SP1 Analysis

Orfeas Gkourlias

3/10/2022

This markdown document aims to utilize R to discover, analyse and sketch DEG data. All the data used can be found in the corresponding git repo. Before getting started with the data, all libraries will need to be loaded in first. Most of these are Bioconductor packages.

```
knitr::opts_chunk$set(echo = TRUE)
library(affy)
library(scales)
library(DESeq2)
library(pheatmap)
library(ggplot2)
library(PoiClaClu)
library(edgeR)
library(EnhancedVolcano)
library(pathview)
library(pander)
library(SPIA)
```

3. Exploratory Data Analysis

The data will be need to be extracted from the count file first. Since it's in .tsv format, the seperator is going to be tab based. The inclusion of headers adds an X to the sequence ID's, because R is unable to make headers out of just integers. To counteract this, the colnames will be manually added.

3.1 Loading the data

```
file <- c("..\\data\\GSE152262_RNAseq_Raw_Counts.tsv")
# Raw_Data will be the primary dataframe that gets worked on.
raw_data <- read.table(file, sep = '\t', header = TRUE, row.names = 1)

# The first two and single last columns are the case samples.
# Control samples are indicated with con.
colnames(raw_data) <- c("case24275", "case24277", "con4279", "con4280", "con4280a", "case24281")

# Rearranging the columns so that the first three are the case samples.
raw_data <- raw_data[, c(1,2,6,3,4,5)]

# Showing the first five rows as an example.
raw_data[1:5,]
```

```
##                 case24275 case24277 case24281 con4279 con4280 con4280a
## ENSG00000000003        23        30         8      11      43       31
## ENSG00000000005         0         0         0       0       0        2
## ENSG00000000419       778       910       1051      838      911      1113
## ENSG00000000457       378       438        389      441      772       738
## ENSG00000000460        44        51         28       58       61        65
```

```r
# Showing the dimension and structure of the raw_data data frame.
dim(raw_data)
```

```
## [1] 58307     6
```

```r
str(raw_data)
```

```
## 'data.frame':    58307 obs. of  6 variables:
##  $ case24275: int  23 0 778 378 44 14575 30 54 213 546 ...
##  $ case24277: int  30 0 910 438 51 21109 23 89 206 589 ...
##  $ case24281: int  8 0 1051 389 28 27759 68 50 180 561 ...
##  $ con4279  : int  11 0 838 441 58 7164 94 105 333 452 ...
##  $ con4280  : int  43 0 911 772 61 11710 151 77 419 407 ...
##  $ con4280a : int  31 2 1113 738 65 11846 148 69 384 373 ...
```

The data is now loaded in as a data frame. Every row shows the raw counts of a specific gene being expressed. 4275, 4277 and 4281 are the variant types. The datatypes are correct in this case_log2. There should only be integers included, except for the gene names.

Now that the data has been properly loaded, objects can be made to differentiate the control_log2 and case_log2 counts. Before separating the groups, it'll be useful to apply a log2 function to our data. This makes it so that the data is more informative and tidier, because of outliers and the big range being worked with.

```r
# Transforming the read data of every columns to the log2 value
# 1 is added to every column to make sure there are no log2(0) values.
raw_data_log2 <- log2(raw_data + 1)

# Dividing the case and controls columns into separate data frames for later use.
case <- raw_data[,c(1:3)]
control <- raw_data[,c(3:6)]

# Applying the same division, but with the log values for plotting purposes.
case_log2 <- raw_data_log2[,c(1:3)]
control_log2 <- raw_data_log2[,c(4:6)]

# Displaying the first rows of divided data frames.
case_log2[1,]
```

```
##                 case24275 case24277 case24281
## ENSG00000000003  4.584963  4.954196  3.169925
```

```r
control_log2[1,]
```

```
##                  con4279  con4280 con4280a
## ENSG00000000003 3.584963 5.459432        5
```

The control_log2 and case_log2 data is now stored in different variables, as shown above.

3.3 Visualizing using boxplot and density plot

More insight on the data can be gained by plotting and summarizing it. Every column will first be summarized. Following that, the mean values will be compared in a box plot.

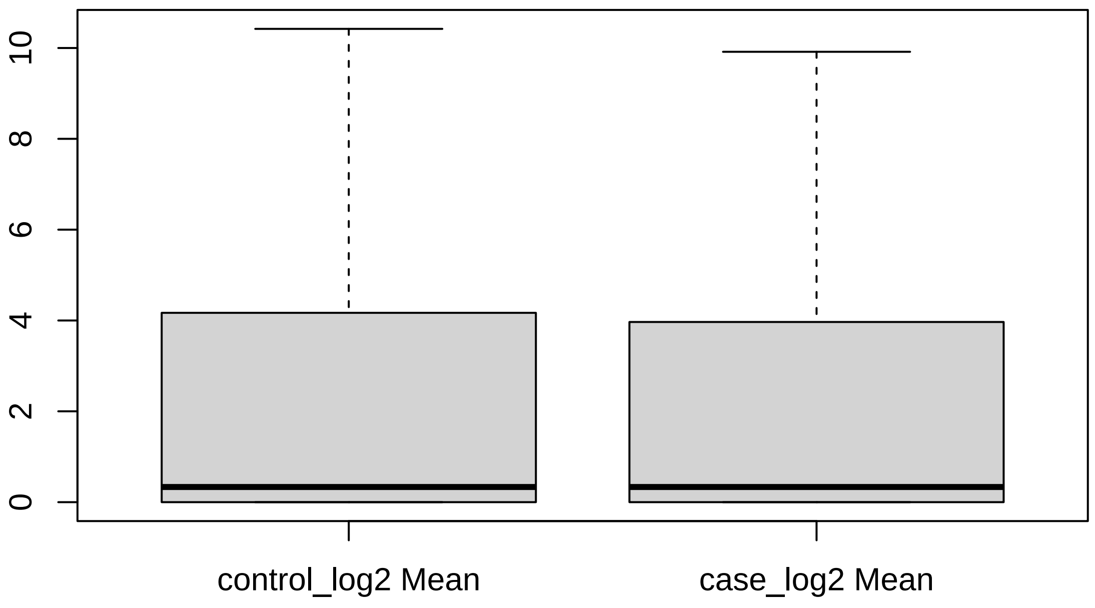
```r
# Applying a summary on all the log2 data.
summary(raw_data_log2)
```

```
##    case24275        case24277        case24281         con4279
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 0.000   Median : 0.000   Median : 0.000   Median : 0.000
##  Mean   : 2.424   Mean   : 2.552   Mean   : 2.405   Mean   : 2.524
##  3rd Qu.: 3.907   3rd Qu.: 4.248   3rd Qu.: 3.907   3rd Qu.: 4.170
##  Max.   :23.704   Max.   :23.582   Max.   :23.642   Max.   :23.549
##     con4280          con4280a
##  Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 0.000   Median : 0.000
##  Mean   : 2.579   Mean   : 2.571
##  3rd Qu.: 4.248   3rd Qu.: 4.170
##  Max.   :23.675   Max.   :24.155
```

```r
# Getting the mean values of both controlled and case sample expression values
# For every gene. This might be useful later.
case_log2$mean = apply(X = case_log2[1:3], MARGIN = 1, FUN = mean)
control_log2$mean = apply(X = control_log2[1:3], MARGIN = 1, FUN = mean)

# Doing the same to the raw data frames.
case$mean = apply(X = case[1:3], MARGIN = 1, FUN = mean)
control$mean = apply(X = control[1:3], MARGIN = 1, FUN = mean)

# Plotting the log2 data.
boxplot(control_log2$mean, case_log2$m, outline = FALSE,
        names = c("control_log2 Mean", "case_log2 Mean"))
```



