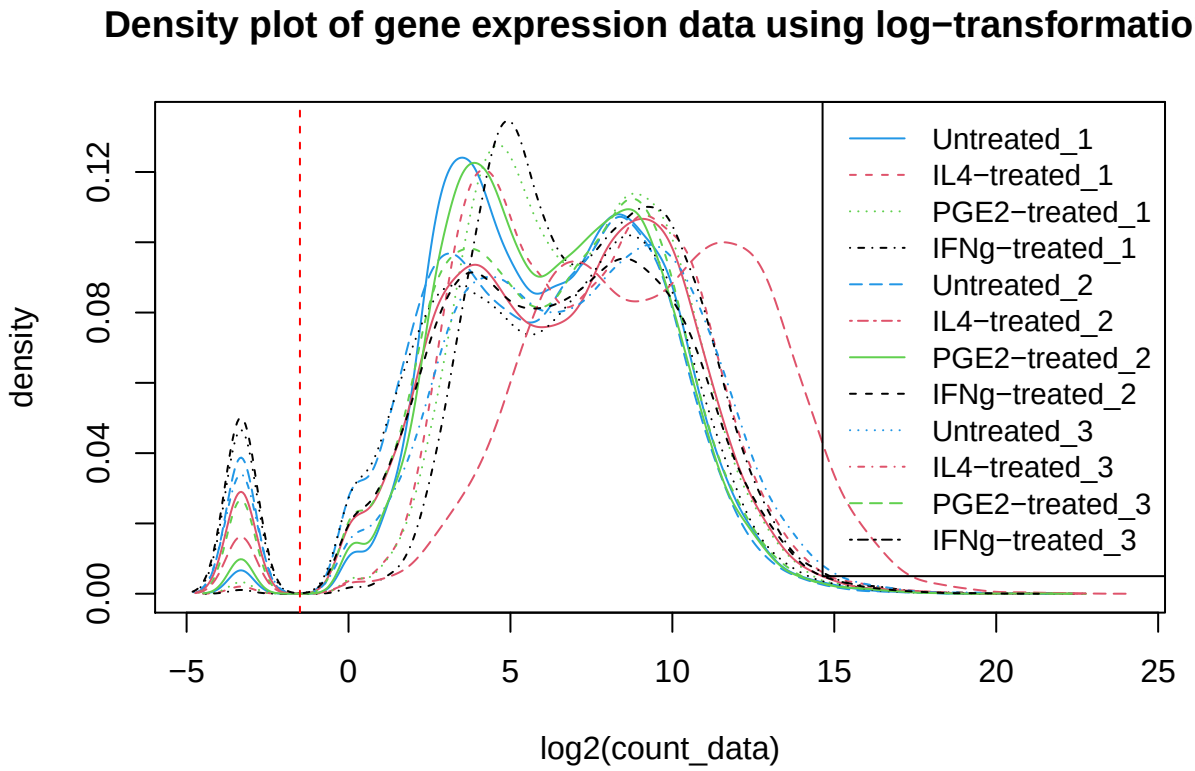

Figure 7: Density plot of gene expression data using count threshold and log-transformation

In the density plot above it becomes clear that genes have been removed from the y-axis that displays the density, as the y-max-value went from 0.15 to 0.12. Also a large amount of the non expressive genes seems to be filtered as well.

4.2 Fold Change

In the code block below the means are calculated for two groups (experiment and control) per gene using the *rowMeans()* function. Then the log fold change is calculated by substracting the control averagescompanying article showing differences in performance compared to other methods and packages. from the experiments averages.

Finally a histogram is made with breaks of forty for clearer visualisation of the deviation. Also an two vertical lines are added at -1 and 1 indicating low fold-change values in between the borders and vice versa.

```
# step 1
counts.fpm$experiment_1_avg <- rowMeans(counts.fpm[case2])
counts.fpm$control_avg <- rowMeans(counts.fpm[control])
LFC_column <- counts.fpm$experiment_1_avg - counts.fpm$control_avg
hist(LFC_column, breaks = 40, main = "Log fold change of group PGE2-treated",
     col = "orange")
abline(v = c(-1 ,1), lt = 2,  lwd = 2, col = "red")
```

