Bioinformatics Resources Project 2023

15/01/2024

```
knitr::opts_chunk$set(echo = TRUE)
library("ggplot2")
library("biomaRt")
library("edgeR")
library("tidyverse")
library("GenomicFeatures")
library("clusterProfiler")
library("org.Hs.eg.db")
library("enrichplot")
library("pathview")
library("ggnewscale")
library("fgsea")
library("PWMEnrich")
library("PWMEnrich.Hsapiens.background")
library("MotifDb")
library("igraph")
library("statmod")
library("devtools")
library("rbioapi")
library("rgl")
```

We have chosen the RNA-seq count data extracted from the lung squamous cell carcinoma dataset, knowing that from the original TCGA data 50 cases (tumor samples) and 50 controls (normal samples) were randomly selected.

1. Load the RData file.

```
path <- getwd()
load(paste(path, "/Lung_squamous_cell_carcinoma.RData", sep=""))
```

We can see that three different dataframes have been uploaded. Let us inspect them separately.

```
head(raw_counts_df, 2)
```

```
##                 TCGA-85-A4QQ-01A TCGA-94-7557-01A TCGA-98-A53A-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-NC-A5HR-01A TCGA-56-7730-11A TCGA-33-6737-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-22-5491-11A TCGA-66-2744-01A TCGA-22-5482-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
```

```
##                 TCGA-77-7338-11A TCGA-56-A4BW-01A TCGA-43-3394-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-56-8083-11A TCGA-66-2768-01A TCGA-NC-A5HP-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-39-5040-11A TCGA-77-8007-11A TCGA-56-8201-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-56-7731-11A TCGA-56-7582-11A TCGA-43-3394-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-34-7107-11A TCGA-21-5786-01A TCGA-77-8148-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-77-8144-01A TCGA-90-6837-11A TCGA-33-4587-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-56-8628-01A TCGA-22-5478-11A TCGA-63-A5MR-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-98-8022-01A TCGA-LA-A7SW-01A TCGA-56-8623-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-56-8082-11A TCGA-22-5481-11A TCGA-43-6771-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-34-8454-11A TCGA-L3-A524-01A TCGA-34-5239-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-98-A539-01A TCGA-85-7710-11A TCGA-43-7657-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-92-7340-11A TCGA-33-AASL-01A TCGA-77-7337-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-56-7580-11A TCGA-43-6773-11A TCGA-60-2697-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-92-8063-01A TCGA-43-7658-11A TCGA-58-8386-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-56-A4BX-01A TCGA-79-5596-01A TCGA-51-4080-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-37-4133-01A TCGA-56-7579-11A TCGA-39-5031-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-85-7710-01A TCGA-77-8008-11A TCGA-60-2723-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-43-5670-11A TCGA-60-2715-01A TCGA-56-8309-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
```

```
##                 TCGA-51-6867-01A TCGA-NC-A5HE-01A TCGA-18-4721-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-22-4609-11A TCGA-51-4079-11A TCGA-22-5483-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-68-A59I-01A TCGA-92-8065-01A TCGA-22-5471-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-22-4593-11A TCGA-37-4132-01A TCGA-J1-A4AH-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-90-A59Q-01A TCGA-63-A5MV-01A TCGA-22-5489-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-77-7142-11A TCGA-77-7138-11A TCGA-66-2781-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-60-2709-11A TCGA-22-5472-11A TCGA-77-8140-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-18-3410-01A TCGA-O2-A52V-01A TCGA-51-4081-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-90-7767-11A TCGA-43-6770-01A TCGA-66-2800-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-92-8064-01A TCGA-85-A4QR-01A TCGA-56-7222-11A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-XC-AA0X-01A TCGA-77-7335-11A TCGA-22-4609-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-43-6143-11A TCGA-43-6647-11A TCGA-66-2793-01A
## ENSG00000278704                0                0                0
## ENSG00000277400                0                0                0
##                 TCGA-85-8351-01A
## ENSG00000278704                0
## ENSG00000277400                0
```

```
cat("\nDimension: ", dim(raw_counts_df))
```

```
##
## Dimension:  62940 100
```

The data-frame *raw_counts_df* represents the raw RNA-seq counts (sequence reads per gene/how many reads have mapped to each gene): the rows correspond to the Ensembl Gene IDs (ex: ENSG00000278704) and the columns to the tumor and normal samples from which the reads were counted for each gene (ex: TCGA-22-4609-01A). 62940 genes are included in the dataframe and 100 samples are taken into account, 50 cases and 50 controls, respectively.

```
head(c_anno_df, 5)
```

```
##              sample condition
## 11 TCGA-85-A4QQ-01A      case
## 13 TCGA-94-7557-01A      case
## 16 TCGA-98-A53A-01A      case
## 21 TCGA-NC-A5HR-01A      case
## 26 TCGA-56-7730-11A   control
```

```
cat("\nDimension: ", dim(c_anno_df))
```

```
##
## Dimension:  100 2
```

