DataAnalysis

Wisse Schuuring

2/15/2022

```
knitr::opts_chunk$set(cache=TRUE)

# Read in all libraries used in the EDA.
library(ggplot2)
library(dplyr)
library(xtable)
library(PoiClaClu)
library(affy)
library(scales)
library(pheatmap)
library(DESeq2)
library(pander)
library(edgeR)
library(VennDiagram)
library(EnhancedVolcano)
```

Exploratory Data Analysis

Before utilizing the data set, it must be assembled and cleaned if necessary.

```
#reading in the parental sample files.
sample1P <- read.csv(file = "Data/SamplesTSV/GSM3733674_sample1.tsv",
                       sep = "\t", header=T, skip = 3)
sample4P <- read.csv(file = "Data/SamplesTSV/GSM3733677_sample4.tsv",
                       sep = "\t", header=T, skip = 3)
sample5P <- read.csv(file = "Data/SamplesTSV/GSM3733678_sample5.tsv",
                       sep = "\t", header=T, skip = 3)

#reading in the IRF2 knockout files.
sample2KO <- read.csv(file = "Data/SamplesTSV/GSM3733675_sample2.tsv",
                        sep = "\t", header=T, skip = 3)
sample3KO <- read.csv(file = "Data/SamplesTSV/GSM3733676_sample3.tsv",
                        sep = "\t", header=T, skip = 3)
sample6KO <- read.csv(file = "Data/SamplesTSV/GSM3733679_sample6.tsv",
                        sep = "\t", header=T, skip = 3)


#removing the "values" column in both parental and knockout files.
s1P <- sample1P[c("ID_REF", "count")]
s4P <- sample4P[c("ID_REF", "count")]
s5P <- sample5P[c("ID_REF", "count")]
```

```r
s2K <- sample2KO[c("ID_REF", "count")]
s3K <- sample3KO[c("ID_REF", "count")]
s6K <- sample6KO[c("ID_REF", "count")]
```

Read in the files and removed the unnecessary value value, for the only ones of interest are the id and the counts value.

```r
# Merge the parental data
firstMergeP <- merge(s1P, s4P, by="ID_REF")
finalMergeP <- merge(firstMergeP, s5P, by="ID_REF")
names(finalMergeP)[2] <- c("counts_1_p")
names(finalMergeP)[3] <- c("counts_4_p")
names(finalMergeP)[4] <- c("counts_5_p")


# Merge the IRF2KO data
firstMergeIRF2KO <- merge(s2K, s3K, by="ID_REF")
finalMergeIRF2KO <- merge(firstMergeIRF2KO, s6K, by="ID_REF")
names(finalMergeIRF2KO)[2] <- c("counts_2_KO")
names(finalMergeIRF2KO)[3] <- c("counts_3_KO")
names(finalMergeIRF2KO)[4] <- c("counts_6_KO")


# Merge both datasets into a completed dataset
myData <- merge(finalMergeP, finalMergeIRF2KO, by="ID_REF")

# Write the merged dataset to a seperate file " myData.csv".
write.csv(myData,file = 'Data/Samples/Reformat/myData.csv', row.names=F)
```

With the data cleaned up, a simple read.table() shall reveal its contents.

Reading in the Data

```r
# Reading the first five rows of myData.csv file into R as a table.
read.table(file="Data/Samples/Reformat/myData.csv", sep=",",
           header=T, row.names=1, nrows = 5)
```

```
##       counts_1_p counts_4_p counts_5_p counts_2_KO counts_3_KO counts_6_KO
## 11287          0          0          0           3           0           1
## 11298          4          1          0           0           2           1
## 11302       1482       1404       1715        1329        1383        1527
## 11303         34         29         23          16          17          23
## 11304          0          0          0           0           0           0
```

Showing the first five rows of the myData file, shown to contain the raw count data for the control case samples 1, 4 and 5, as well as the test case samples 2, 3 and 6.

Data Dimensions

The data set contains the identifier for the gene, followed by the amount of times this gene has been expressed within it's respective sample. In addition, there are "total" columns with the sum of these counts for the IRF2 Knockout samples, and the parental samples. These can be used for comparison's sake. All of these values are integers, therefore easy to work with.

```r
# Reading the data set as myData
myData <- read.csv(file = "Data/Samples/Reformat/myData.csv", header = T, row.names = 1)
```

```
#Show the amount of rows and columns in myData
dimensions <- dim(myData)
cat("The data set contains", dimensions[1],
    "rows and",dimensions[2],"columns.\n")
```

```
## The data set contains 36528 rows and 6 columns.
```

```
#Show the structure of the data set.
pander(str(myData))
```

'data.frame': 36528 obs. of 6 variables: $ counts_1_p : int 0 4 1482 34 0 6951 4425 1552 4227 3946 … $ counts_4_p : int 0 1 1404 29 0 6712 3479 1357 3539 3873 … $ counts_5_p : int 0 0 1715 23 0 7791 4089 1603 4010 4536 … $ counts_2_KO: int 3 0 1329 16 0 5334 3374 2123 3451 3294 … $ counts_3_KO: int 0 2 1383 17 0 5036 2870 2086 3219 3172 … $ counts_6_KO: int 1 1 1527 23 0 5986 3614 2529 3741 3710 …

```
cat("All values within the data set are integers,
    as expected from a numeric value such as a count.\n")
```

```
## All values within the data set are integers,
##     as expected from a numeric value such as a count.
```

Summarising and Boxplot

```
# Summarise the Data
pander(summary(myData))
```

Table 1: Table continues below

counts_1_p	counts_4_p	counts_5_p	counts_2_KO
Min. : 0.0 1st Qu.: 0.0 Median : 0.0 Mean : 1068.0 3rd Qu.: 356.2 Max. :165702.0	Min. : 0 1st Qu.: 0 Median : 0 Mean : 915 3rd Qu.: 306 Max. :143468	Min. : 0 1st Qu.: 0 Median : 0 Mean : 1093 3rd Qu.: 367 Max. :171942	Min. : 0.0 1st Qu.: 0.0 Median : 0.0 Mean : 821.1 3rd Qu.: 287.0 Max. :122478.0

counts_3_KO	counts_6_KO
Min. : 0.0 1st Qu.: 0.0 Median : 0.0 Mean : 795.8 3rd Qu.: 277.0 Max. :133071.0	Min. : 0.0 1st Qu.: 0.0 Median : 0.0 Mean : 932.2 3rd Qu.: 326.0 Max. :144885.0

```
# Create a boxplot displaying the logged data of the six samples.
boxplot(log2(myData + 1),
        data=myData, main="Gene expression between Parental and IRF2 KO",
        xlab="Gene Samples",
        ylab="Log of Gene expressions",
        col = c("orange","orange",
                "orange","dark Blue",
                "dark Blue","dark Blue"),
        las=2, par(cex.axis=0.46),
        names=c("Parent 1","Parent 2",
                "Parent 3","Knockout 1",
                "Knockout 2", "Knockout 3"))
```