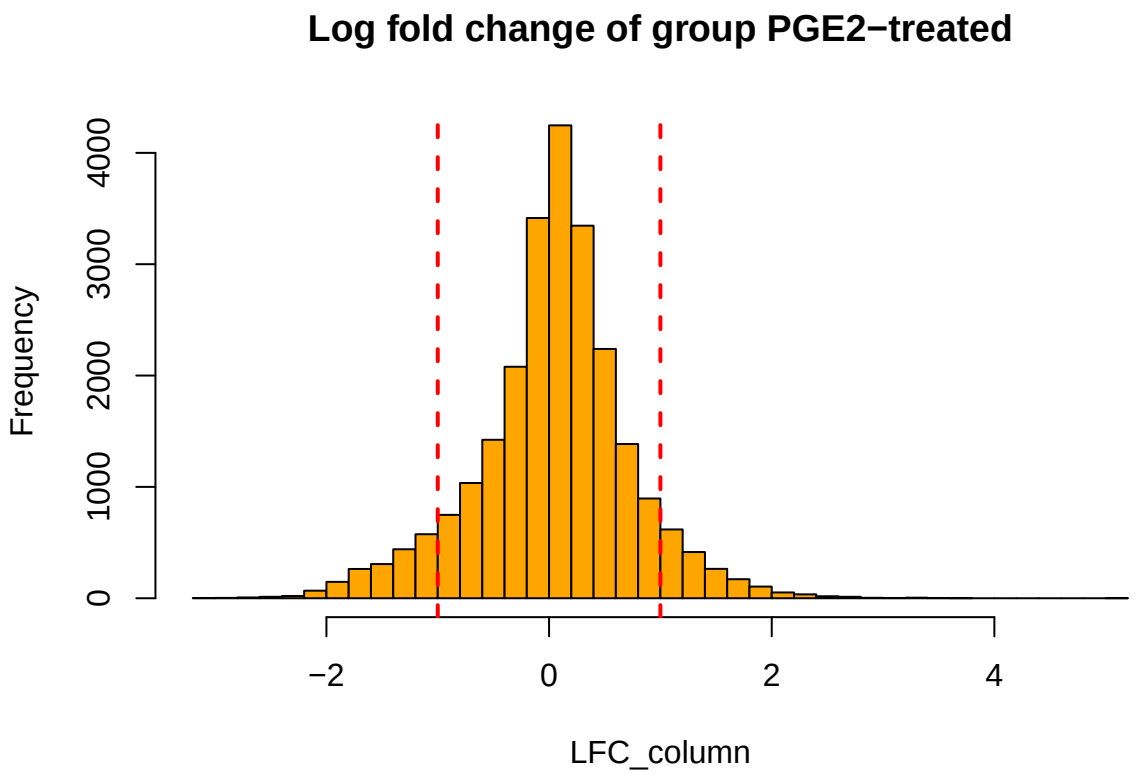

Figure 8: Log fold change of group PGE2-treated

In the figure 1.8 above a histogram is made of the log fold change values between two groups. A clear normal deviation is visible.

4.3 Using Bioconductor Packages

In the research's article it's mentioned that the t-test and chi-test are used for calculating the p-values.

4.3.1 The Design (matrix)

Here a model matrix is formed that can be used as the design in the ddsmatrix.

```
design.matrix <- model.matrix(~ Treated_genes, data=annotation)
pander(design.matrix)
```

Table 13: Table continues below

	(Intercept)	Treated_genesIL4	Treated_genesPGE2
Untreated_1	1	0	0
IL4-treated_1	1	1	0
PGE2-treated_1	1	0	1
IFNg-treated_1	1	0	0
Untreated_2	1	0	0
IL4-treated_2	1	1	0
PGE2-treated_2	1	0	1
IFNg-treated_2	1	0	0
Untreated_3	1	0	0
IL4-treated_3	1	1	0

	(Intercept)	Treated_genesIL4	Treated_genesPGE2
PGE2-treated_3	1	0	1
IFNg-treated_3	1	0	0

	Treated_genesIFNg
Untreated_1	0
IL4-treated_1	0
PGE2-treated_1	0
IFNg-treated_1	1
Untreated_2	0
IL4-treated_2	0
PGE2-treated_2	0
IFNg-treated_2	1
Untreated_3	0
IL4-treated_3	0
PGE2-treated_3	0
IFNg-treated_3	1

Table 4.2: A model matrix with all experiment groups against intercept (control)

4.3.2 DESeq2

Below a new ddsMat is set up with as main difference passing an factor with annotated data to 'colData' and a model.matrix is passed as design.

After setting up this matrix passed as an DESeqDataSet object and the names of the individual groups are extracted using the 'resultsNames()' function.

Finally these names can be passed to the 'results()' and the summary of each result object shows the amount of up or down regulated genes for that group, also an adjusted p-value, outliers, low counts and a mean count.

```
## using supplied model matrix
```

```
## estimating size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

```
## final dispersion estimates
```

```
## fitting model and testing
```

```
## [1] "Intercept"         "Treated_genesIL4"  "Treated_genesPGE2"
## [4] "Treated_genesIFNg"
```

A way for removing noise of log2 fold change data of low count genes is by getting the results from the *lfcShrink()* instead of the 'results()' function.

It's beneficial to do this since when plotting so that the log fold changes stand out.