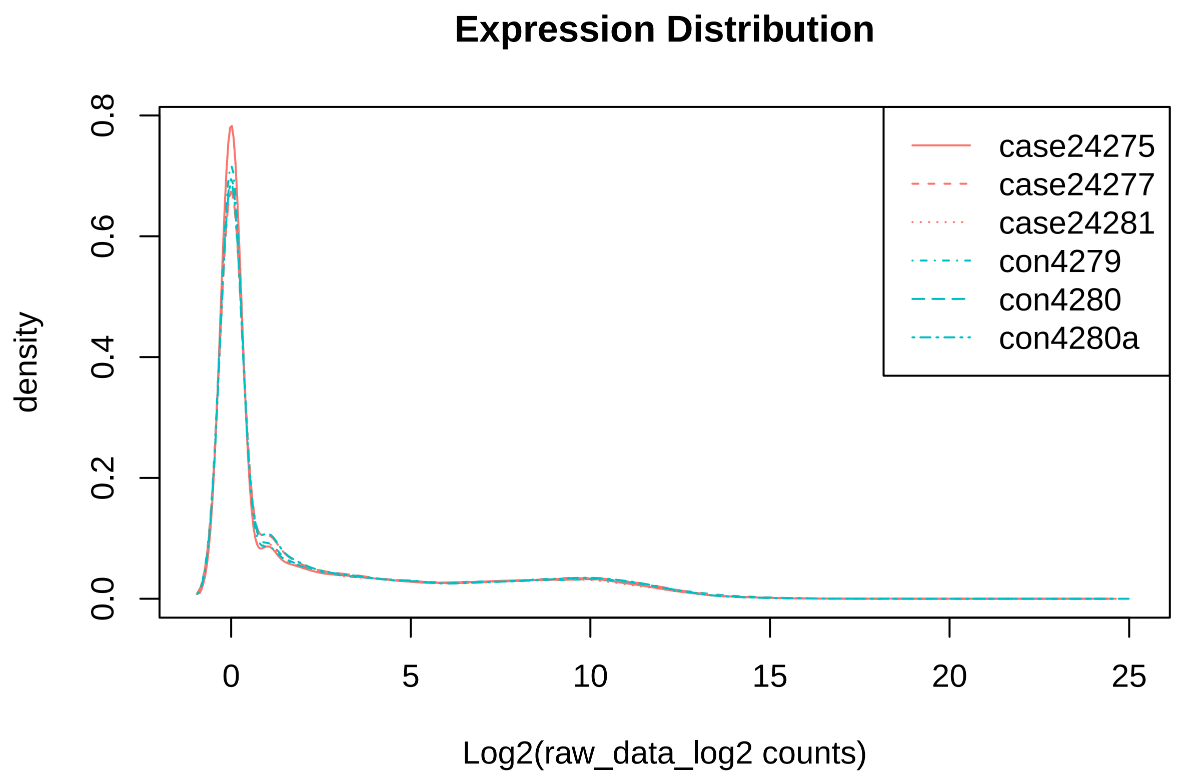
These boxplots are not yet very informative. The only thing that can be seen from them is that the case_log2s have a slightly lower expression level on average

Maybe a density plot allows for a more informative figure.

```r
# Creating the recurring colors that will be used for the columns
myColors <- hue_pal()(2)

# Density plotting the log2 data, using the colors created above.
plotDensity(raw_data_log2, col=rep(myColors, each=1),
            lty=c(1:ncol(raw_data_log2)),
            main = "Expression Distribution",
            xlab = "Log2(raw_data_log2 counts)")

# Adding a legend for clarity.
legend('topright', names(raw_data_log2), lty=c(1:ncol(raw_data_log2)),
       col=rep(myColors, each=3))
```



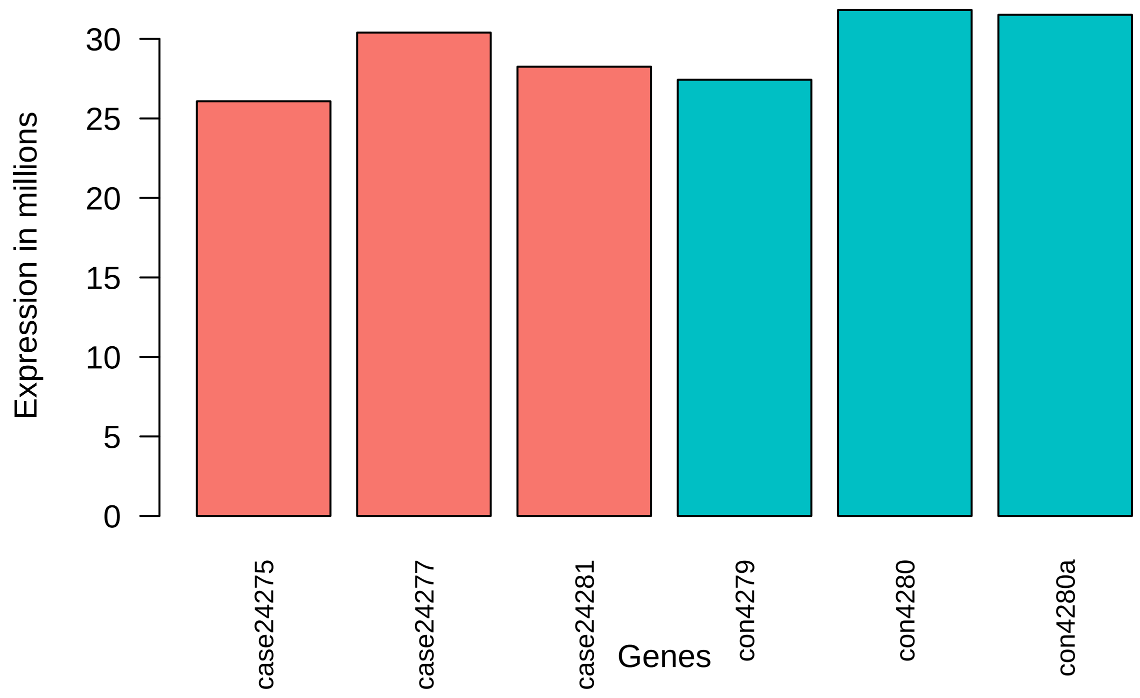
As can be seen in the plot, the highest amount of expressions, besides 0, seem to be around 10.

3.4 Visualizing using heatmap and MDS

Before continuing with this step, the data will have to be normalized. There are 5 which rows are not actual genes. They will be removed. After that, a barplot will be generated to show whether theres a difference in expression in millions, using col sums.

```
# There are 5 rows which do not count for actual genes.
# These rows are currently not relevant, but shouldn't be present from now on.
remove_rows <- c("__not_aligned", "__no_feature", "__no_feature",
                 "__alignment_not_unique", "__too_low_aQual", "__ambiguous")
raw_data <- raw_data[!(row.names(raw_data) %in% remove_rows),]

# Bar plotting the new data. Division by 1e6 shows the values in millions.
barplot(colSums(raw_data) / 1e6, las = 2, cex.names = 0.8,
        col = c(rep(myColors[1],3), rep(myColors[2],3)), xlab = "Genes",
        ylab = "Expression in millions")
```



Judging by that figure, the control group seems to have a higher average expression when summarised on all genes.