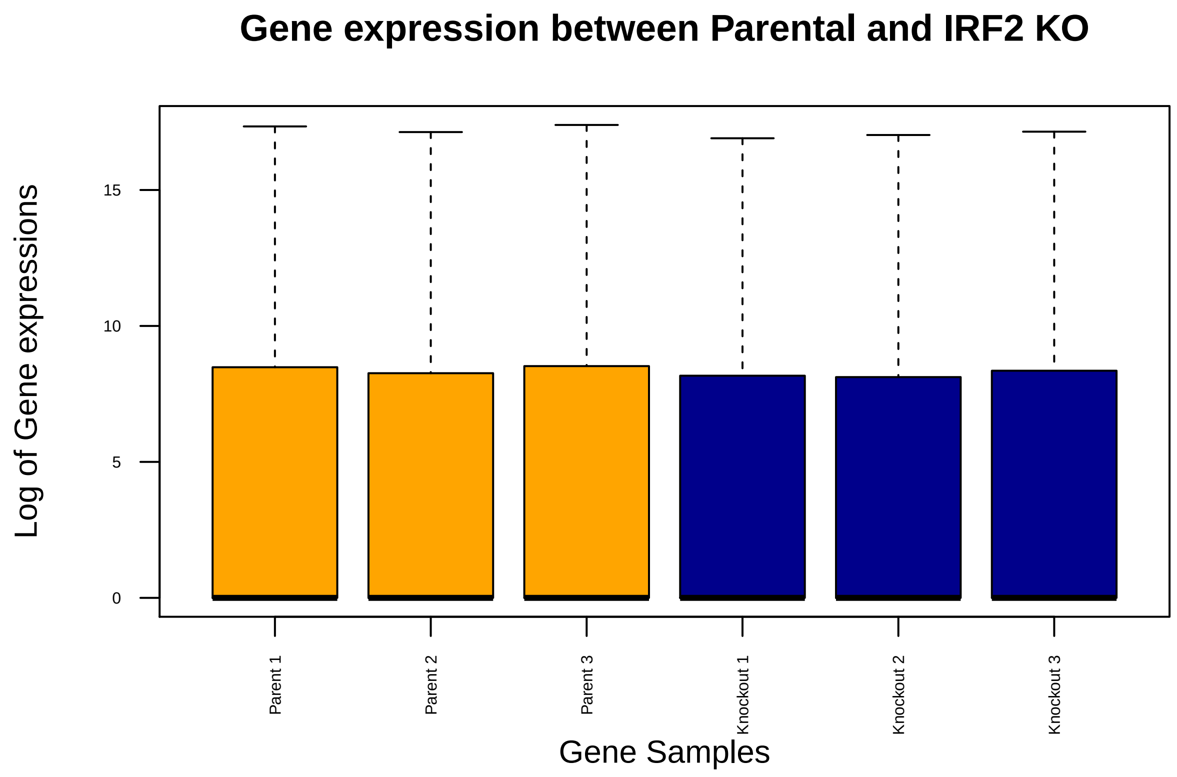

Figure 1: Boxplot showing the log value of the gene counts are balanced as desired.

A summary shows that most data points do indeed equal zero. The following boxplot reveals however that the data samples overall are very balanced, as they should be, and can all be used for the research.

Density Plot

```
## Create a vector of 2 colors to use.
myColors <- c("Orange","dark Blue")

## Plot the log2-transformed data with a 0.1 pseudocount
plotDensity(log2(myData + 0.1), col=rep(myColors, each=3),
            lty=c(1:ncol(myData)), xlab='Log2(count)',
            main='Expression Distribution')

## Add a legend and vertical line
legend('topright', names(myData), lty=c(1:ncol(myData)),
       col=rep(myColors, each=3))
abline(v=-1.5, lwd=1, col='red', lty=2)
```

Considering there are barely to no shifts between the peaks of the different samples, it can be concluded that the data is of high quality.

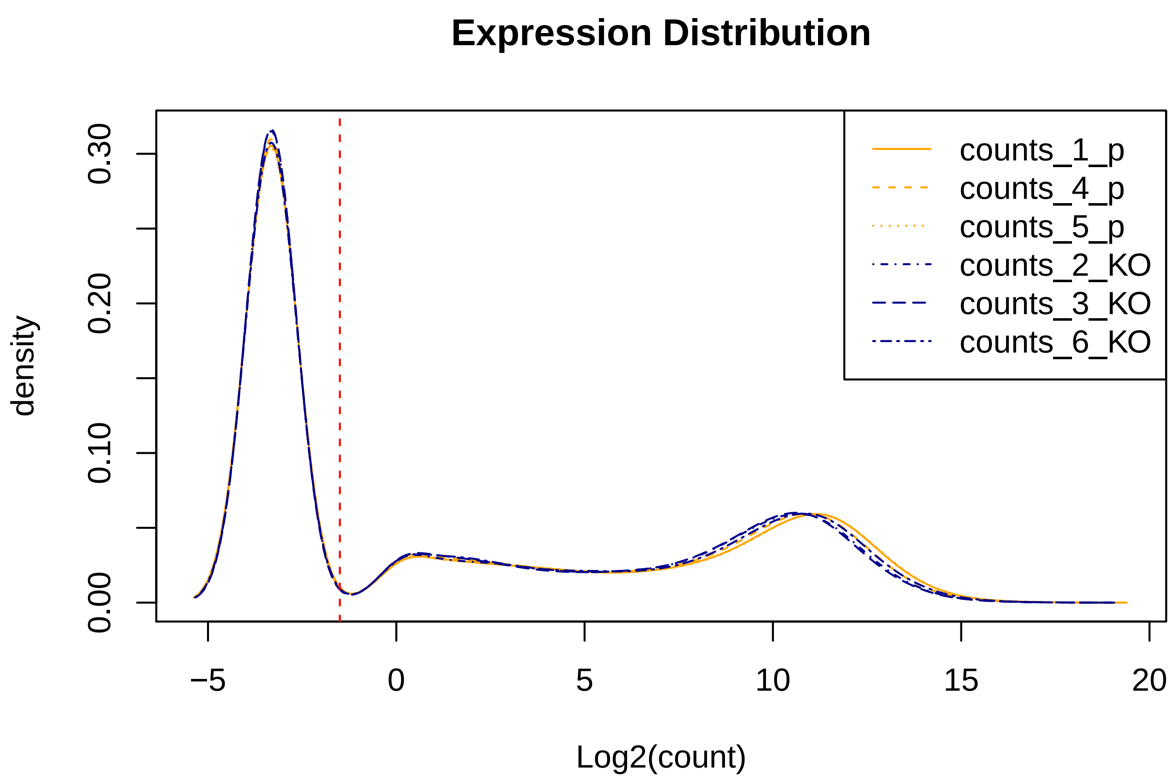

Figure 2: Expression distribution between all six sample data.