```
Treated_genesIL4_shrink <- lfcShrink(dds, coef="Treated_genesIL4", type="apeglm")
```

```
FALSE using 'apeglm' for LFC shrinkage. If used in published research, please cite:
FALSE     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
```

```
FALSE     sequence count data: removing the noise and preserving large differences.
FALSE     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895
```

```
Treated_genesPGE2_shrink <- lfcShrink(dds, coef="Treated_genesPGE2", type="apeglm")
```

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

```
Treated_genesIFNg_shrink <- lfcShrink(dds, coef="Treated_genesIFNg", type="apeglm")
```

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

Below three MA plots are made with the 'plotMA()' function. The results per group from 'lfcShrink()' are passed to each plot. With ylim a broader or more narrow plot can be graphed. In case of these plots the ylimits are more stretched out to include more expression values into view.

```
par(mfrow = c(1,3))
plotMA(Treated_genesIL4_shrink, ylim=c(-5,10), main = "Log2 fold change of IL4")
plotMA(Treated_genesPGE2_shrink, ylim=c(-5,8), main = "Log2 fold change of PGE2")
plotMA(Treated_genesIFNg_shrink, ylim=c(-5,10), main = "Log2 fold change of IFNg")
```

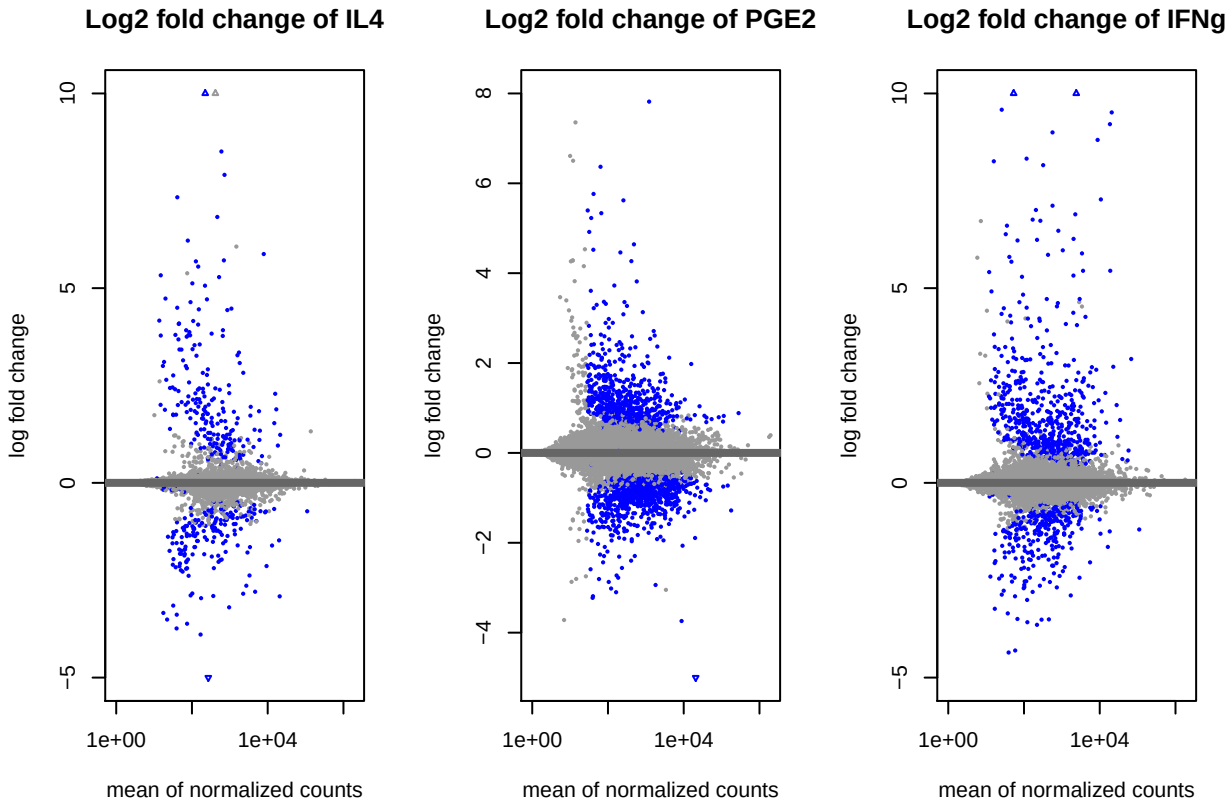

Figure 9: Log fold change of group PGE2-treated with shrinked low-count values

In the MA plots above are the log2 fold changes shown per treatment as the blue dots, instead the grey dots are noise.

On the y-axis the log fold change and on the x-axis the mean of the normalized counts. The further the points stray from the 0-axis the more they are expressed.

Genes (blue dots) are up regulated when above the 0-axis and down regulated when below.

In all plots most genes have a low up and down regulation and a few genes that are strongly up or down regulated.

5. Data analysis and visualisation

5.1 Volcano Plot

Below the function deseq.volcano receives shrunken datasets (exl. noise) and a parameter for the name of the group.

A dataset is passed to the *EnhancedVolcano()* function that is responsible for making the volcano plot.

Two important parameters are pCutoff and FCcutoff, since these can be set to the thresholds of the article in order to see comparable results.

Here $p < 0.001$ and the log2 fold change > 0.15.

Because of the strict thresholds lots of non expressive data is filtered out. Also it seems no genes suffice to the p-threshold of below 0.001 accompanying with a log2 fold change below its threshold. Though be aware these will appear as blue when a higher p-value e.g. 0.05 is set.

In the three figures below are on the y-axis the p-values, they indicate significance. On the x-axis log2 fold change is displayed, which indicates expressiveness from low < 0 (no expression) $>$ high.