Now the DESeq2 library will be used to normalize the data. the VST function within this package is the next function. The data will first have to make a Summarized Experiment object, which is done first.

```
# Creating the dds Matrix, so that it ca nbe used in the vst function
(ddsMat <- DESeqDataSetFromMatrix(countData = raw_data,
                                  colData = data.frame(samples=names(raw_data)),
                                  design = ~ 1))
```

```
## class: DESeqDataSet
## dim: 58302 6
## metadata(1): version
## assays(1): counts
## rownames(58302): ENSG00000000003 ENSG00000000005 ... ENSG00000284747
##   ENSG00000284748
## rowData names(0):
## colnames(6): case24275 case24277 ... con4280 con4280a
## colData names(1): samples
```

```
# Applying vst and saving it into the rld.dds object.
rld.dds <- vst(ddsMat)

# Applying assay on that object then saving it into rld.
rld <- assay(rld.dds)
```

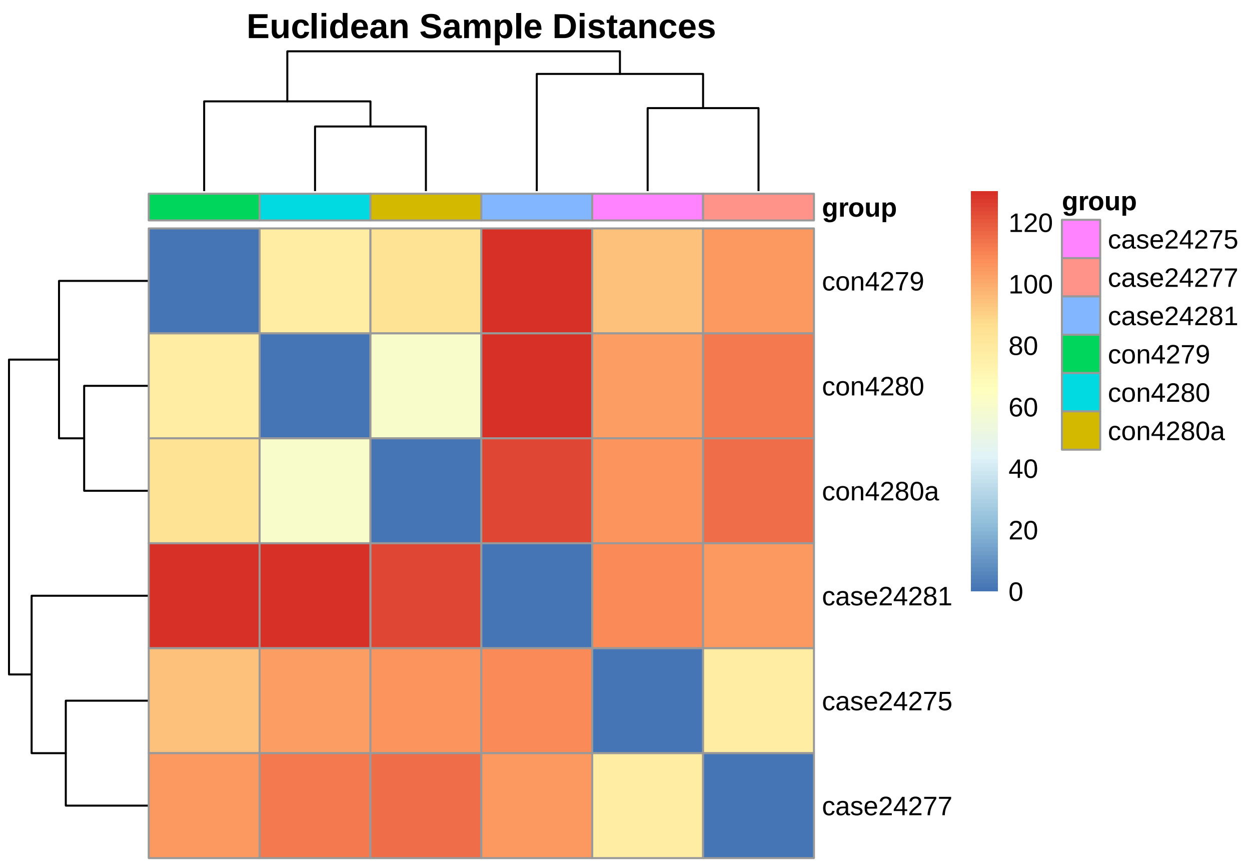
Distance calculation may now be performed on the normalized data. The matrix will first have to be transposed. After distance calculations have been performed, a heat map may be constructed.

```
# To create the heatmap, distances first get calculated & stored in a matrix
sampledists <- dist( t( rld ))
sampleDistMatrix <- as.matrix(sampledists)

# Annotation dataframe gets created for heatmap.
annotation <- data.frame(group = factor(c(1:6),
                                        labels = c(names(raw_data))))

# Rownames for the annotation get taken from raw_data.
rownames(annotation) <- names(raw_data)

# Heat map function gets called on the matrix and annotation objects.
pheatmap(sampleDistMatrix, show_colnames = FALSE,
         annotation_col = annotation,
         clustering_distance_rows = sampledists,
         clustering_distance_cols = sampledists,
         main = "Euclidean Sample Distances")
```



The resulting heatmap shows where the large differences in expression are located.

The distances can also be shown using a 2d-plot, by performing multi dimensional scaling.

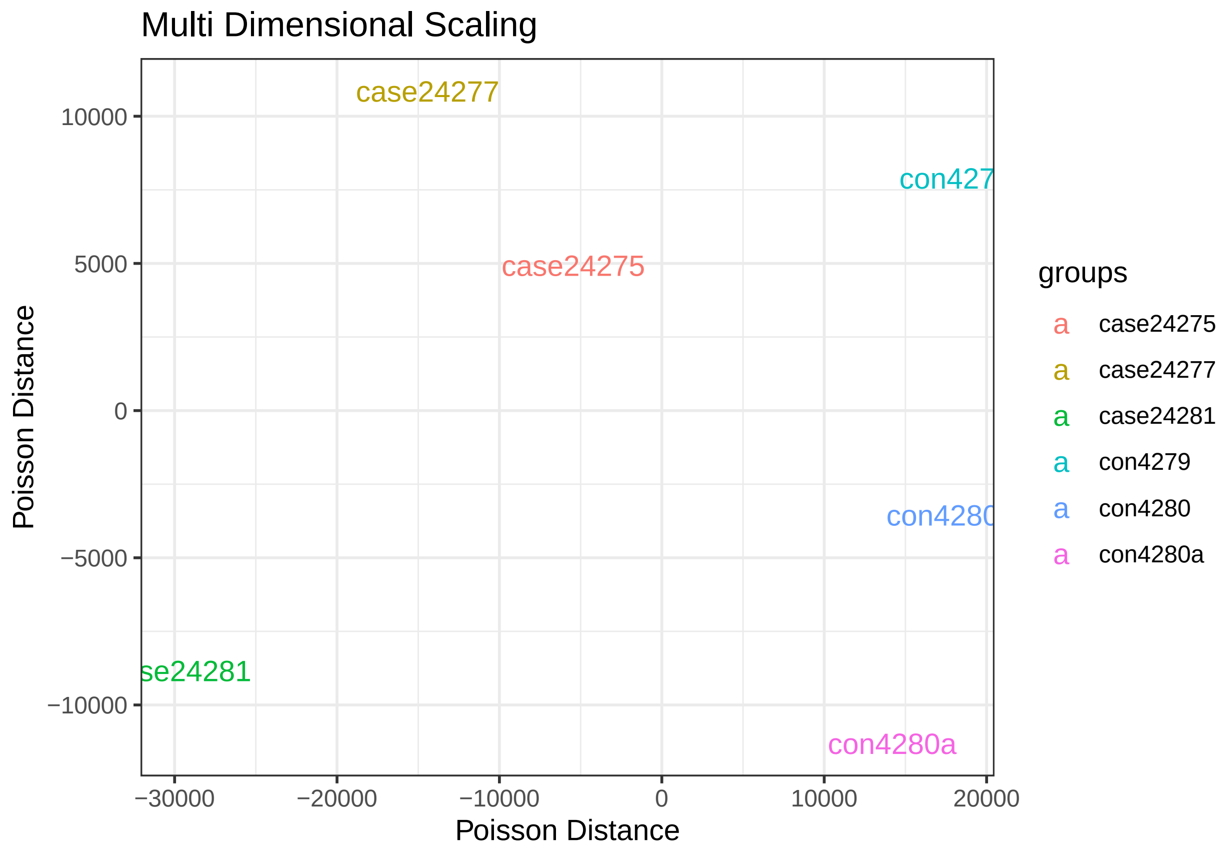
```
# Creating the objects required by ggplot for mds.
dds <- assay(ddsMat)
poisd <- PoissonDistance( t(dds), type = "deseq")
samplePoisDistMatrix <- as.matrix(poisd$dd)
mdsPoisData <- data.frame( cmdscale(samplePoisDistMatrix) )
```

```r
# Creating names for the coords.
names(mdsPoisData) <- c('x_coord', 'y_coord')

# Getting the columns as factors.
groups <- factor(rep(1:6, each=1),
                 labels = names(raw_data))

# Column name extraction.
coldata <- names(raw_data)

# Plotting the distance data in a 2d plot with 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()
```