The dataframe *c_anno_df* specifies the condition (*condition* column) corresponding to each sample name (*sample* column), either "case" or "control". The number of rows (samples) is 100, properly matching the number of columns (samples) of *raw_counts_df*.

```
head(r_anno_df, 5)
```

```
##                 ensembl_gene_id external_gene_name length
## ENSG00000278704 ENSG00000278704                      2236
## ENSG00000277400 ENSG00000277400                     61428
## ENSG00000274847 ENSG00000274847              MAFIP  61461
## ENSG00000277428 ENSG00000277428              Y_RNA    100
## ENSG00000276256 ENSG00000276256                      6225
```

```
cat("\nDimension: ", dim(r_anno_df))
```

```
##
## Dimension:  62940 3
```

The dataframe *r_anno_df* maps each *Ensembl gene ID* from *raw_counts_df* to its *external gene name*, or gene symbol, and *length* (the number of bases in the primary transcript prior to splicing). So the number of rows is 62940.

2. Extract protein-coding genes

Since we want to perform differential gene expression analysis, we are interested only in protein-coding genes, so we need to filter *raw_counts_df* according to this experimental choice. The R package **biomaRt** allows us to connect to a specific biomaRt database, in this case the ENSEMBL database, and to select the Human genes dataset.

```
ensembl<- useEnsembl(biomart = "ensembl", dataset = "hsapiens_gene_ensembl", mirror="useast")
```

The *getBM()* function is the main query function in biomaRt and it has four main arguments: *attributes*, *filters*, *values* and *mart*. We have already set the latter and assigned it to the variable *ensembl*. The coding/non-coding information is included in the attribute *gene_biotype*. We have chosen to filter the Human genes dataset according to the *ensembl_gene_id* attribute, which we have at disposal to identify the genes belonging to our starting dataset related to lung squamous cell carcinoma.

```
anno_query <- getBM(attributes = c("ensembl_gene_id", "gene_biotype"),
                    filters = ("ensembl_gene_id"),
                    values = r_anno_df$ensembl_gene_id,
                    mart = ensembl)
head(anno_query,5)
```

```
##   ensembl_gene_id   gene_biotype
## 1 ENSG00000056678         lncRNA
## 2 ENSG00000096150 protein_coding
## 3 ENSG00000096155 protein_coding
## 4 ENSG00000096171 protein_coding
## 5 ENSG00000111971 protein_coding
```

```
cat("\nDimension: ", dim(anno_query))
```

```
##
## Dimension:  62872 2
```

All the genes that were recognized by *Ensembl* are now matched to their type so we can subset the lung squamous cell carcinoma dataset to include only protein-coding genes.

```
coding_query <- anno_query[which(anno_query$gene_biotype == "protein_coding"), ]
cat("\nDimension: ", dim(coding_query))
```

```
##
## Dimension:  22138 2
```

```
head(coding_query,5)
```

```
##   ensembl_gene_id   gene_biotype
## 2 ENSG00000096150 protein_coding
## 3 ENSG00000096155 protein_coding
## 4 ENSG00000096171 protein_coding
## 5 ENSG00000111971 protein_coding
## 6 ENSG00000112459 protein_coding
```