```
deseq.volcano(res= Treated_genesIL4_shrink, datasetName = "IL4 ~ untreated")
```

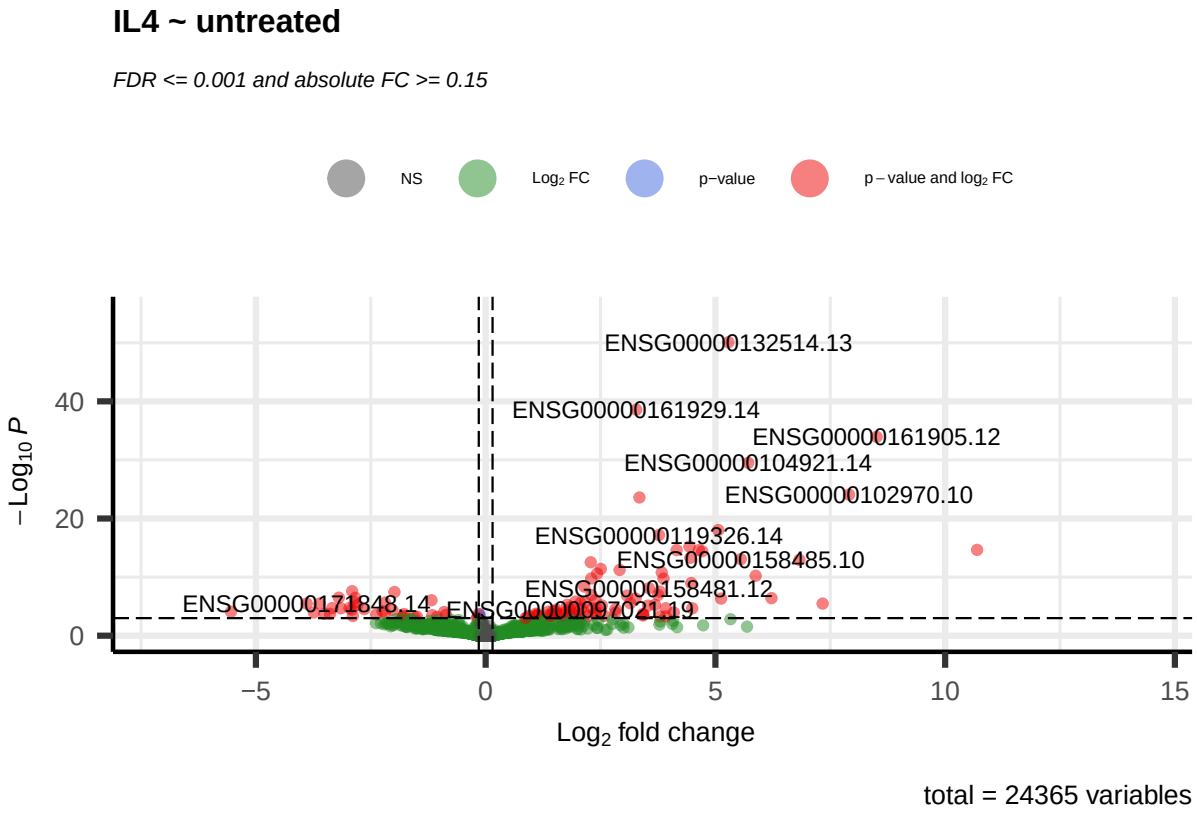

Figure 10: Volcano plot of group IL4

In the figure above of IL4 against untreated, more genes seem to be up regulated (on the right side) indicated as red dots.

Also the green dots show a higher log2 FC on the right. Though to be mentioned green dots don't meet the p-value threshold, thus are not changing significantly.