## 3.5 Cleaning Data After examination of the case and control groups, there shouldn't be any samples removed. This would also not be possible, because at least 3 samples are required per group.

4 Discovering Differentialy Expressed Genes (DEGs)

Proceeding all the insight gained from plotting the data, it may now all be analysed in R. The purpose being is the discovery of DEGs, differentialy expressed genes. The earlier plots showed that there will most likely be plenty of those. The observed mutation also causes a frame shift, increasing the likelihood of DEGs greatly.. Before performing the analysis steps, the data will need to go through a pre-processing phase.

4.1 Pre-processing

First, the FPM, fragments per million mapped fragments, will be calculated for every row/gene.

```
# Applying the FPM calculation then creating a data frame out of it.
raw_data.fpm <- log2( (raw_data/ (colSums(raw_data) / 1e6 )) + 1)
```

There are quite a lot of inactive genes within the dataset. Filtering these out will help in further analysis. The paper does not provide a method for filtering out these genes. First, the most fitting method will have to be chosen.

Let's first see what the actual sum values are of all the genes.

```
# Make a column which sums up the log2 reads.
raw_data.fpm$sum = apply(X = raw_data.fpm, MARGIN = 1, FUN = sum)

# Calculating the percentage of genes with a total of 0 counts across all groups
sum(raw_data.fpm$sum == 0) / nrow(raw_data.fpm) * 100
```

```
## [1] 40.54921
```

The calculation above returns a value of 40.55. So 40.55% of the genes have not been expressed in any group or sample. Before applying other calculations to detect more inactive genes, it can be concluded that this 40% is surely inactive. It's therefore safe to remove.

There's not a definitive answer as to when a gene may be considered inactive and irrelevant. Discussion is still ongoing, but an answer that was observed multiple times is that an FPM sum above 0.5 indicates a statistically considerable gene. Let's see how much of the data is retained if everything below 0.5 would be removed.

```
sum(raw_data.fpm$sum > 0.5)
```

```
## [1] 21586
```

```
sum(raw_data.fpm$sum > 0.5) / nrow(raw_data.fpm) * 100
```

```
## [1] 37.02446
```

This would result in 21586 genes remaining for further analysis. Which is 37.02% of the original data. While the percentage is a little low, it's still 21 thousand genes, which is enough for analysis.

```
raw_data.fpm <- raw_data.fpm[raw_data.fpm$sum > 0.5,]
```

4.2 The Fold Change Value

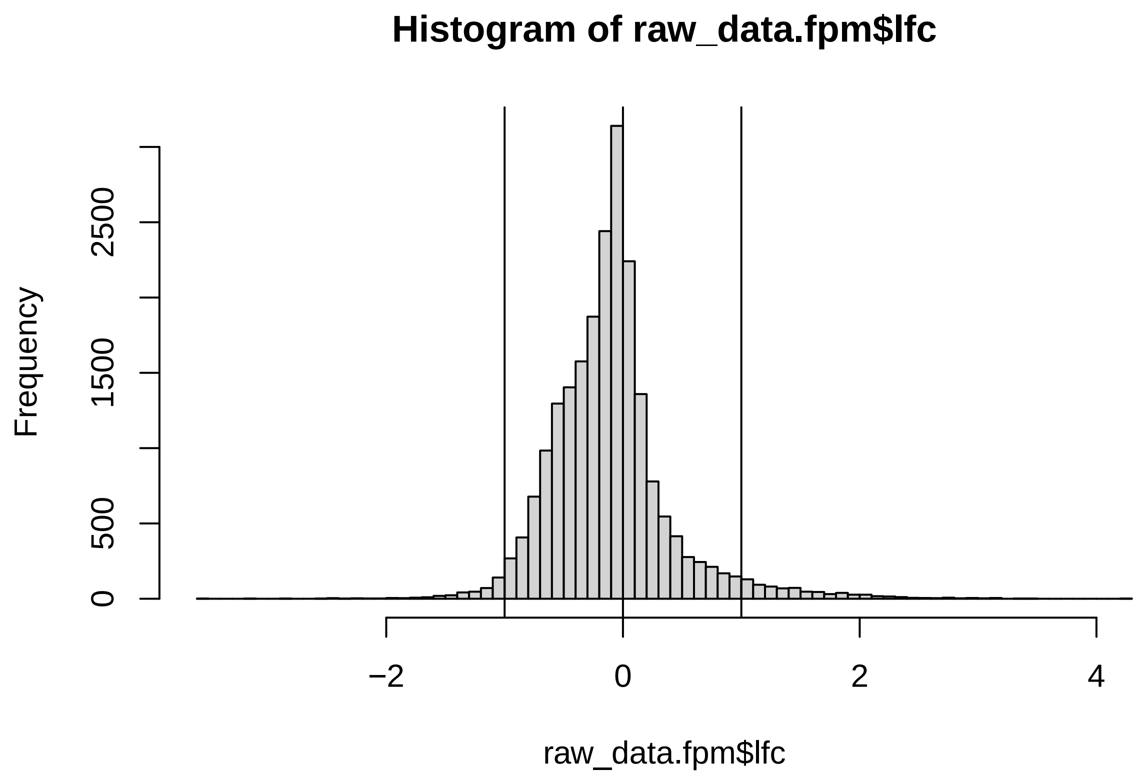
To gain insight into how the control and experiment groups differ in expression, a FC value will be calculated. First, to calculate the FC values, the averages will be calculated. These will be subtracted from eachother to get a LFC value. The FC value but with log2 applied.

```
# Calculating the means and savin them in columns
raw_data.fpm$case_avg = apply(X = raw_data.fpm[,1:3], MARGIN = 1, FUN = mean)
raw_data.fpm$con_avg = apply(X = raw_data.fpm[,4:6], MARGIN = 1, FUN = mean)

# Calculating the LFC values.
raw_data.fpm$lfc = raw_data.fpm$case_avg - raw_data.fpm$con_avg
```

Following this, a histogram may be made out of the new values.

```
hist(raw_data.fpm$lfc, breaks = 60)
abline(v = -1:1)
```



As can be seen in the histogram, there are quite some LFC values higher than 1, indicating that there is indeed increased expression on multiple genes.