Normalisation

```
## Created a boxplot using MDS
barplot(colSums(myData) / 1e6, las = 2,
        col=rep(myColors, each=3),
        main = "Sequencing Depth",
        xlab = "Gene Samples",
        ylab = "Sequencing Depth (in millions)",
        names=c("Parent 1","Parent 2","Parent 3",
                "Knockout 1","Knockout 2", "Knockout 3"),
        cex.names=0.45)
```

```
# DESeq2 will construct a SummarizedExperiment object and combine this
# into a 'DESeqDataSet' object. The 'design' argument usually indicates the
# experimental design using the condition(s) names as a 'factor',
# for now we use just '~ 1'
(ddsMat <- DESeqDataSetFromMatrix(countData = myData,
                                  colData = data.frame(samples = names(myData)),
                                  design = ~ 1))
```

```
## class: DESeqDataSet
## dim: 36528 6
## metadata(1): version
## assays(1): counts
## rownames(36528): 11287 11298 ... 103611158 103611159
## rowData names(0):
## colnames(6): counts_1_p counts_4_p ... counts_3_KO counts_6_KO
```

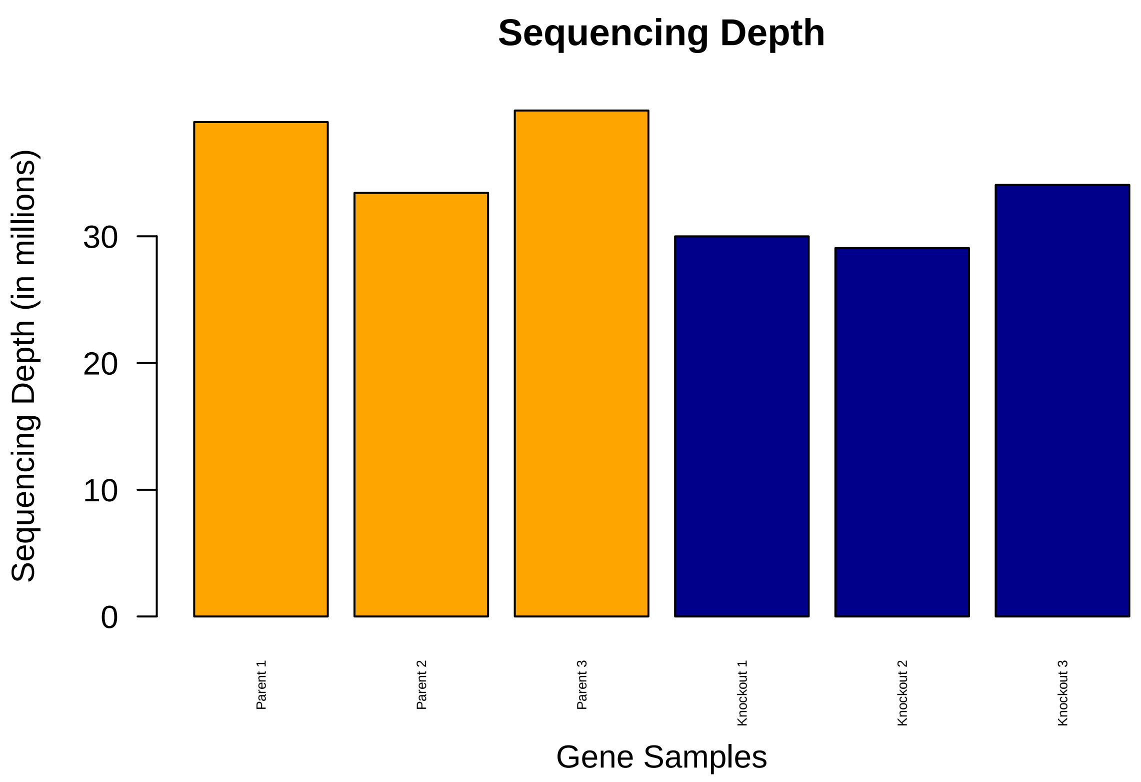

Figure 3: A boxplot showing the frequency of gene counts within the samples.

```
## colData names(1): samples
```

```r
# Perform normalization
rld.dds <- vst(ddsMat)

# 'Extract' normalized values
rld <- assay(rld.dds)
```

Distance Calculation

```r
# Calculate basic distance metric (using euclidean distance, see '?dist')
sampledists <- dist( t( rld ))
```

Heatmap

```r
# Convert the 'dist' object into a matrix for creating a heatmap.
sampleDistMatrix <- as.matrix(sampledists)

# The annotation is an extra layer that will be plotted above the heatmap.
# columns indicating the cell type
annotation <- data.frame(Type = factor(rep(1:2, each = 3),
                                        labels = c("Parental (control)",
                                                   "IRF2 KO (test)")))

# Set the rownames of the annotation dataframe to the sample names (required).
rownames(annotation) <- names(myData)
```

```
pheatmap(sampleDistMatrix, show_colnames = FALSE,
         annotation_col = annotation,
         clustering_distance_rows = sampledists,
         clustering_distance_cols = sampledists,
         main = "IRF2 Knockout Sample Distances")
```

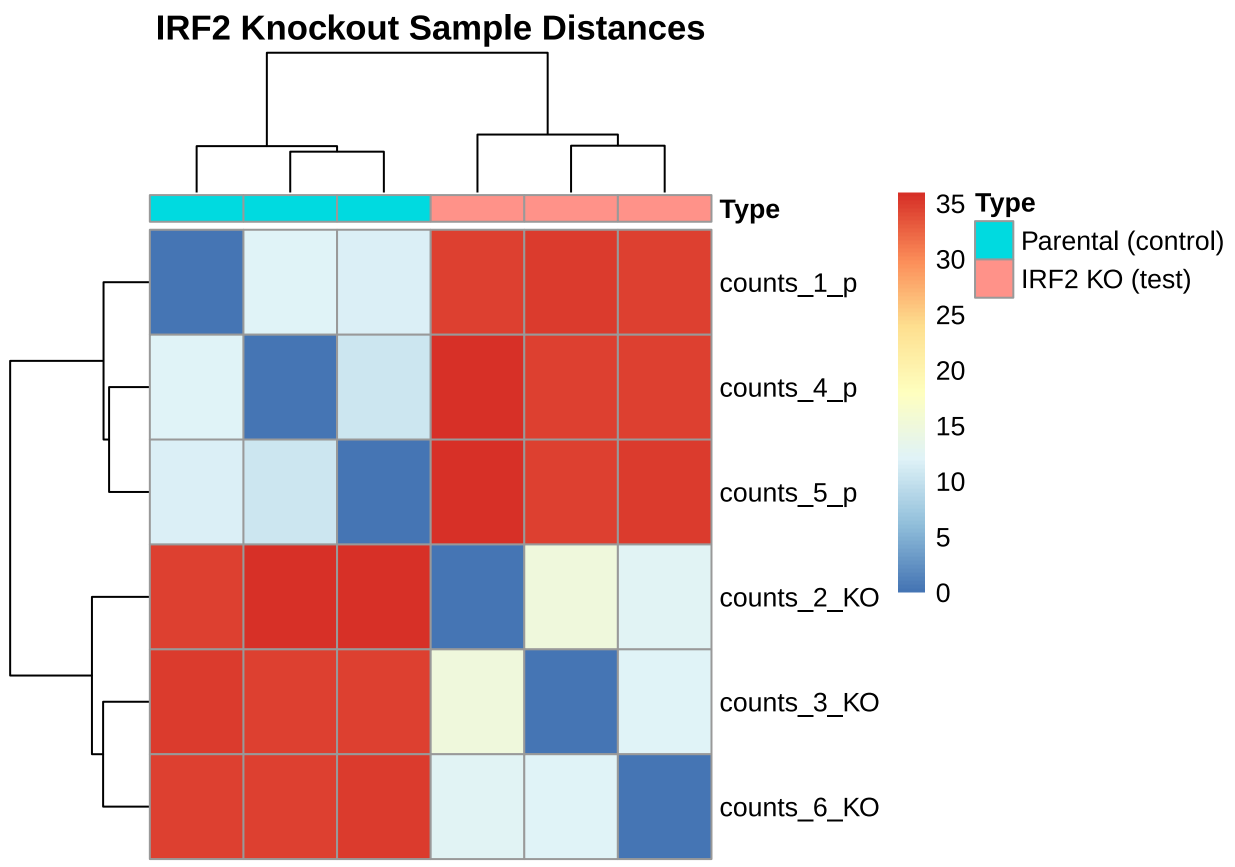

Figure 4: A heatmap showing the euclidean distance between the points of the control and test case.