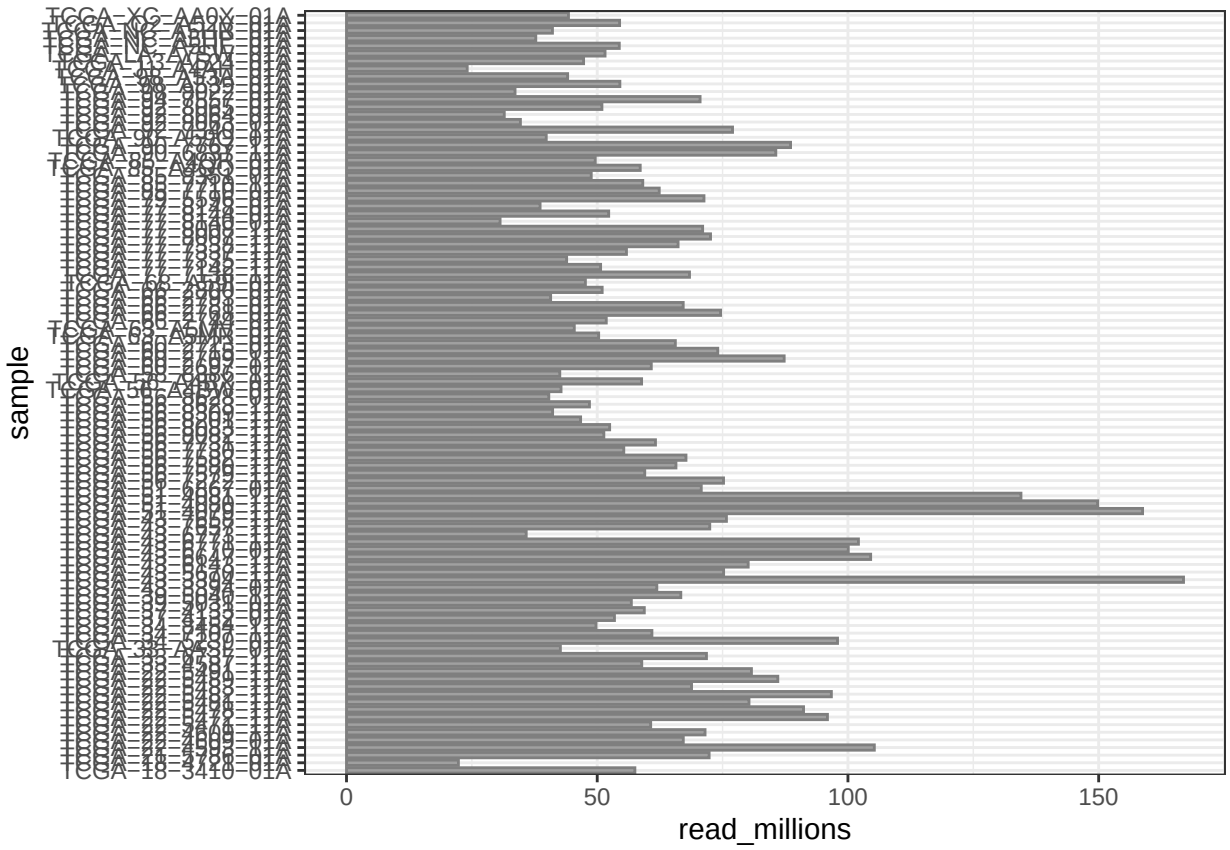
The number of genes has approximately become ~1/3 of the initial number of genes. After merging the *coding_query* result with the starting dataset, we have the updated version of the dataset, containing only protein coding genes (*clean_coding_raw*).

```
tmp <-raw_counts_df
coding_raw <- merge(coding_query,tmp, by.y = 'row.names', by.x = "ensembl_gene_id")
clean_coding_raw <- coding_raw[1:nrow(coding_raw),3:ncol(coding_raw)]
row.names(clean_coding_raw) <- coding_raw$ensembl_gene_id
```

3.Perform differential expression analysis

The dataset contains standard counts, so the number of reads X_i for each gene transcript i in N given samples/replicates. We need to obtain the counts per million (CPM, *read_millions*), so we can check the library size of each sample. We have generated a bar plot having the number of reads (per million) in the x-axis and the samples in the y-axis. In this way, we can qualitatively asses a threshold for the necessary number of reads, valid for all samples, averaged across all gene transcripts (for each sample).

```
size_raw <- data.frame("sample" = colnames(clean_coding_raw),
                       "read_millions" = colSums(clean_coding_raw/1000000))
ggplot(data=size_raw, aes(sample,read_millions))+
  geom_bar(stat="identity", fill="grey50",colour="grey50", width=0.7, alpha=0.7) +
  coord_flip() + theme_bw()
```