4.3 Using Bioconductor Packages

It's now possible to perform T tests to determine which genes are significantly different in expression. Instead of doing this manually, bioconductor packages can be utilized. Because this experiment used edgeR, it will also be used here.

The raw_data data frame will need to be used again, since edgeR requires the raw counts. Before trying to detect DEGs, some filtering can be applied to make the results more significant. As discussed in 4.1, filtering out low counts tends to be beneficial. edgeR has a built in function which uses it's own filtering algorithm to remove these low counts. This will be done after the dataframe has been converted to the appropriate format, the DGElist

```r
# Defining the DGElist object.Group indicates which rows are cases/control.
dge <- DGEList(counts = raw_data, group = c(2,2,2,1,1,1))
# Showing the amount of rows read from the raw counts (All rows).
nrow(dge)
```

```
## [1] 58302
```

```r
# Marking the rows that should be kept because they have sufficient expression levels
keep <- filterByExpr(dge)
# Only keeping the rows which have sufficient expression. Showing the amount of rows kept.
dge <- dge[keep, , keep.lib.sizes=FALSE]
nrow(dge)
```

```
## [1] 16242
```

Now that the DGE object has been created and low counts have been filtered out, further normalization can be applied. In the case of this experiment, that wont be needed.

The classic edgeR pipeline will be followed here, since there's nothing that needs to be done to the count data after the filtering done above. The first step of the pipeline is to calculate gene dispersions. This can be done using the estimtaeDisp function.

```
dge <- calcNormFactors(dge)
dge <- estimateDisp(dge)
```

```
## Using classic mode.
```

```
dge
```

```
## An object of class "DGEList"
## $counts
##                 case24275 case24277 case24281 con4279 con4280 con4280a
## ENSG00000000003        23        30         8      11      43       31
## ENSG00000000419       778       910      1051     838     911     1113
## ENSG00000000457       378       438       389     441     772      738
## ENSG00000000460        44        51        28      58      61       65
## ENSG00000000938     14575     21109     27759    7164   11710    11846
## 16237 more rows ...
##
## $samples
##           group lib.size norm.factors
## case24275     2 26051564    1.0025402
## case24277     2 30362023    1.0110245
## case24281     2 28218458    0.8360335
## con4279       1 27399053    1.1195734
## con4280       1 31785502    1.0302009
## con4280a      1 31480407    1.0231477
##
## $common.dispersion
## [1] 0.07307054
##
## $trended.dispersion
## [1] 0.13107421 0.05345253 0.06133254 0.11722423 0.05223241
## 16237 more elements ...
##
## $tagwise.dispersion
## [1] 0.20109455 0.04470198 0.04952511 0.05586617 0.07798986
## 16237 more elements ...
##
## $AveLogCPM
## [1] -0.1807794  5.0152469  4.1503171  0.8412837  9.1385029
## 16237 more elements ...
##
## $trend.method
## [1] "locfit"
##
## $prior.df
## [1] 4.232264
##
## $prior.n
## [1] 1.058066
```

```
##
## $span
## [1] 0.2916128
```

Now that the dispersions have been calculated, testing for the DE genes may be performed. This is done by using the ExactTest function, which looks at the two groups and performs a t.test, to then determine the P value. The genes which get assigned a P value of less than 0.01 shall be stored in a results dataframe. 0.01 Was used in the paper. In order of lowest to highest P value. The are the adjusted P values.

```
et <- exactTest(dge, pair=c(1,2))
res <- topTags(et, n = Inf, p = 0.01)$table
```

Now the significant DEGs have been selected and assigned with LogFC values. Further analysis can be done by plotting the results.

5 Data Analysis and Visualization

There's multiple ways to display the DEGs. Since the data set consist of gene counts for every gene, there'll be too many rows to properly display for some visualizations. It might be beneficial to therefore look at the pathways to which these important genes belong to. But, a volcano plot can still be made out of the initial results.

5.1 Volcano Plot

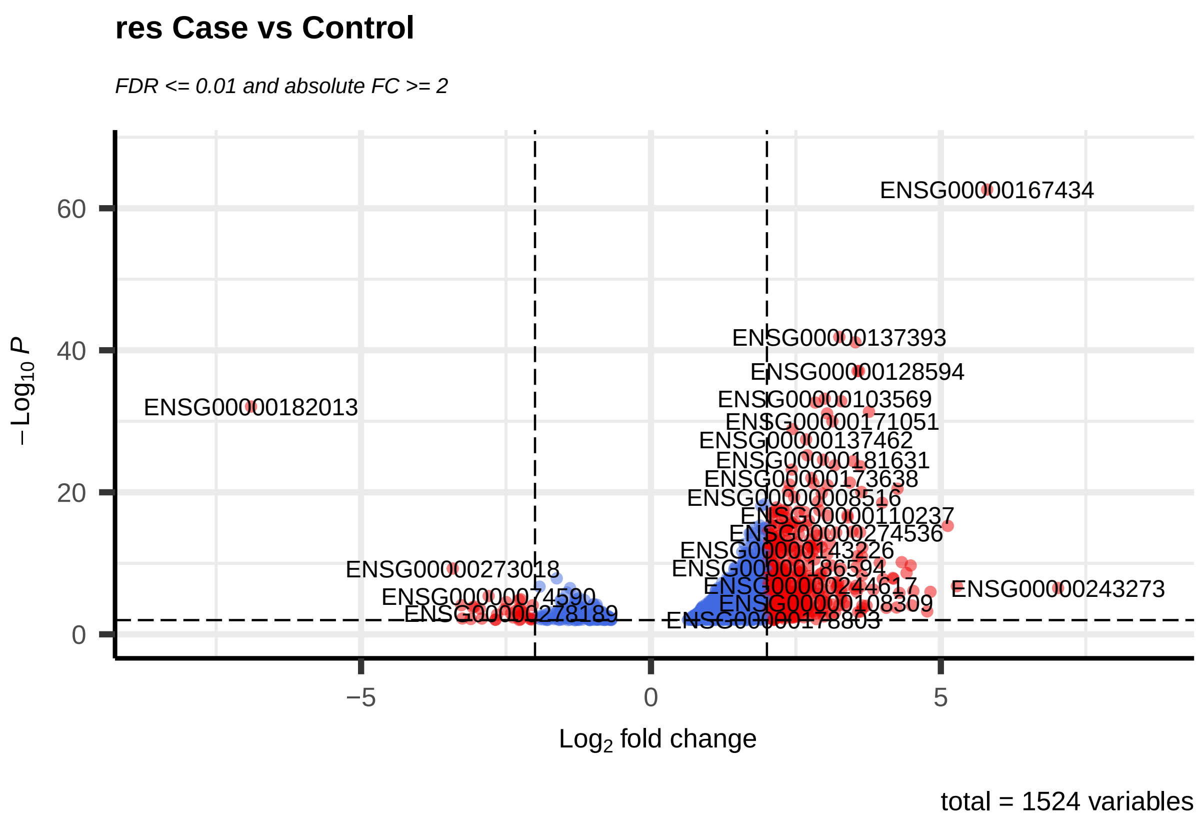
A volcano plot can show which DEGs are of most importance with the given LogFC values. The paper also contains a volcano plot, which has a FC cutoff of 2. This cutoff point will also be used here.

```
sum(res$logFC < 0)
```

```
## [1] 198
```

```
deseq.volcano <- function(res, datasetName) {
  return(EnhancedVolcano(res, x = 'logFC', y = 'FDR',
                         lab=rownames(res),
                         title = paste(datasetName, "Case vs Control"),
                         subtitle = bquote(italic('FDR <= 0.01 and absolute FC >= 2')),
                         # 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 = 0.01, FCcutoff = 2))
}

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



There's a significant upstream regulation of the genes in the case samples, as can be seen in the volcano plot. So far, the analysis has been performed on the individual DEGs. While these can be examined on their own, it would be more insightful at this point to look at the relevant pathways the genes may be part of.

5.2 Pathway Analysis

To find which pathways the DEGs belong to, DAVID will be used. DAVID requires the user to upload a list or file with the genes. Multiple different identifiers for the genes can be used, including ensembl ID. Because all the genes in this experiment have been annotated by their ensembl IDs.

The row names will first be extracted and sent to the DAVID website.