```
deseq.volcano(res= Treated_genesPGE2_shrink, datasetName = "PGE2 ~ untreated")
```

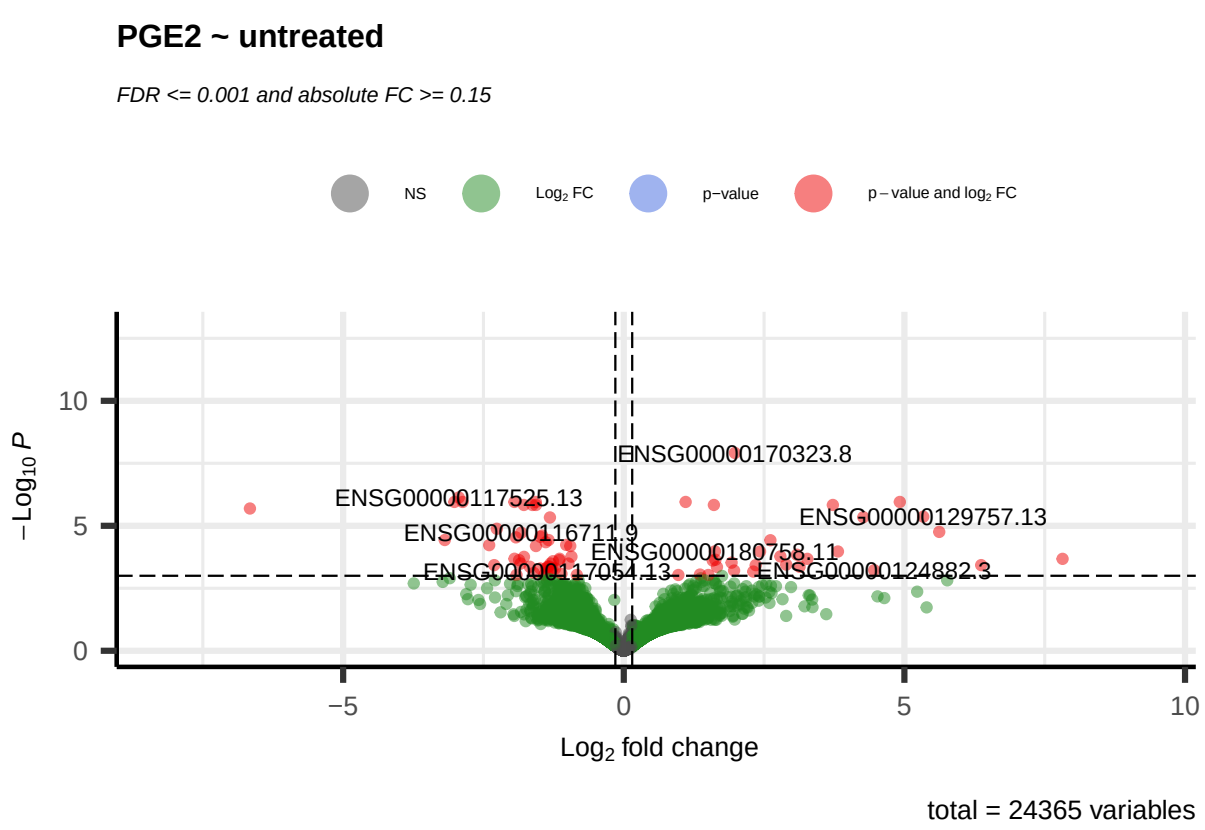

Figure 11: Volcano plot of group PGE2

In the figure above of PGE2 against untreated, less genes come to expression and the number of up and down regulated genes seem relatively equal.

Opposed to the previous plot of IL4 there's a larger variety in p-values and log2 FC in the green area.