Multi Dimensional Scaling (MDS)

```
# Use the raw (not r-log transformed!) counts.
dds <- assay(ddsMat)
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) )

# And set some better readable names for the columns.
names(mdsPoisData) <- c('x_coord', 'y_coord')

# Separate the annotation factor (as the variable name is used as label).
```

```r
groups <- factor(rep(1:2, each=3),
                 labels = c("Parental", "IRF2 KO"))
coldata <- names(myData)

# 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") +
  theme_bw()
```

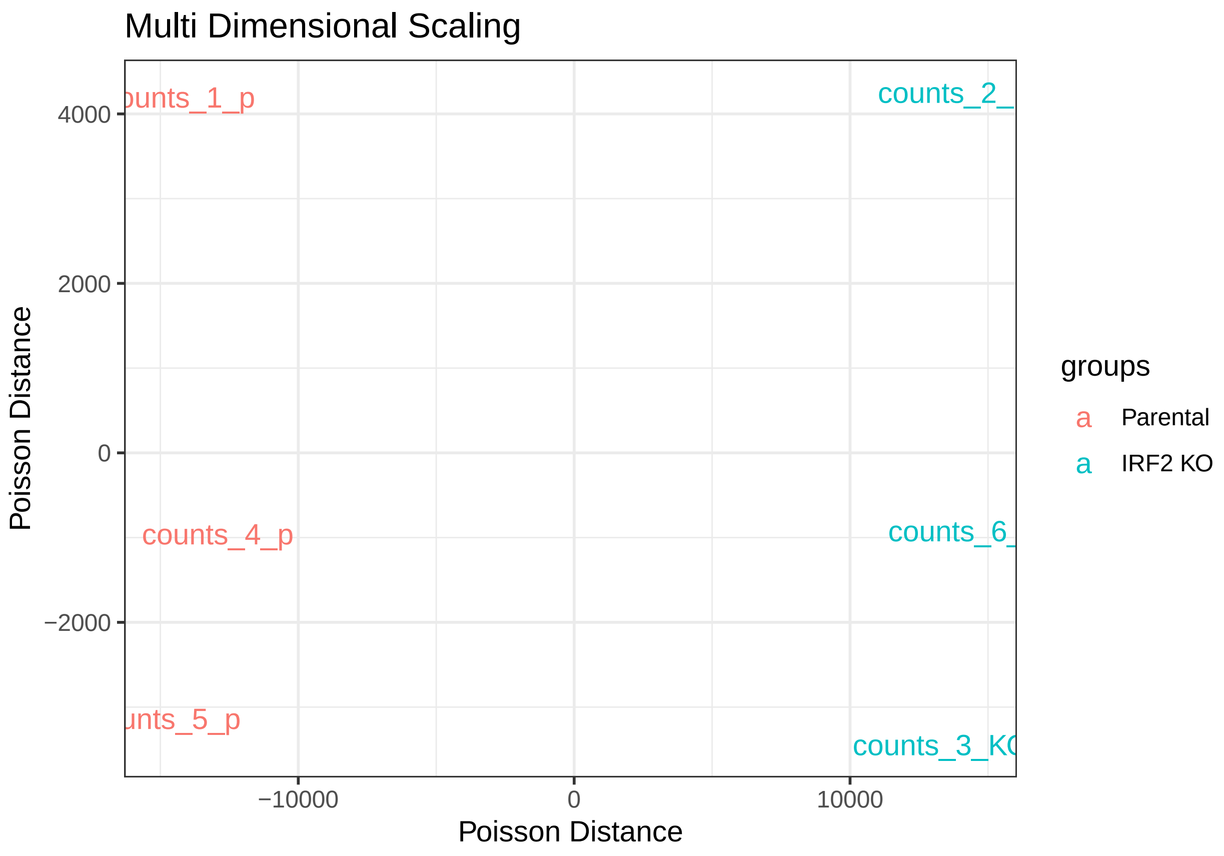

Figure 5: MDS plot revealing the poisson distance between the control and test case, showing a clear split between the two groups.

There is a clear poisson distance between the parental and the IRF2 KO gene samples. As this data set only contains 3 samples of each group, as well as the fact the data itself does not contain any missing values, it can be fully utilized for research.

Discovering Differentialy Expressed Genes

A gene is declared differentially expressed if a difference or change observed in read counts or expression levels/index between two experimental conditions is statistically significant. In the case of this experiment, Genes were considered to be differentially expressed if the log2 of the fold change was >1 or < minus 1 and the adjusted P value was < 0.05.

```r
# Perform a naive FPM normalization.
counts.fpm <- log2( (myData / (colSums(myData) / 1e6)) + 1 )

# if any value of a row have values of below five, remove them.
row_sub <- apply(counts.fpm, 1, function(row) all(row > 5 ))
row_removed <- apply(counts.fpm, 1, function(row) any(row <= 5))

# Show the identifiers of the genes that were removed due to containing 0 values.
removed_genes <- counts.fpm[row_removed,]

# Show the remaining genes and their count values.
genes <- counts.fpm[row_sub,]

totalAmount <- dplyr::count(myData)[[1]]
keepAmount <- dplyr::count(genes)[[1]]
removeAmount <- dplyr::count(removed_genes)[[1]]

keepPercent <- round((keepAmount*100)/totalAmount)
removedPercent <- round((removeAmount*100)/totalAmount)

cat("The total amount of genes in the data is",totalAmount,
    ".\n Of those genes,",keepAmount,
    " were above a count of zero,\n with",removeAmount,
    " equal to it or below. In conclusion,\n",keepPercent,
    "% of the data will be kept, while",removedPercent,"% will be removed.")
```

```
## The total amount of genes in the data is 36528 .
##  Of those genes, 5777  were above a count of zero,
##  with 30751  equal to it or below. In conclusion,
##  16 % of the data will be kept, while 84 % will be removed.
```

The Fold Change Value (FC)