```
ensembl <- row.names(res)
write.table(ensembl, file = "ensembl.txt", row.names = FALSE, col.names = FALSE, quote = FALSE)
```

The text file created can be uploaded to DAVID. Extracting the pathway results into a chart allows a representing file to be downloaded. The contents of the file are as follows:

```
pathways <- read.table('chart.txt', header = TRUE, sep = '\t')
pathways[1,]
```

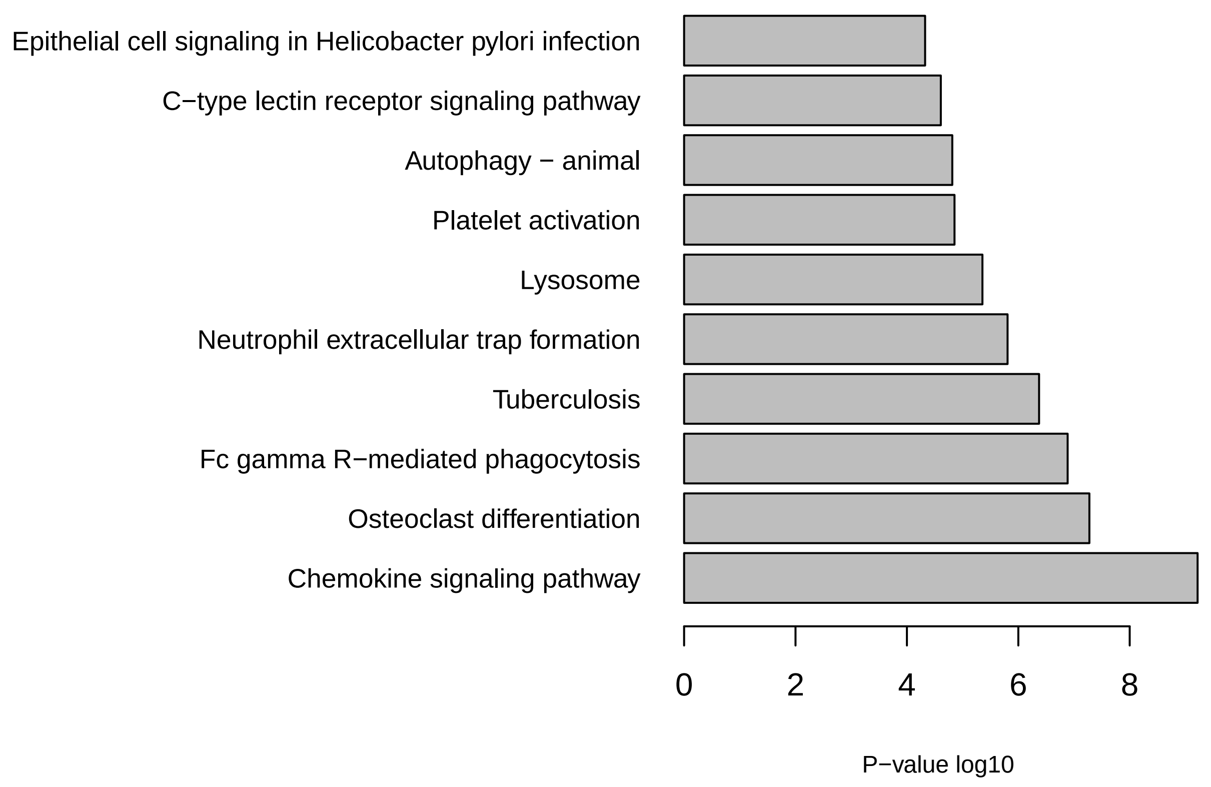
```
##       Category                                 Term Count        X.       PValue
## 1 KEGG_PATHWAY hsa04062:Chemokine signaling pathway    42 3.043478 6.042316e-10
##
## 1 ENSG00000183625, ENSG00000177885, ENSG00000180871, ENSG00000163737, ENSG00000141506, ENSG0000014190
##   List.Total Pop.Hits Pop.Total Fold.Enrichment   Bonferroni   Benjamini
## 1        612      192      8112         2.89951 1.903329e-07 1.90333e-07
##            FDR
## 1 1.534748e-07
```

As can be seen above, these are the five most significant pathway entries in this experiment. Pathway analysis, when done diligently, can be used to draw some conclusions. The following code will be dedicated to trying to gain more understanding in how these pathways affect the phenotype. Some extra columns will first be added for clarity.

```
# Copying the ID from the term column.
pathways$pathway.id <- sapply(strsplit(pathways$Term, ":"), function(splitted) return(splitted[1]))
# Copying the name from the term column.
pathways$pathway.name <- sapply(strsplit(pathways$Term, ":"), function(splitted) return(splitted[2]))
# Assigning the X marked column name with an actual name.
names(pathways)[4] <- "Percentage"
```

The most 10 most significant pathways may now be displayed in a simple plot.

```
# Adjusting the margins because of the long names
par(mar=c(4,18,2,1))
barplot(height = -log10(pathways$PValue)[1:10],
        names.arg = pathways$pathway.name[1:10],
        horiz = TRUE, las=1, xlab = 'P-value log10',
        cex.names = 0.8,
        cex.lab = 0.75)
```



While the results may be promising, not only the top 10 results should be considered. There's another 81 relevant pathways. Pathview from bioconductor may offer a more complete picture for an entire pathway analsysis.

```
# Taking the data from the human pathways.
data("paths.hsa")
# Displaying 5 of the pathways.
```

```r
pander(head(paths.hsa, n=5))
```

Table 1: Table continues below

hsa00010	hsa00020	hsa00030	hsa00040	hsa00051
Glycolysis / Gluconeogenesis	Citrate cycle (TCA cycle)	Pentose phosphate pathway	Pentose and glucuronate interconversions	Fructose and mannose metabolism

Showing an example of some of the the pathways which are available in humans(First 5). The structure of the paths data is as follows:

```r
data(gene.idtype.list);
pander(gene.idtype.list)
```

SYMBOL, *GENENAME*, *ENSEMBL*, *ENSEMBLPROT*, *UNIGENE*, *UNIPROT*, *ACCNUM*, *ENSEMBLTRANS*, *REFSEQ*, *ENZYME*, *TAIR*, *PROSITE* and *ORF*

Now the experiments data will need to be prepared for the package operations. The data set the experiment provides dose not have entrez IDs. The ensembl IDs will have to converted to entrez equivelants.

```r
# Selecting for rows with an FDR lower than 0.01.
res.logFC <- subset(res, FDR < 0.01, select = logFC)
# The ID problem being resolved by the id2eg function.
res.logFC$entrez <- id2eg(ids = rownames(res.logFC), org = "Hs", category = "ENSEMBL")[,2]
```

```
## 'select()' returned 1:many mapping between keys and columns

## [1] "Note: 155 of 1524 unique input IDs unmapped."
```

```r
# The above function does not remove NA. Doing it manually here.
res.logFC.ent <- subset(res.logFC, !(is.na(entrez)) )
# Now the rownames may be extracted and used.
rownames(res.logFC.ent) <- res.logFC.ent$entrez
# The row names are now entrez, so the column gets removed.
res.logFC.ent$entrez = NULL
```

The prepared data may be given to path view now. Significant up-regulation had been observed in genes relevant to signaling pathways. The two highest count pathways are related to cancer and chemokine signaling. Both of which are quite relevant to the paper. These will therefore be used in pathview.