We are explicitly asked to filter raw counts data retaining only genes with a raw count > 20 in at least 5 Cases or 5 Control samples. To do so, thanks to *apply()*, we use the *by()* function on each row (gene) of the clean dataset (with only protein-coding genes) to divide case samples from control samples and sum only the number of case samples and control samples (separately), that record (for each gene) a number of reads greater than the threshold. So, for each gene, we have two numbers, respectively for case and control.

```
threshold<-20
cat("\nType of condition: ", typeof(c_anno_df$condition))
```

```
##
## Type of condition:  character
```

```
# Convert condition in factor (now it is a character) to use by() function
c_anno_df$condition <- as.factor(c_anno_df$condition)
c_anno_df$condition <- factor(c_anno_df$condition, levels=c('control', 'case'))
cat("\nType of condition: ", typeof(c_anno_df$condition))
```

```
##
## Type of condition:  integer
```

```
cat("\n Is condition a factor? ",is.factor(c_anno_df$condition))
```

```
##
##  Is condition a factor?  TRUE
```

```
filter_vec <- apply(clean_coding_raw,1,
    function(y) (by(y, c_anno_df$condition, function(x) sum(x > threshold))))
cat("\n")
```

```
print(filter_vec[,1250:1254])
```

```
##         ENSG00000074276 ENSG00000074317 ENSG00000074319 ENSG00000074356
## control              21               3              50              50
## case                 41              23              50              50
##         ENSG00000074370
## control              50
## case                 50
```

```
table(filter_vec)
```

```
## filter_vec
##     0     1     2     3     4     5     6     7     8     9    10    11    12
##  8761   645   364   282   222   197   162   136   147   133   128   124   130
##    13    14    15    16    17    18    19    20    21    22    23    24    25
##   100   103    98    95    87    97    98    95    93    91    91    89   106
##    26    27    28    29    30    31    32    33    34    35    36    37    38
##    90   103   106    96   109    96    92   114   108   110   110   134   136
##    39    40    41    42    43    44    45    46    47    48    49    50
##   167   137   162   168   196   200   243   255   367   493   918 26892
```

Since in the next step we need to pick at least 5 Case or 5 Control samples, we now have to make an arbitrary choice between the two numbers: taking the maximum or minimum (for equal values the default choice is the first element, so control) will impact the downstream results, since we do not know which condition is chosen for each gene.

```
filter_vec_min <- apply(clean_coding_raw,1,
    function(y) min(by(y, c_anno_df$condition, function(x) sum(x > threshold))))
filter_vec_min[1250:1254]
```

```
## ENSG00000074276 ENSG00000074317 ENSG00000074319 ENSG00000074356 ENSG00000074370
##              21               3              50              50              50
```

```
filter_vec_max <- apply(clean_coding_raw,1,
    function(y) max(by(y, c_anno_df$condition, function(x) sum(x > threshold))))
filter_vec_max[1250:1254]
```

```
## ENSG00000074276 ENSG00000074317 ENSG00000074319 ENSG00000074356 ENSG00000074370
##              41              23              50              50              50
```

```
repl_thr<- 5

ct_min <-table(filter_vec_min)
ct_min
```

```
## filter_vec_min
##     0     1     2     3     4     5     6     7     8     9    10    11    12
##  4951   272   187   127   109   122    76    75    79    58    76    62    73
##    13    14    15    16    17    18    19    20    21    22    23    24    25
##    48    55    48    50    48    61    47    41    52    59    54    51    61
##    26    27    28    29    30    31    32    33    34    35    36    37    38
##    49    56    54    53    60    48    50    70    62    60    63    73    75
##    39    40    41    42    43    44    45    46    47    48    49    50
##   103    70    86    97   115   113   145   164   210   303   579 12438
```

```
ct_min_df <- as.data.frame(ct_min)
cat("\nNumber of genes with 5 or more samples (case or control) with a number of reads > 20 (min): ",
    sum(ct_min_df[ct_min_df$filter_vec_min[repl_thr+1:51],2], na.rm = TRUE))
```

```
##
## Number of genes with 5 or more samples (case or control) with a number of reads > 20 (min):  16492
```

```
cat("\n")
```

```
ct_max <-table(filter_vec_max)
ct_max
```

```
## filter_vec_max
##     0     1     2     3     4     5     6     7     8     9    10    11    12
##  3810   373   177   155   113    75    86    61    68    75    52    62    57
##    13    14    15    16    17    18    19    20    21    22    23    24    25
##    52    48    50    45    39    36    51    54    41    32    37    38    45
##    26    27    28    29    30    31    32    33    34    35    36    37    38
##    41    47    52    43    49    48    42    44    46    50    47    61    61
##    39    40    41    42    43    44    45    46    47    48    49    50
##    64    67    76    71    81    87    98    91   157   190   339 14454
```

```
ct_max_df <- as.data.frame(ct_max)
cat("\nNumber of genes with 5 or more samples (case or control) with a number of reads > 20 (max): ",
    sum(ct_max_df[ct_max_df$filter_vec_max[repl_thr+1:51],2],na.rm = TRUE))
```

```
##
## Number of genes with 5 or more samples (case or control) with a number of reads > 20 (max):  17510
```

```
cat("\n")
```

Looking at the contingency tables, we have decided to go for *max()*, since the number of genes that meet the 5 replicates condition is greater (Also, see how many genes have respectively 0 and 50 samples that have a number of reads greater than 20). We proceed to remove genes with low signal, according to the 5 samples filtering.

```
filter_counts_df <- clean_coding_raw[filter_vec_max>=repl_thr,]
cat ("\nDimension: ", dim(filter_counts_df))
```

```
##
## Dimension:  17510 100
```