Fold change (FC) is a measure describing the degree of quantity change between final and original value.

```r
# Create a function that performs a t.test per row in a data frame.
myTTest <- function(y){
  return(t.test(y[1:3], y[4:6])$p.value)
}

# assign the row and column names.
gene.names <- rownames(genes)
samples <- colnames(genes)

# Apply the function to the logged data set.
pVec <- apply(genes, 1, myTTest)
alpha <- 0.05

#Show how many DEGs have been found.
cat("Number of DEGs found through EdgeR:\n")
```

```
## Number of DEGs found through EdgeR:
```

```
( n.DEG <- sum(pVec < alpha) )
```

```
## [1] 2735
```

```
# Create two columns, one with means of parental and one with Knockout.
genes$avg_p <- rowMeans(genes[1:3])
genes$avg_KO <- rowMeans(genes[4:6])

# Calculate the FC values.
genes$foldChange <- (genes$avg_KO - genes$avg_p)

# Display the values as a histogram.
hist(as.matrix(genes$foldChange), breaks=80,
     main="Fold Change Value of the Gene Counts", xlab="FC value",
     col= rgb(0.5,0,0.8,1/2))

abline(v = c(-1,1), col="red")
```

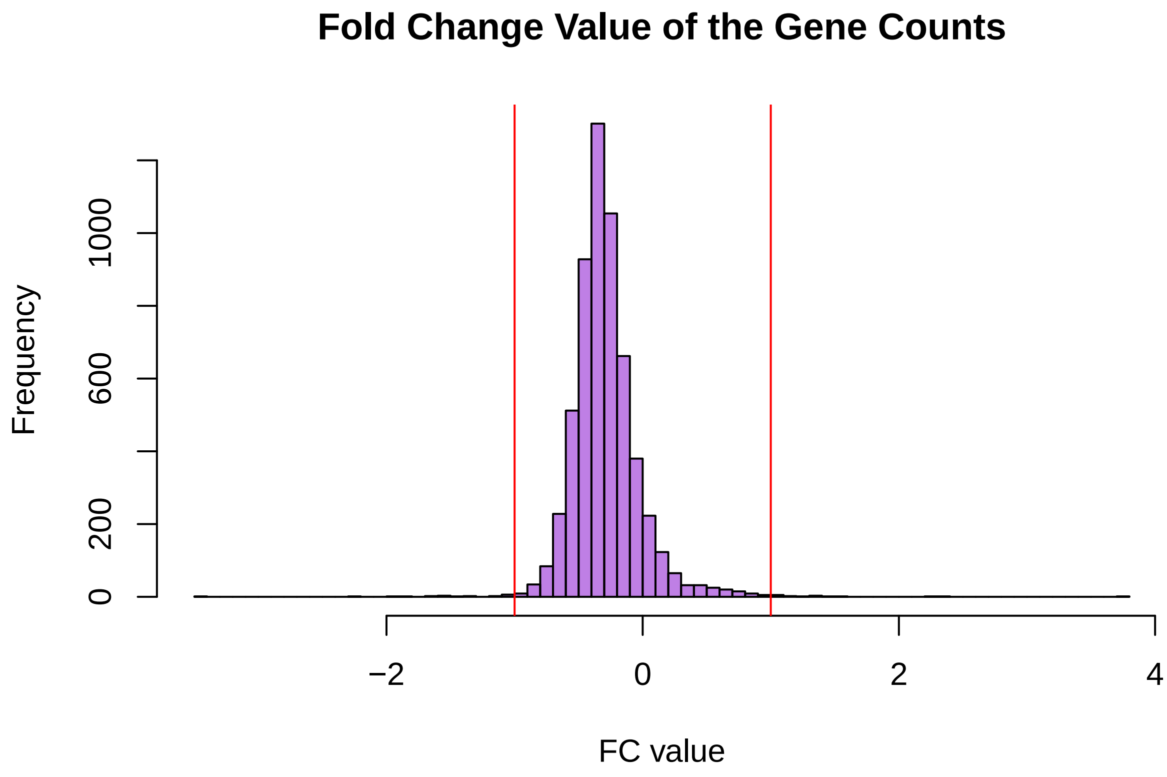

Figure 6: barplot showing the frequency of FC values within the normalised data, with those beyond the red lines to be considered DEGs

Using Bioconductor Packages

Differential expression analysis was performed in the original research paper using the voom/limma R package. Genes were considered to be differentially expressed if the log2 of the fold change was >1 or < 1 and the adjusted P value was <0.05. In addition to this package, the EdgeR package is used afterwards for result comparison.

Limma package

```
# Read in the file containing the count data.
counts <- read.csv("Data/Samples/Reformat/myData.csv", row.names = 1)
head(counts)
```

```
##       counts_1_p counts_4_p counts_5_p counts_2_KO counts_3_KO counts_6_KO
## 11287          0          0          0           3           0           1
## 11298          4          1          0           0           2           1
## 11302       1482       1404       1715        1329        1383        1527
## 11303         34         29         23          16          17          23
## 11304          0          0          0           0           0           0
## 11305       6951       6712       7791        5334        5036        5986
```

```
# Create DEGList object.
d0 <- DGEList(counts, group = c(1,1,1,2,2,2))

#Calculate normalization factors.
d0 <- calcNormFactors(d0)
```

```
# Filter low-expressed genes
cutoff <- 5
drop <- which(apply(cpm(d0), 1, max) < cutoff)
d <- d0[-drop,]

# number of genes left
dim(d)
```

```
## [1] 10357     6
```

```
# Multidimensional scaling (MDS) plot.
group <- factor(c(1,1,1,2,2,2), labels = c("Control", "Case"))
plotMDS(d, col = as.numeric(group))
```

```
# Voom transformation and calculation of variance weights.
mm <- model.matrix(~0 + group)

y <- voom(d, mm, plot = T)
```

This model was created through the following steps:

1. Counts are transformed to log2 counts per million reads (CPM), where "per million reads" is defined based on the normalization factors we calculated earlier
2. A linear model is fitted to the log2 CPM for each gene, and the residuals are calculated
3. A smoothed curve is fitted to the sqrt(residual standard deviation) by average expression (see red line in plot above)
4. The smoothed curve is used to obtain weights for each gene and sample that are passed into limma along with the log2 CPMs.

```
# Fitting linear models in limma

fit <- lmFit(y, mm)
head(coef(fit))
```

```
##       groupControl groupCase
## 11302     5.345580  5.518169
## 11305     7.568806  7.465676
```