```r
pathview(gene.data=res.logFC.ent,
         pathway.id = c("05200","04062"),
         species="hsa"
)
```

```
## 'select()' returned 1:1 mapping between keys and columns

## Info: Working in directory C:/Users/solai/Documents/thema7/r_files

## Info: Writing image file hsa05200.pathview.png

## Info: some node width is different from others, and hence adjusted!

## 'select()' returned 1:1 mapping between keys and columns
## Info: Working in directory C:/Users/solai/Documents/thema7/r_files

## Info: Writing image file hsa04062.pathview.png
```

The resulting pathway analysis for hsa04062 and hsa05200 can be seen at the end of the document, because of R markdown delays.

There's several affected molecules in these pathways, showing significant up regulation. The coloring is based on the corresponding logFC values.

To close off this experiment, the topology of how gene interactions will be examined. The SPIA Bioconductor package is used in the following examinations.

```
## [1] "Note: 23336 of 58302 unique input IDs unmapped."

##
## Done pathway 1 : RNA transport..

##
## Done pathway 2 : RNA degradation..

##
## Done pathway 3 : PPAR signaling pathway..

##
## Done pathway 4 : Fanconi anemia pathway..

##
## Done pathway 5 : MAPK signaling pathway..

##
## Done pathway 6 : ErbB signaling pathway..

##
## Done pathway 7 : Calcium signaling pathway..

##
## Done pathway 8 : Cytokine-cytokine receptor int..

##
## Done pathway 9 : Chemokine signaling pathway..

##
## Done pathway 10 : NF-kappa B signaling pathway..

##
## Done pathway 11 : Phosphatidylinositol signaling..

##
## Done pathway 12 : Neuroactive ligand-receptor in..

##
## Done pathway 13 : Cell cycle..

##
## Done pathway 14 : Oocyte meiosis..

##
## Done pathway 15 : p53 signaling pathway..
## Done pathway 16 : Sulfur relay system..

##
## Done pathway 17 : SNARE interactions in vesicula..
```

```
##
## Done pathway 18 : Regulation of autophagy..

##
## Done pathway 19 : Protein processing in endoplas..

##
## Done pathway 20 : Lysosome..

##
## Done pathway 21 : mTOR signaling pathway..

##
## Done pathway 22 : Apoptosis..

##
## Done pathway 23 : Vascular smooth muscle contrac..

##
## Done pathway 24 : Wnt signaling pathway..

##
## Done pathway 25 : Dorso-ventral axis formation..

##
## Done pathway 26 : Notch signaling pathway..

##
## Done pathway 27 : Hedgehog signaling pathway..

##
## Done pathway 28 : TGF-beta signaling pathway..

##
## Done pathway 29 : Axon guidance..

##
## Done pathway 30 : VEGF signaling pathway..

##
## Done pathway 31 : Osteoclast differentiation..

##
## Done pathway 32 : Focal adhesion..

##
## Done pathway 33 : ECM-receptor interaction..
## Done pathway 34 : Cell adhesion molecules (CAMs)..
## Done pathway 35 : Adherens junction..

##
## Done pathway 36 : Tight junction..

##
## Done pathway 37 : Gap junction..

##
## Done pathway 38 : Complement and coagulation cas..

##
## Done pathway 39 : Antigen processing and present..

##
## Done pathway 40 : Toll-like receptor signaling p..
```

```
##
## Done pathway 41 : NOD-like receptor signaling pa..
##
## Done pathway 42 : RIG-I-like receptor signaling ..
##
## Done pathway 43 : Cytosolic DNA-sensing pathway..
##
## Done pathway 44 : Jak-STAT signaling pathway..
##
## Done pathway 45 : Natural killer cell mediated c..
##
## Done pathway 46 : T cell receptor signaling path..
##
## Done pathway 47 : B cell receptor signaling path..
##
## Done pathway 48 : Fc epsilon RI signaling pathwa..
##
## Done pathway 49 : Fc gamma R-mediated phagocytos..
##
## Done pathway 50 : Leukocyte transendothelial mig..
##
## Done pathway 51 : Intestinal immune network for ..
##
## Done pathway 52 : Circadian rhythm - mammal..
##
## Done pathway 53 : Long-term potentiation..
##
## Done pathway 54 : Neurotrophin signaling pathway..
##
## Done pathway 55 : Retrograde endocannabinoid sig..
##
## Done pathway 56 : Glutamatergic synapse..
##
## Done pathway 57 : Cholinergic synapse..
##
## Done pathway 58 : Serotonergic synapse..
##
## Done pathway 59 : GABAergic synapse..
##
## Done pathway 60 : Dopaminergic synapse..
##
## Done pathway 61 : Long-term depression..
```

```
##
## Done pathway 62 : Olfactory transduction..
##
## Done pathway 63 : Taste transduction..
##
## Done pathway 64 : Phototransduction..
##
## Done pathway 65 : Regulation of actin cytoskelet..
##
## Done pathway 66 : Insulin signaling pathway..
##
## Done pathway 67 : GnRH signaling pathway..
##
## Done pathway 68 : Progesterone-mediated oocyte m..
##
## Done pathway 69 : Melanogenesis..
##
## Done pathway 70 : Adipocytokine signaling pathwa..
##
## Done pathway 71 : Type II diabetes mellitus..
##
## Done pathway 72 : Type I diabetes mellitus..
##
## Done pathway 73 : Maturity onset diabetes of the..
##
## Done pathway 74 : Aldosterone-regulated sodium r..
##
## Done pathway 75 : Endocrine and other factor-reg..
##
## Done pathway 76 : Vasopressin-regulated water re..
##
## Done pathway 77 : Salivary secretion..
##
## Done pathway 78 : Gastric acid secretion..
##
## Done pathway 79 : Pancreatic secretion..
##
## Done pathway 80 : Carbohydrate digestion and abs..
##
## Done pathway 81 : Bile secretion..
##
## Done pathway 82 : Mineral absorption..
```

```
##
## Done pathway 83 : Alzheimer's disease..
##
## Done pathway 84 : Parkinson's disease..
##
## Done pathway 85 : Amyotrophic lateral sclerosis ..
##
## Done pathway 86 : Huntington's disease..
##
## Done pathway 87 : Prion diseases..
##
## Done pathway 88 : Cocaine addiction..
##
## Done pathway 89 : Amphetamine addiction..
##
## Done pathway 90 : Morphine addiction..
##
## Done pathway 91 : Alcoholism..
##
## Done pathway 92 : Bacterial invasion of epitheli..
##
## Done pathway 93 : Vibrio cholerae infection..
##
## Done pathway 94 : Epithelial cell signaling in H..
##
## Done pathway 95 : Pathogenic Escherichia coli in..
##
## Done pathway 96 : Shigellosis..
##
## Done pathway 97 : Salmonella infection..
##
## Done pathway 98 : Pertussis..
##
## Done pathway 99 : Legionellosis..
##
## Done pathway 100 : Leishmaniasis..
##
## Done pathway 101 : Chagas disease (American trypa..
##
## Done pathway 102 : African trypanosomiasis..
##
## Done pathway 103 : Malaria..
```