```
deseq.volcano(res= Treated_genesIFNg_shrink, datasetName = "IFNg ~ untreated")
```

In the figure above of IFNg against untreated something really interesting is happening.

Way more genes come to expression with a larger quantity that are up regulated (red dots on rights side)

Also the p-values of the green dots show less variety, thus indicating for those genes a comparable significance. In addition the log2 FC values act more consistent.

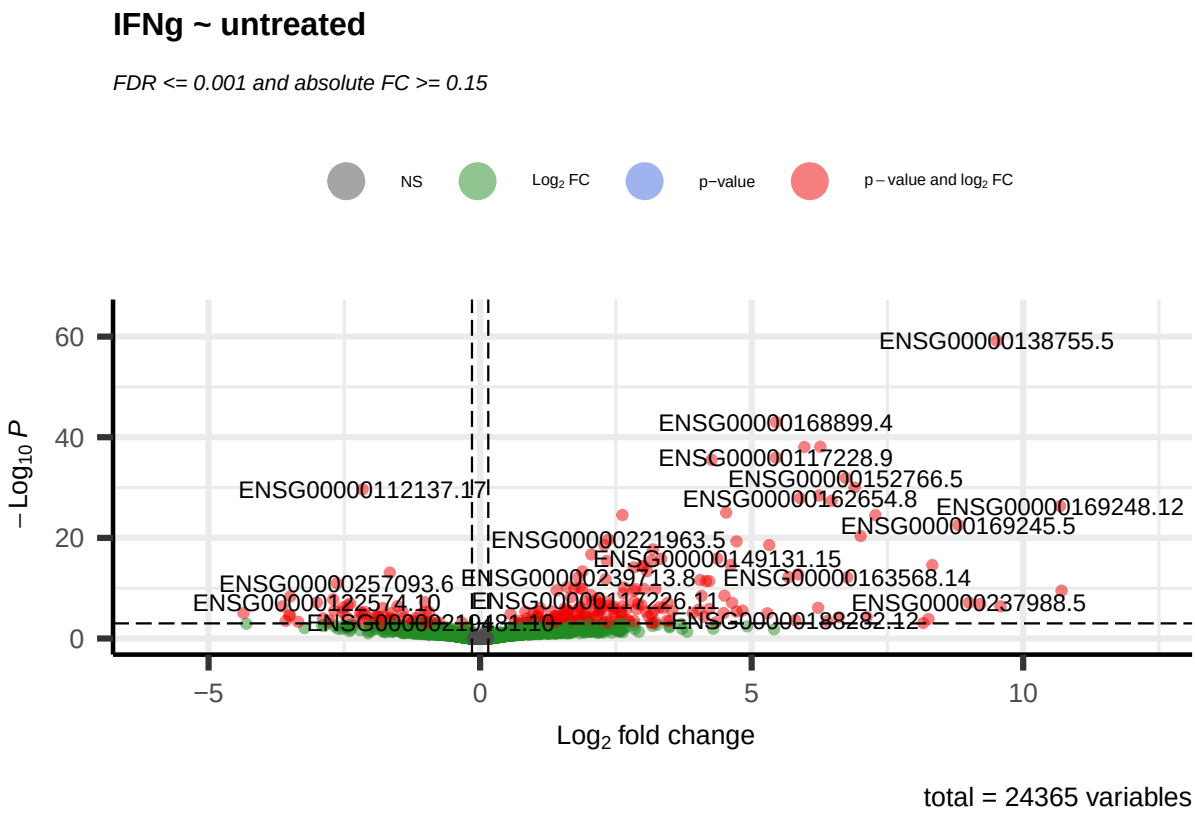

Figure 12: Volcano plot of group IFNg

5.2 Venn Diagram

In the chunk below all DEGs are extracted from the shrunk experiment groups' data at a p-value below 0.05.

```
## Data preparation
pval_threshold <- 0.05
IFNg.degs <- row.names(Treated_genesIFNg_shrink[which(Treated_genesIFNg_shrink$padj<= pval_threshold), ]
IL4.degs <- row.names(Treated_genesIL4_shrink[which(Treated_genesIL4_shrink$padj <= pval_threshold), ])
PGE2.degs <- row.names(Treated_genesPGE2_shrink[which(Treated_genesPGE2_shrink$padj <= pval_threshold),
```

Using the function *draw.triple.venn()* from the *VennDiagram* library, a Venn diagram is drawn.

The Venn diagram function takes the intersect of all experiment groups in duo's and one intersect between all experiment groups by intersecting one group with an intersection of a duo.