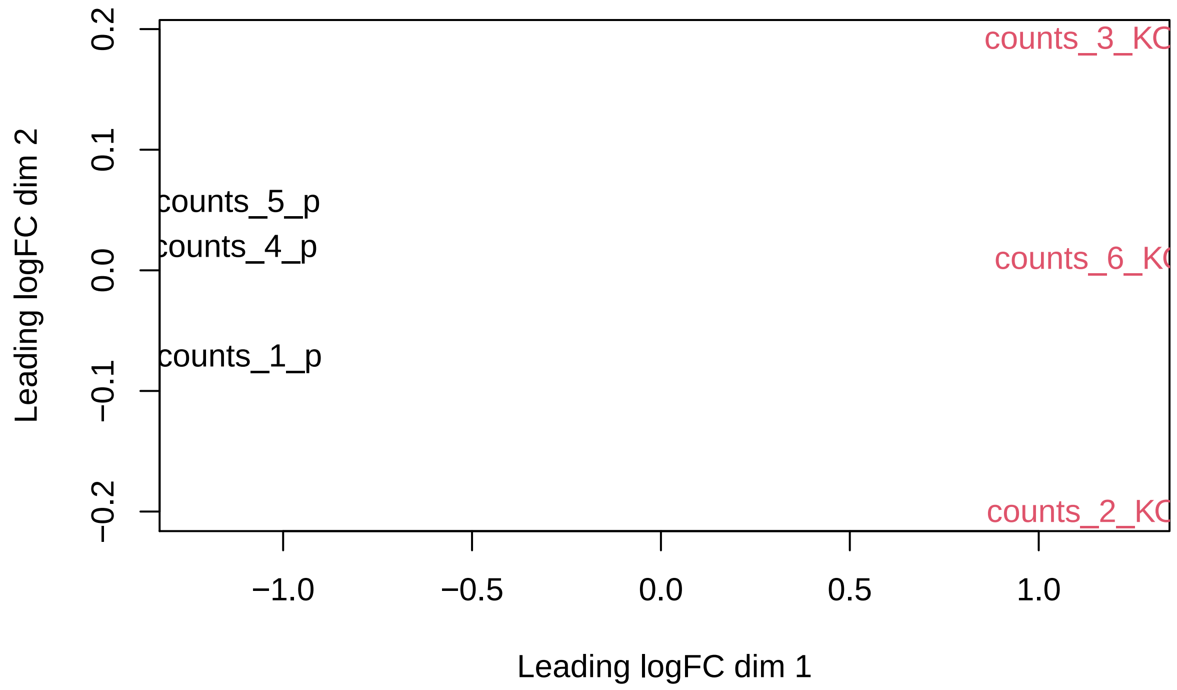

Figure 7: The parental control and knockout test samples are shown to have witnessable clustering and observably different logFC values.

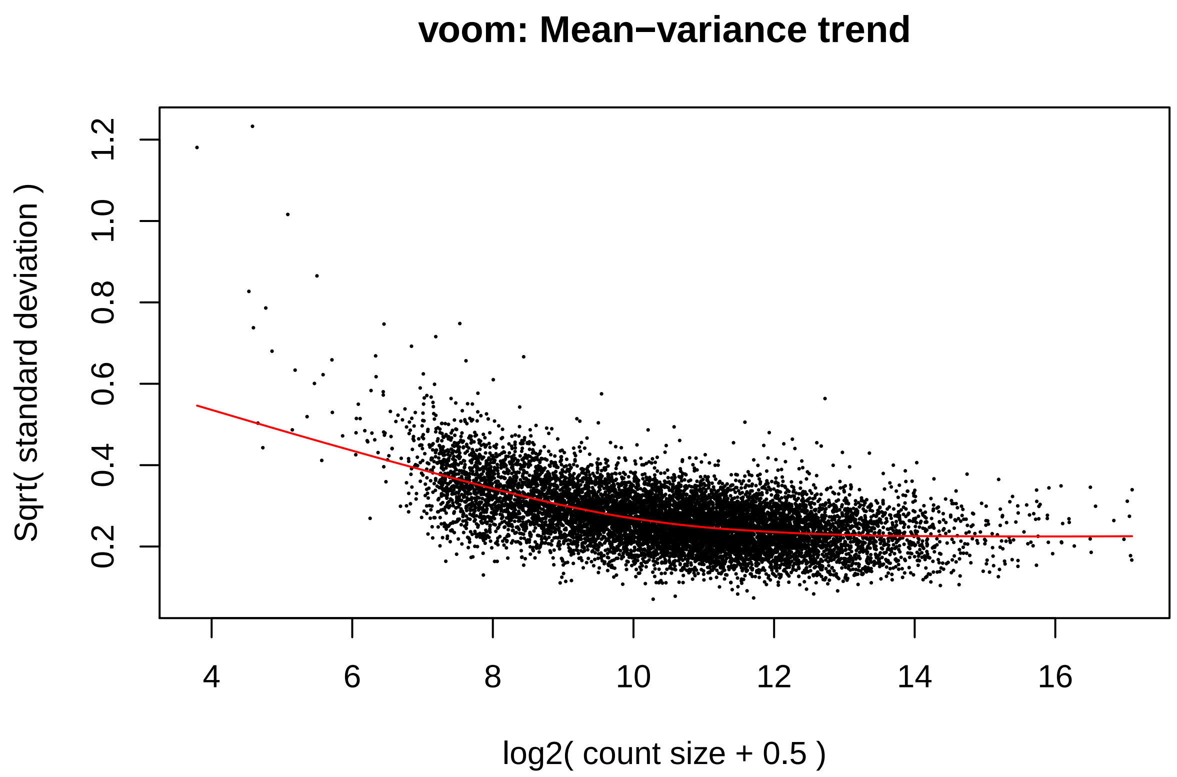

Figure 8: An MA plot revealing the differences between measurements taken in the test vs control case, by transforming the data onto M (log ratio) and A (mean average) scales, then plotting these values.

```
## 11306     6.726433  6.732848
## 11307     5.319189  6.185097
## 11308     6.702686  6.814790
## 11350     6.771748  6.781481
```

```
# Comparison between control (parental) and case (KO)
contr <- makeContrasts(groupCase - groupControl, levels = colnames(coef(fit)))
contr
```

```
##               Contrasts
## Levels         groupCase - groupControl
##   groupControl                       -1
##   groupCase                           1
```

```
# Estimate contrast for each gene
tmp <- contrasts.fit(fit, contr)

# Empirical Bayes smoothing of standard errors
tmp <- eBayes(tmp)

# What genes are most differentially expressed?
top.table <- topTable(tmp, sort.by = "P", n = Inf)
head(top.table, 20)
```

```
##             logFC    AveExpr         t      P.Value    adj.P.Val        B
## 229900  4.534602  6.7927150  76.58869 2.924981e-22 1.665398e-18 41.24494
## 19039   4.112900  7.3489751  76.14268 3.215986e-22 1.665398e-18 41.24415
## 20249  -3.184822  7.7135569 -73.72097 5.435623e-22 1.876558e-18 40.77069
## 16956   2.705099  7.5232999  63.07838 6.824919e-21 1.767142e-17 38.29375
## 55932   6.689844  3.5365449  55.37042 5.645200e-20 1.169347e-16 34.59495
## 16145   2.879905  5.4213448  51.77341 1.675406e-19 2.892030e-16 35.07902
## 246727  9.082081  2.2073891  51.11224 2.062914e-19 3.052228e-16 32.12644
## 18109  -4.234664  4.2583134 -50.04836 2.899576e-19 3.753864e-16 34.18354
## 22169   4.589853  4.9623625  49.21820 3.800827e-19 4.373908e-16 34.14997
## 74153   2.594233  6.8803883  48.75046 4.435793e-19 4.594151e-16 34.10292
## 229898  7.722112  3.2942887  47.36853 7.062762e-19 6.649911e-16 32.11212
## 102294  2.628729  5.5334475  47.10054 7.741414e-19 6.681486e-16 33.56805
## 21355  -4.178850  4.1104715 -46.07122 1.106549e-18 8.815794e-16 32.92791
## 20411   2.192752  6.0264632  45.72847 1.248510e-18 9.236298e-16 33.07514
## 13058   2.907935  5.5023790  45.45908 1.373620e-18 9.484392e-16 32.99501
## 93675  -3.461161  5.3280281 -44.62634 1.851916e-18 1.198769e-15 32.68660
## 67775  10.773940 -0.3175057  43.34787 2.961711e-18 1.767268e-15 30.29234
## 229474  3.049953  4.4196772  43.25034 3.071432e-18 1.767268e-15 32.15631
## 58185   8.447734  2.9183258  42.33695 4.335331e-18 2.343871e-15 30.18556
## 54396   2.367912  5.7466106  42.22413 4.526158e-18 2.343871e-15 31.78160
```