```
##
## Done pathway 104 : Toxoplasmosis..
##
## Done pathway 105 : Amoebiasis..
##
## Done pathway 106 : Staphylococcus aureus infectio..
##
## Done pathway 107 : Tuberculosis..
##
## Done pathway 108 : Hepatitis C..
##
## Done pathway 109 : Measles..
##
## Done pathway 110 : Influenza A..
##
## Done pathway 111 : HTLV-I infection..
##
## Done pathway 112 : Herpes simplex infection..
##
## Done pathway 113 : Epstein-Barr virus infection..
##
## Done pathway 114 : Pathways in cancer..
##
## Done pathway 115 : Transcriptional misregulation ..
##
## Done pathway 116 : Viral carcinogenesis..
##
## Done pathway 117 : Colorectal cancer..
##
## Done pathway 118 : Renal cell carcinoma..
##
## Done pathway 119 : Pancreatic cancer..
##
## Done pathway 120 : Endometrial cancer..
##
## Done pathway 121 : Glioma..
##
## Done pathway 122 : Prostate cancer..
##
## Done pathway 123 : Thyroid cancer..
##
## Done pathway 124 : Basal cell carcinoma..
```

```
##
## Done pathway 125 : Melanoma..

##
## Done pathway 126 : Bladder cancer..

##
## Done pathway 127 : Chronic myeloid leukemia..

##
## Done pathway 128 : Acute myeloid leukemia..

##
## Done pathway 129 : Small cell lung cancer..

##
## Done pathway 130 : Non-small cell lung cancer..

##
## Done pathway 131 : Asthma..
## Done pathway 132 : Autoimmune thyroid disease..

##
## Done pathway 133 : Systemic lupus erythematosus..

##
## Done pathway 134 : Rheumatoid arthritis..

##
## Done pathway 135 : Allograft rejection..

##
## Done pathway 136 : Graft-versus-host disease..

##
## Done pathway 137 : Arrhythmogenic right ventricul..

##
## Done pathway 138 : Dilated cardiomyopathy..

##
## Done pathway 139 : Viral myocarditis..
```

This concluded the experiment. Now, to show some of the results of the SPIA function:

```
spia_result[1:5,1:4]
```

```
##                                Name    ID pSize NDE
## 1      Chemokine signaling pathway 04062   184  40
## 2                     Tuberculosis 05152   175  35
## 3 Fc gamma R-mediated phagocytosis 04666    94  26
## 4       Osteoclast differentiation 04380   129  29
## 5                    Leishmaniasis 05140    68  17
```

5.3 Pathview Results

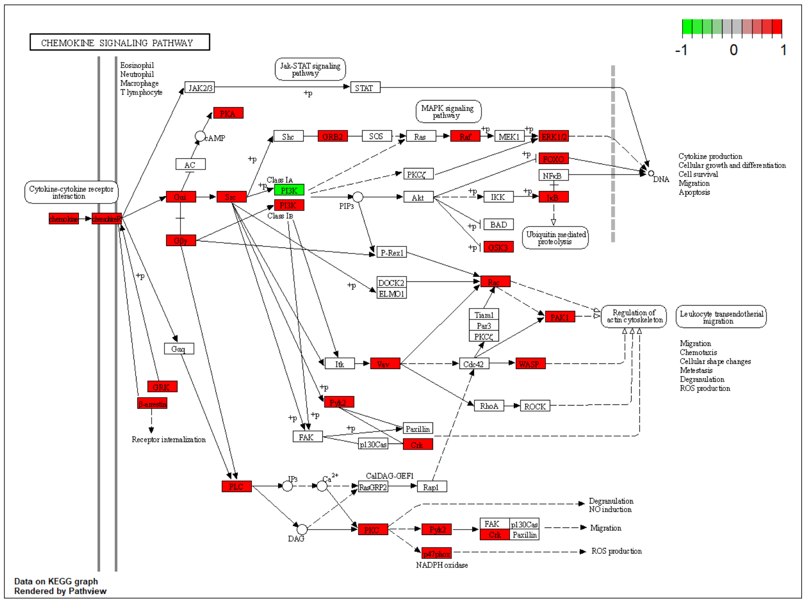


Figure 1: Pathway

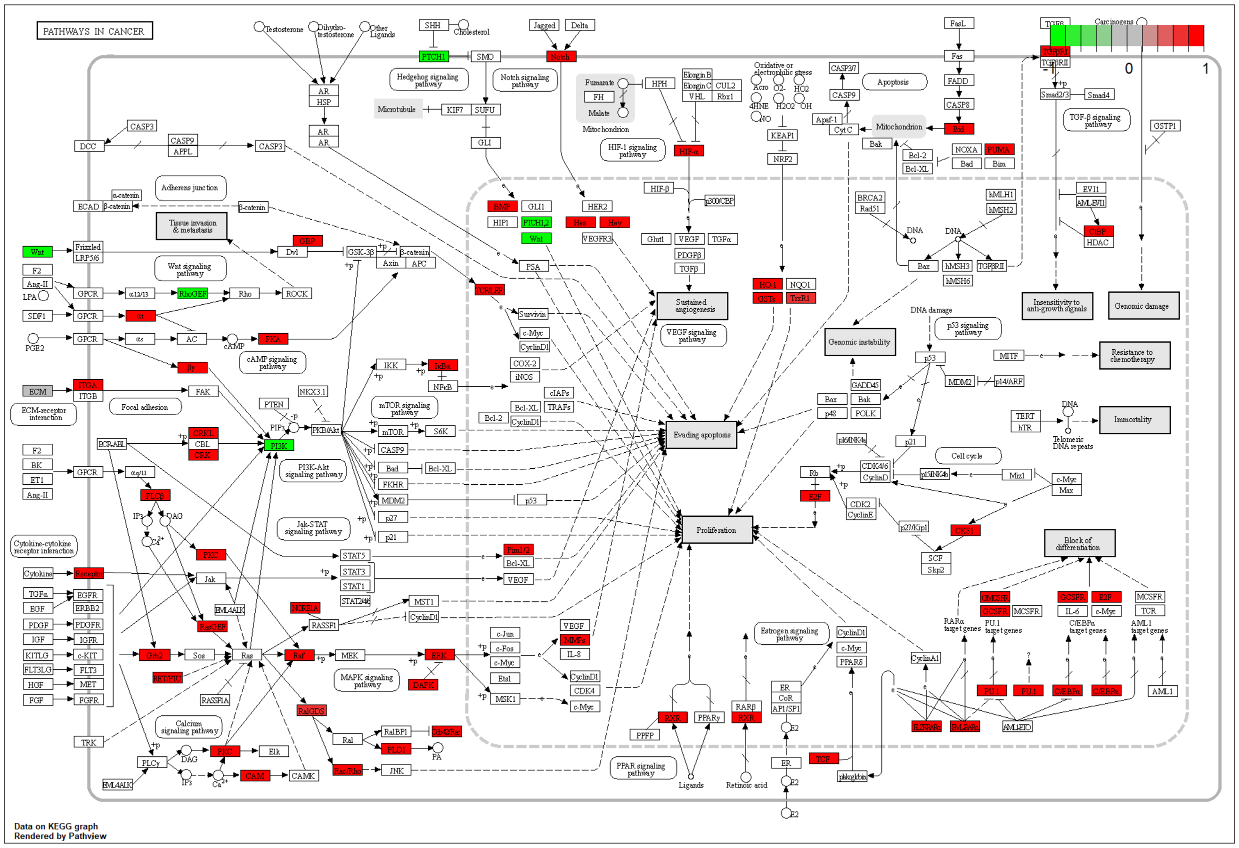


Figure 2: Pathway