```
ins.1 <- length( intersect(IFNg.degs, IL4.degs) )
ins.2 <- length( intersect(IFNg.degs, PGE2.degs) )
ins.3 <- length( intersect(IL4.degs, PGE2.degs) )
triple.ins <- length( intersect(intersect(IL4.degs, PGE2.degs), IFNg.degs)  )
venn.plot <- draw.triple.venn(length(IFNg.degs),
                              length(IL4.degs),
                              length(PGE2.degs),
                              # Calculate the intersections of two sets
                                ins.1,
                                ins.2,
                                ins.3,
                                triple.ins,
                                category = c("IFNg.degs", "IL4.degs",
                                             "PGE2.degs"), scaled = F,
                                fill = c("red", "green", "cyan"), alpha = rep(0.5, 3),
                                cat.pos = c(0, 0, 180))
```

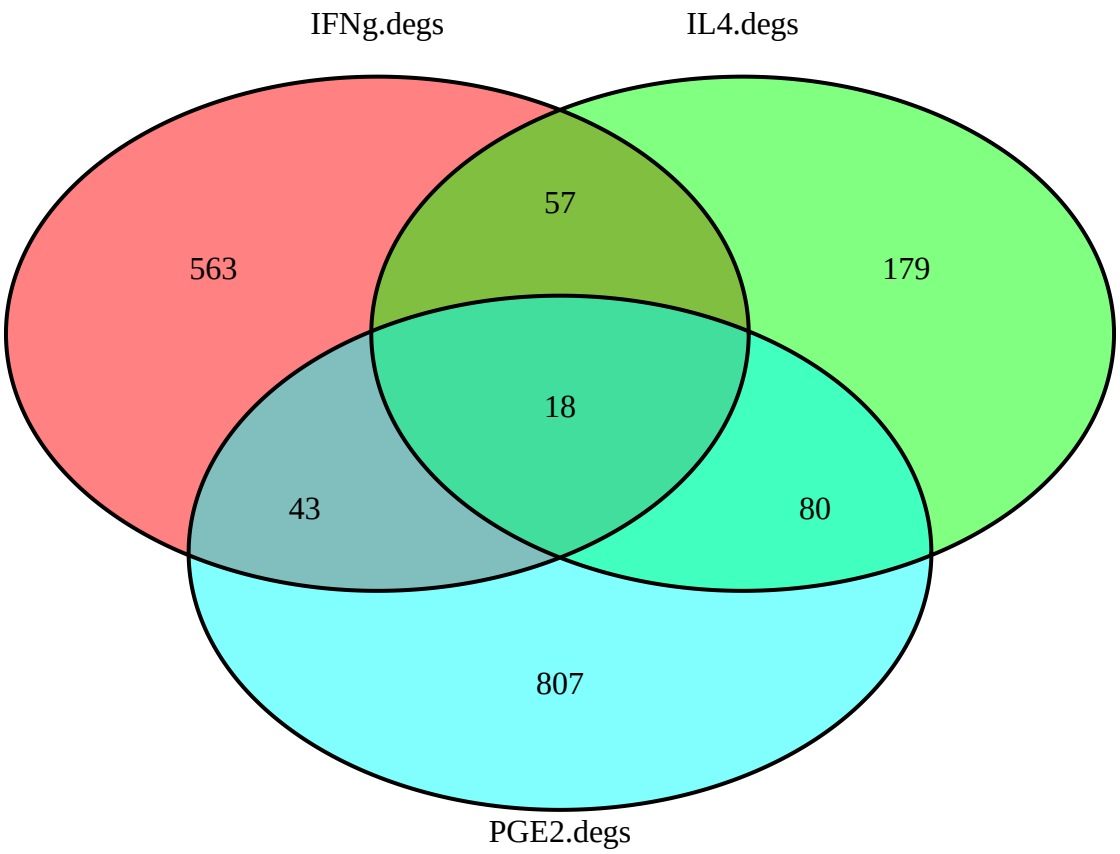

Figure 13: Venn diagram showing overlap between DEGs

5.3 Looking for DEGs from the article

Most DEGs found are annotated in the following way *ENSG00000159348* , this is an ENSEMBLE variant id. When comparing with other genes, it's preferred to have that same format. Though in the resulting DEGs, genes with id's such as *ENSG00000159348.2* exist. The .2 means a variant of the ENSEMBLE id.

Since the DEGS are annotated in this manner, all .etc must be removed. The function *tools::file_path_sans_ext* removes file extensions and is preferred in this case. Since this is an easy way to tackle the problem and instantly receive the wanted output without indexing on an object.

Using *sapply* every line of the DEG variables' data is passed to the function. Finally just to be sure of the correct data type, the data is made type *character* with the *as.character()* function

In the code chunk below all DEGs ensemble gene id's are extracted per group and written away, for future analysis like e.g. looking for biological processes the DEGs are correlated to.