logFC: log2 fold change of Control/Case

AveExpr: Average expression across all samples, in log2 CPM

t: logFC divided by its standard error

P.Value: Raw p-value (based on t) from test that logFC differs from 0

adj.P.Val: Benjamini-Hochberg false discovery rate adjusted p-value

B: log-odds that gene is DE (arguably less useful than the other columns)

```r
cat("Number of DEGs found by using the limma package:\n")
```

```
## Number of DEGs found by using the limma package:
```

```r
cat(length(which(top.table$logFC < -1 | top.table$logFC > 1)))
```

```
## 438
```

```r
# Display the values as a histogram.
hist(as.matrix(top.table$logFC), breaks=80,
     main="Fold Change Value of the Gene Counts",
     xlab="FC value", col= rgb(0.5,0,0.8,1/2))
abline(v = c(-1,1), col="red")
```

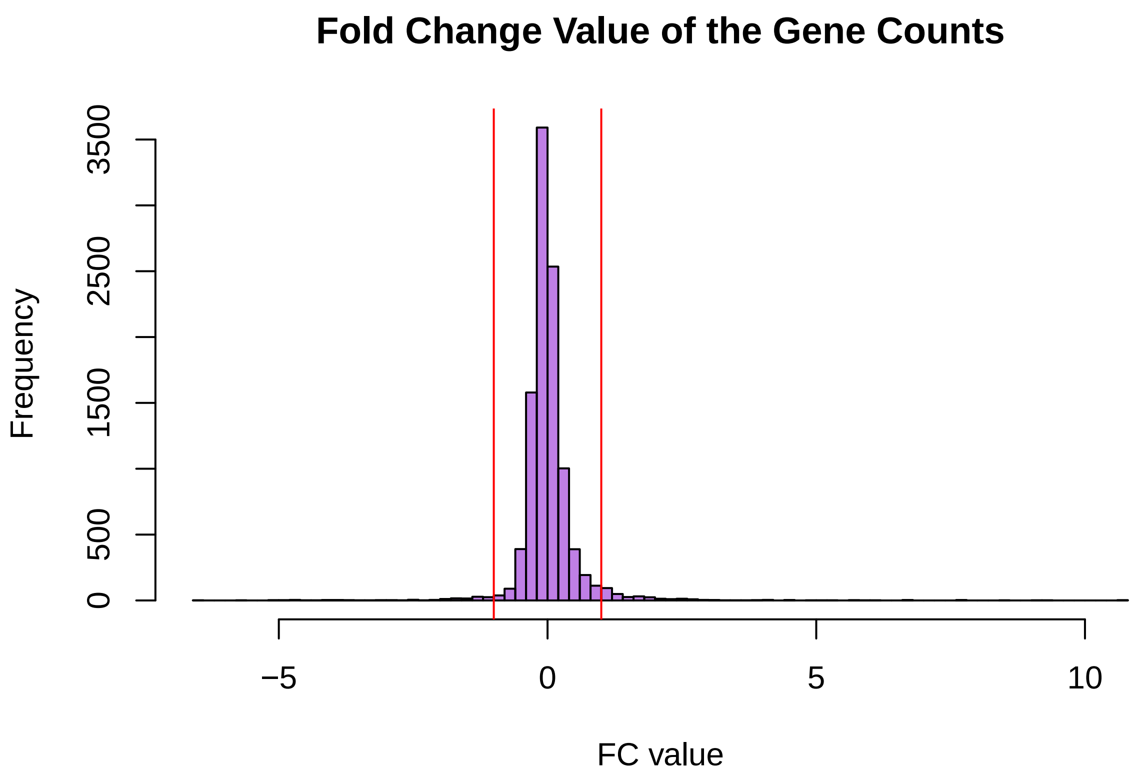

Figure 9: barplot showing the frequency of FC values within the normalised data of limma, with those beyond the red lines to be considered DEGs.

Data Analysis and Visualisation

Volcano plot

```r
## Simple function for plotting a Volcano plot, returns a ggplot object
deseq.volcano <- function(res, datasetName) {
  return(EnhancedVolcano(res, x = 'logFC', y = 'adj.P.Val',
                         lab=rownames(res),
                         title = paste(datasetName, "Parental vs Knockout"),
                         subtitle = bquote(italic('adj.P.Val <= 0.05 and absolute FC >= 1')),
                         # Change text and icon sizes
                         labSize = 3, pointSize = 1.5, axisLabSize=10,
                         titleLabSize=12, subtitleLabSize=8, captionLabSize=10,
                         # Disable legend
                         legendPosition = "none",
                         # Set cutoffs
                         pCutoff = 0.05, FCcutoff = 1))
}

deseq.volcano(res = top.table, datasetName = "top")
```