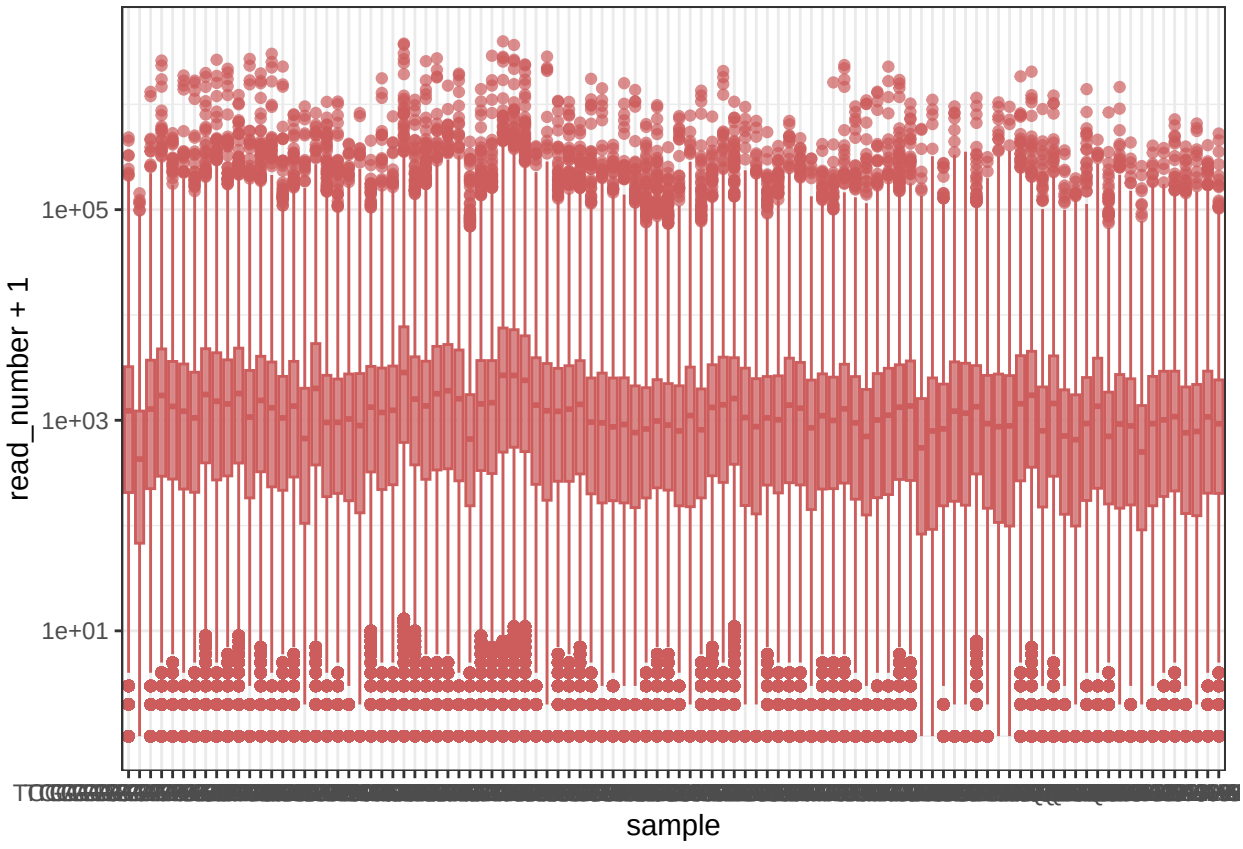
The dimension of the filtered dataset got even smaller, with 17510 genes as rows, so 4645 less. We have filtered the *r_anno_df* dataset as well to update the annotation corresponding to the filtered genes.

```
filter_anno_df <- r_anno_df[rownames(filter_counts_df),]
dim(filter_anno_df)
```

```
## [1] 17510     3
```

In order to understand the distribution of the numbers of reads (each related to a gene ID) per sample, we have produced a box and whiskers plot, using the *geom_boxplot()* function, considering the numbers of reads as continuous variables, that we had to scale in log10 + 1 to avoid infinite values when the number of reads are zero for any gene ID of the considered replicate/sample.

```
long_counts_df <- gather(filter_counts_df, key = "sample", value = "read_number")
ggplot(data=long_counts_df,aes(sample,read_number+1)) +
   geom_boxplot(colour="indianred",fill="indianred",alpha=0.7) +
   theme_bw() +
   scale_y_log10()
```



Under the assumption of independent and identically distributed variables, and being the number of variables (gene ID) large enough and the distribution not wild, we can assume that the central limit theorem holds and normalization of the number of reads per sample can take place.