```
IFNg.degs.symbol <- as.character(sapply(IFNg.degs, tools::file_path_sans_ext))
IL4.degs.symbol <- as.character(sapply(IL4.degs, tools::file_path_sans_ext))
PGE2.degs.symbol <- as.character(sapply(PGE2.degs, tools::file_path_sans_ext))
write.table(IFNg.degs.symbol, file = "data/IFNg-deseq-deg-names.txt",
            row.names = FALSE, quote = FALSE, col.names = FALSE)
write.table(IL4.degs.symbol, file = "data/IL4-deseq-deg-names.txt",
            row.names = FALSE, quote = FALSE, col.names = FALSE)
write.table(PGE2.degs.symbol, file = "data/PGE2-deseq-deg-names.txt",
            row.names = FALSE, quote = FALSE, col.names = FALSE)
```

Next the gene symbol list of *Table 2* from the article [3] is read so they can be compared with the DEGs per experiment group. Though those gene symbols have first been converted to ensemble gene id's using the webtool from SynGO [4].

Also the function *check_degs_contain_targets()* has been written for comparing DEGs between two data sets containing gene ensemble id's. The algorithm iterates over the size of parameter y and if that item is found in parameter x, a flag is raised of TRUE and the index of that item will be catted.

```
target_genes <- read.table("/homes/ltstein/Kwartaal_7/Thema_opdracht/Opdrachten/data/target_genes_artic
check_degs_contain_targets <- function(x, y) {
  size <- length(y)
  for (i in 1:size){
      if(x[i,] %in% y){
          flag <- TRUE
          cat(i, flag)
      }
  }
}
```

In total three genes from the article are found as is shown below

```
cat("Experiment group IFNg\n")
```

```
## Experiment group IFNg
```

```
check_degs_contain_targets(target_genes, IFNg.degs.symbol)
```

```
## 5 TRUE35 TRUE
```

```
# Using square brackets indexing and a vector multiple indices can be
# passed to receive several gene id's at a time. An example below:

target_genes$V1[c(5, 35)]
```

```
## [1] "ENSG00000159348" "ENSG00000148848"
```

```
cat("\nExperiment group PGE2\n")
```

```
##
## Experiment group PGE2
```

```
check_degs_contain_targets(target_genes, PGE2.degs.symbol)
```

```
## 32 TRUE
```

```
target_genes$V1[c(32)]
```

```
## [1] "ENSG00000165138"
```

```
cat("\nExperiment group IL4\n")
```

```
##
## Experiment group IL4
```

```
check_degs_contain_targets(target_genes, IL4.degs.symbol)

cat("No corresponding DEGs found\n")
```

```
## No corresponding DEGs found
```

The last step is converting these gene id's back to gene symbols. Again the webtool SynGO [4] is used for this. For group IFNg the two genes are called CYB5R1 en ADAM12 and group PGE2 contains gene ANKS6. For a final check are the gene symbol searched for within *Table 2* from the article [3] and are indeed present.

Conclusion and discussion

All things considered in total 1963 significant DEGs were found with a core overlap of 18 genes as shown in the Venn diagram. Furthermore within experiment group IFNg two genes were found that overlapped with the genes from the article, in experiment group PGE2 one gene was found and group IL4 had none to be found.

Thus after normalization from the total of 24365 genes with thresholds of a p-value < 0.05 and FC $>= 2$ 24365 minus 1963 equals to 22.402 genes have not passed as differentially expressed genes.

Also per experiment group the following amount of DEGs were determined PGE2 (n=948), INFg (n=681) and IL4 (n=334). This means that experimental group Prostaglandin E2 (PGE2) expressed the most DEGs.

Unfortunately since the original article had combined its RNA-seq data with another micro-array data set, the reproducibility was disappointing. The original plan of approach was to obtain similar plots by using the corresponding parameters from the article, however now that intention has changed. In the future a more disciplined start of researching is therefore sought.

Bibliography

1. Fontaine, M. A. C., Jin, H., Gagliardi, M., Rousch, M., Wijnands, E., Stoll, M., Li, X., Schurgers, L., Reutelingsperger, C., Schalkwijk, C., van den Akker, N. M. S., Molin, D. G. M., Gullestad, L., Eritsland, J., Hoffman, P., Skjelland, M., Andersen, G. Ø., Aukrust, P., Karel, J., ... Biessen, E. A. (2022). Blood Milieu in Acute Myocardial Infarction Reprograms Human Macrophages for Trauma Repair. Accessed at 18-04-2023. The link: https://doi.org/10.1002/advs.202203053

2. GEO Browser. Accessed at 14-02-2023. The link: https://www.ncbi.nlm.nih.gov/geo/browse/.

3. Geo Browser. Blood milieu in acute myocardial infarction reprograms human macrophages for trauma repair. Accessed at 14-02-2023. De link: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE215801

4. SYNGO. SYNGO - ID conversion tool. Accessed at 20-03-2023. The link: https://www.syngoportal.org/convert