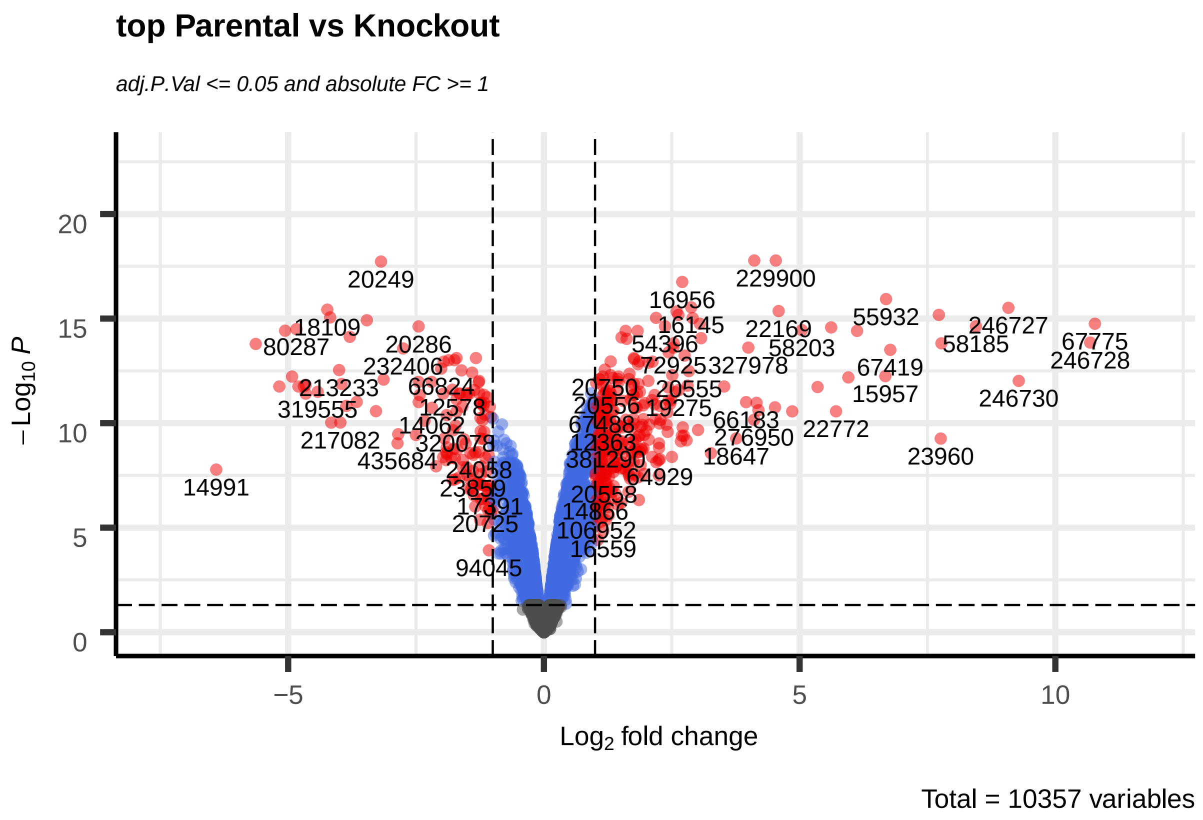

Figure 10: A volcano plot displaying the gene id's of the normalised count data with an absolute FCvalue of >= 2 and an adjusted P value <= 0.05

This plot shows the amount of genes that are considered DEG's by the results of the limma/voom package. Those in red are DEG's, while those in blue are not.

Venn Diagram

```
# Create a DGEList object
y <- DGEList(counts, group = c(1,1,1,2,2,2))

# calculate the normalised factors.
y <- calcNormFactors(y)

# estimate the dispersion.
y <- estimateDisp(y)
```

```
## Design matrix not provided. Switch to the classic mode.
```

```r
# estimate the common dispersion.
y <- estimateCommonDisp(y)

# estimate the tagwise dispersion.
y <- estimateTagwiseDisp(y)

# excecute the exactTest upon the DGEList object.
et <- exactTest(y)

# retrieve the results from the exactTest.
tt <- topTags(et, n = Inf)

# retrieve the gene id's of the results from the exactTest which align with
# the aforementioned DEG consideration.
edgeR.gene.ids <- row.names(tt$table)[which(abs(tt$table$logFC)
                                            > 1 & tt$table$FDR < 0.05)]
```

EdgeR.gene.ids contain the id's of all genes in the data set that follow the FC value >1 or < -1, as well as the needed p value < 0.05.

```r
# retrieve the gene id's of the results from the limma package toptable
# which align with the aforementioned DEG consideration.
voom.gene.ids <- row.names(top.table)[which(abs(top.table$logFC)
                                            > 1 & top.table$adj.P.Val < 0.05)]



# Create a venn Diagram displaying the results from the limma package and
# the EdgeR package, and compare the resulting DEGs.
venn.plot <- draw.pairwise.venn(length(voom.gene.ids),
                                length(edgeR.gene.ids),
                                # Calculate the intersection of the two sets
                                length(intersect(voom.gene.ids, edgeR.gene.ids)),
                                category = c("Voom", "EdgeR"),
                                scaled = T,
                                fill = c("light blue", "pink"),
                                alpha = rep(0.5, 2),
                                cat.pos = c(0, 0))

# Plot the plot.
grid.draw(venn.plot)
```

The diagram shows that borderline every DEG found by Limma is located in the same data found by EdgeR. While limma is usually not used for RNA sequences but rather microarrays, it nevertheless shows a more strict method to locating the DEGs. in conclusion, those located within both the voom and edgeR diagrams are the Differentially Expressed Genes between the control case and the IRF2 silenced test case.

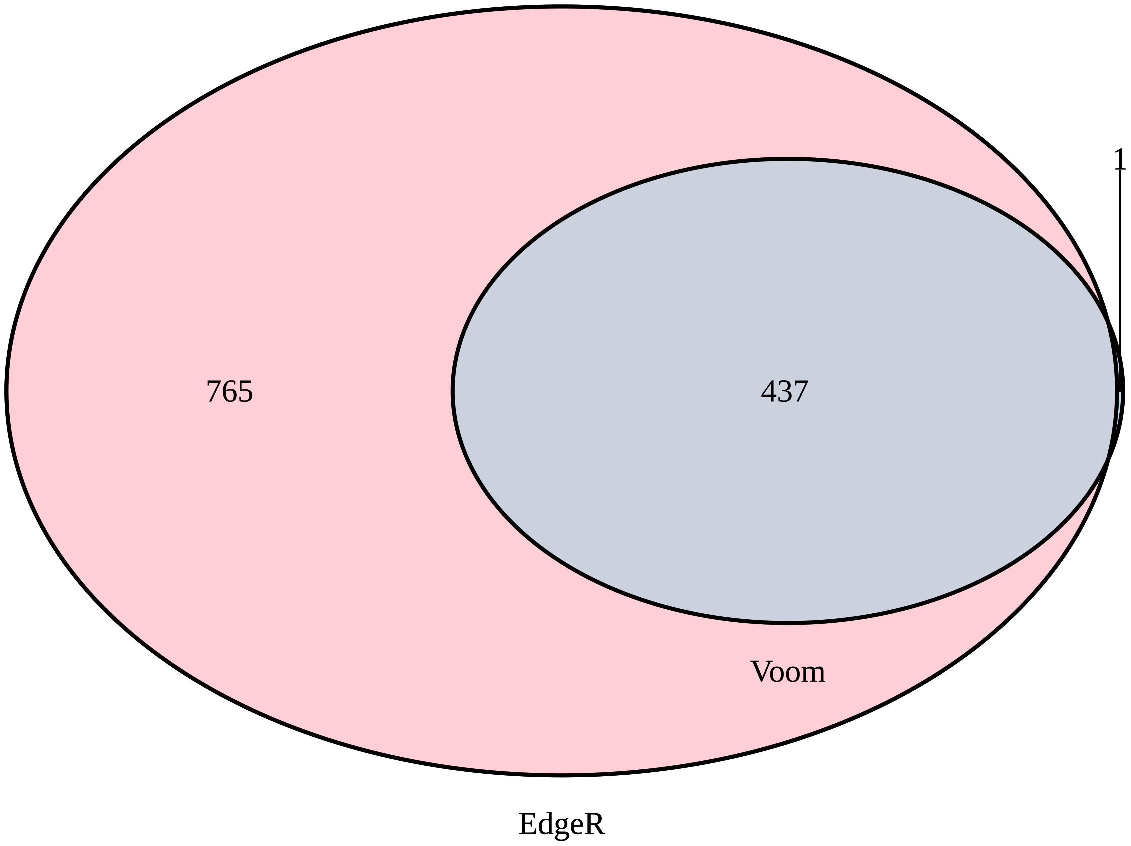

Figure 11: displaying the DEG's found by EdgeR and the limma packages respectively.