First we need to create a DGEList, *edge_c*, which stands for Digital Gene Expression List, having as parameters *counts*, *samples*, *gene* and *group*, all the necessary information to proceed with the number of reads per sample normalization.

```
edge_c <- DGEList(counts=filter_counts_df,group=c_anno_df$condition,
                  samples=c_anno_df,genes=filter_anno_df)
head(edge_c,2)
```

```
## An object of class "DGEList"
## $counts
##                 TCGA-85-A4QQ-01A TCGA-94-7557-01A TCGA-98-A53A-01A
## ENSG00000000003             1733             2484             2591
## ENSG00000000419             2462             3162             1816
##                 TCGA-NC-A5HR-01A TCGA-56-7730-11A TCGA-33-6737-11A
## ENSG00000000003             2017             2084             3000
## ENSG00000000419             2311             1170             1093
##                 TCGA-22-5491-11A TCGA-66-2744-01A TCGA-22-5482-11A
## ENSG00000000003             1699             3931             2795
## ENSG00000000419             1517             2541             1830
##                 TCGA-77-7338-11A TCGA-56-A4BW-01A TCGA-43-3394-11A
## ENSG00000000003             1576             1724             4735
## ENSG00000000419             1800             1901             3555
##                 TCGA-56-8083-11A TCGA-66-2768-01A TCGA-NC-A5HP-01A
## ENSG00000000003             1011             4196             1402
## ENSG00000000419              984             2445             1760
##                 TCGA-39-5040-11A TCGA-77-8007-11A TCGA-56-8201-11A
## ENSG00000000003             1403             1733              780
## ENSG00000000419             3070             2473              884
##                 TCGA-56-7731-11A TCGA-56-7582-11A TCGA-43-3394-01A
## ENSG00000000003             1304             2248             3644
## ENSG00000000419             1468             1606             2503
##                 TCGA-34-7107-11A TCGA-21-5786-01A TCGA-77-8148-01A
## ENSG00000000003              965             5451             2507
## ENSG00000000419              993             2788             1644
##                 TCGA-77-8144-01A TCGA-90-6837-11A TCGA-33-4587-11A
## ENSG00000000003             1459             2024             1683
## ENSG00000000419             2769             1203             2193
##                 TCGA-56-8628-01A TCGA-22-5478-11A TCGA-63-A5MR-01A
## ENSG00000000003             3298             1271             1025
## ENSG00000000419             1070             1491             1830
##                 TCGA-98-8022-01A TCGA-LA-A7SW-01A TCGA-56-8623-11A
## ENSG00000000003              965             3397             1560
## ENSG00000000419             1106             3091             1155
##                 TCGA-56-8082-11A TCGA-22-5481-11A TCGA-43-6771-11A
## ENSG00000000003             1121             2006             1208
## ENSG00000000419             1474             1480             2041
##                 TCGA-34-8454-11A TCGA-L3-A524-01A TCGA-34-5239-01A
## ENSG00000000003              941             2872             2388
## ENSG00000000419             1064             3673             3608
##                 TCGA-98-A539-01A TCGA-85-7710-11A TCGA-43-7657-11A
```

```
## ENSG00000000003             1207             1293             2443
## ENSG00000000419             2165             1872             1681
##                 TCGA-92-7340-11A TCGA-33-AASL-01A TCGA-77-7337-11A
## ENSG00000000003             2334             1233             1295
## ENSG00000000419             1966             1744             1437
##                 TCGA-56-7580-11A TCGA-43-6773-11A TCGA-60-2697-01A
## ENSG00000000003             1184             1106             2575
## ENSG00000000419             1411              653             2246
##                 TCGA-92-8063-01A TCGA-43-7658-11A TCGA-58-8386-11A
## ENSG00000000003             1709             2971             1270
## ENSG00000000419             1053             1881             1113
##                 TCGA-56-A4BX-01A TCGA-79-5596-01A TCGA-51-4080-11A
## ENSG00000000003             4720             4603             2385
## ENSG00000000419             2327             1284             2405
##                 TCGA-37-4133-01A TCGA-56-7579-11A TCGA-39-5031-01A
## ENSG00000000003             8805             2144             3791
## ENSG00000000419             3799             1768             1805
##                 TCGA-85-7710-01A TCGA-77-8008-11A TCGA-60-2723-01A
## ENSG00000000003             1749              995             5062
## ENSG00000000419             3187             1657             1577
##                 TCGA-43-5670-11A TCGA-60-2715-01A TCGA-56-8309-11A
## ENSG00000000003             4640             2345             2470
## ENSG00000000419             1940             1748              941
##                 TCGA-51-6867-01A TCGA-NC-A5HE-01A TCGA-18-4721-01A
## ENSG00000000003             6125             1840             1401
## ENSG00000000419             4183             2646             1161
##                 TCGA-22-4609-11A TCGA-51-4079-11A TCGA-22-5483-11A
## ENSG00000000003             1377             3366             1277
## ENSG00000000419             1594             2637             1331
##                 TCGA-68-A59I-01A TCGA-92-8065-01A TCGA-22-5471-11A
## ENSG00000000003             3850             2725             2145
## ENSG00000000419             1879             2099             1173
##                 TCGA-22-4593-11A TCGA-37-4132-01A TCGA-J1-A4AH-01A
## ENSG00000000003             1585             7436             1982
## ENSG00000000419             2325             2664             1151
##                 TCGA-90-A59Q-01A TCGA-63-A5MV-01A TCGA-22-5489-11A
## ENSG00000000003             1234             3170             3213
## ENSG00000000419             1272             2322             1581
##                 TCGA-77-7142-11A TCGA-77-7138-11A TCGA-66-2781-01A
## ENSG00000000003             1037             1400             4203
## ENSG00000000419             1375             1693             5374
##                 TCGA-60-2709-11A TCGA-22-5472-11A TCGA-77-8140-01A
## ENSG00000000003             2988             2397             4569
## ENSG00000000419             3586             1761             1595
##                 TCGA-18-3410-01A TCGA-O2-A52V-01A TCGA-51-4081-11A
## ENSG00000000003             2209             2794             2582
## ENSG00000000419             2033             1659             2209
##                 TCGA-90-7767-11A TCGA-43-6770-01A TCGA-66-2800-01A
## ENSG00000000003             3286             8317             3827
## ENSG00000000419             2272             4023             3010
##                 TCGA-92-8064-01A TCGA-85-A4QR-01A TCGA-56-7222-11A
## ENSG00000000003              963             2508             1424
## ENSG00000000419              880             1656             1893
##                 TCGA-XC-AA0X-01A TCGA-77-7335-11A TCGA-22-4609-01A
```

```
## ENSG00000000003             2175             1765             1811
## ENSG00000000419             1646             1507             2058
##                 TCGA-43-6143-11A TCGA-43-6647-11A TCGA-66-2793-01A
## ENSG00000000003             1486             1685             2039
## ENSG00000000419             1404             1846             2352
##                 TCGA-85-8351-01A
## ENSG00000000003             3012
## ENSG00000000419             2872
##
## $samples
##                     group lib.size norm.factors           sample condition
## TCGA-85-A4QQ-01A     case 58571703            1 TCGA-85-A4QQ-01A      case
## TCGA-94-7557-01A     case 70493279            1 TCGA-94-7557-01A      case
## TCGA-98-A53A-01A     case 44094066            1 TCGA-98-A53A-01A      case
## TCGA-NC-A5HR-01A     case 41067788            1 TCGA-NC-A5HR-01A      case
## TCGA-56-7730-11A control 55292582            1 TCGA-56-7730-11A   control
## 95 more rows ...
##
## $genes
##                 ensembl_gene_id external_gene_name length
## ENSG00000000003 ENSG00000000003             TSPAN6  12883
## ENSG00000000419 ENSG00000000419               DPM1  24273
```

We recall that TMM doesn't normalize the reads, but instead calculates normalization factors, since there can be composition biases, where certain genes have much higher read counts due to technical reasons, and we do not want to use them when calculating the library size.

Instead of using the total library size (the sum of the reads for all genes), TMM trims off the most highly variable genes (Trimmed mean of M-values, where M-values are the log fold change between each sample and a reference) and then calculates a normalization factor that is used to adjust the library size when we compute logCPM values with the function *cpm()*. We have used the *geom_boxplot()* function again to visually check if the bar plots of the normalized distribution of the numbers of reads were more aligned, as expected.

```
edge_n <- calcNormFactors(edge_c,method="TMM")
head(edge_n,2)
```

```
## An object of class "DGEList"
## $counts
##                 TCGA-85-A4QQ-01A TCGA-94-7557-01A TCGA-98-A53A-01A
## ENSG00000000003             1733             2484             2591
## ENSG00000000419             2462             3162             1816
##                 TCGA-NC-A5HR-01A TCGA-56-7730-11A TCGA-33-6737-11A
## ENSG00000000003             2017             2084             3000
## ENSG00000000419             2311             1170             1093
##                 TCGA-22-5491-11A TCGA-66-2744-01A TCGA-22-5482-11A
## ENSG00000000003             1699             3931             2795
## ENSG00000000419             1517             2541             1830
##                 TCGA-77-7338-11A TCGA-56-A4BW-01A TCGA-43-3394-11A
## ENSG00000000003             1576             1724             4735
## ENSG00000000419             1800             1901             3555
##                 TCGA-56-8083-11A TCGA-66-2768-01A TCGA-NC-A5HP-01A
## ENSG00000000003             1011             4196             1402
## ENSG00000000419              984             2445             1760
```

```
##                 TCGA-39-5040-11A TCGA-77-8007-11A TCGA-56-8201-11A
## ENSG00000000003             1403             1733              780
## ENSG00000000419             3070             2473              884
##                 TCGA-56-7731-11A TCGA-56-7582-11A TCGA-43-3394-01A
## ENSG00000000003             1304             2248             3644
## ENSG00000000419             1468             1606             2503
##                 TCGA-34-7107-11A TCGA-21-5786-01A TCGA-77-8148-01A
## ENSG00000000003              965             5451             2507
## ENSG00000000419              993             2788             1644
##                 TCGA-77-8144-01A TCGA-90-6837-11A TCGA-33-4587-11A
## ENSG00000000003             1459             2024             1683
## ENSG00000000419             2769             1203             2193
##                 TCGA-56-8628-01A TCGA-22-5478-11A TCGA-63-A5MR-01A
## ENSG00000000003             3298             1271             1025
## ENSG00000000419             1070             1491             1830
##                 TCGA-98-8022-01A TCGA-LA-A7SW-01A TCGA-56-8623-11A
## ENSG00000000003              965             3397             1560
## ENSG00000000419             1106             3091             1155
##                 TCGA-56-8082-11A TCGA-22-5481-11A TCGA-43-6771-11A
## ENSG00000000003             1121             2006             1208
## ENSG00000000419             1474             1480             2041
##                 TCGA-34-8454-11A TCGA-L3-A524-01A TCGA-34-5239-01A
## ENSG00000000003              941             2872             2388
## ENSG00000000419             1064             3673             3608
##                 TCGA-98-A539-01A TCGA-85-7710-11A TCGA-43-7657-11A
## ENSG00000000003             1207             1293             2443
## ENSG00000000419             2165             1872             1681
##                 TCGA-92-7340-11A TCGA-33-AASL-01A TCGA-77-7337-11A
## ENSG00000000003             2334             1233             1295
## ENSG00000000419             1966             1744             1437
##                 TCGA-56-7580-11A TCGA-43-6773-11A TCGA-60-2697-01A
## ENSG00000000003             1184             1106             2575
## ENSG00000000419             1411              653             2246
##                 TCGA-92-8063-01A TCGA-43-7658-11A TCGA-58-8386-11A
## ENSG00000000003             1709             2971             1270
## ENSG00000000419             1053             1881             1113
##                 TCGA-56-A4BX-01A TCGA-79-5596-01A TCGA-51-4080-11A
## ENSG00000000003             4720             4603             2385
## ENSG00000000419             2327             1284             2405
##                 TCGA-37-4133-01A TCGA-56-7579-11A TCGA-39-5031-01A
## ENSG00000000003             8805             2144             3791
## ENSG00000000419             3799             1768             1805
##                 TCGA-85-7710-01A TCGA-77-8008-11A TCGA-60-2723-01A
## ENSG00000000003             1749              995             5062
## ENSG00000000419             3187             1657             1577
##                 TCGA-43-5670-11A TCGA-60-2715-01A TCGA-56-8309-11A
## ENSG00000000003             4640             2345             2470
## ENSG00000000419             1940             1748              941
##                 TCGA-51-6867-01A TCGA-NC-A5HE-01A TCGA-18-4721-01A
## ENSG00000000003             6125             1840             1401
## ENSG00000000419             4183             2646             1161
##                 TCGA-22-4609-11A TCGA-51-4079-11A TCGA-22-5483-11A
## ENSG00000000003             1377             3366             1277
## ENSG00000000419             1594             2637             1331
```

```
##                 TCGA-68-A59I-01A TCGA-92-8065-01A TCGA-22-5471-11A
## ENSG00000000003             3850             2725             2145
## ENSG00000000419             1879             2099             1173
##                 TCGA-22-4593-11A TCGA-37-4132-01A TCGA-J1-A4AH-01A
## ENSG00000000003             1585             7436             1982
## ENSG00000000419             2325             2664             1151
##                 TCGA-90-A59Q-01A TCGA-63-A5MV-01A TCGA-22-5489-11A
## ENSG00000000003             1234             3170             3213
## ENSG00000000419             1272             2322             1581
##                 TCGA-77-7142-11A TCGA-77-7138-11A TCGA-66-2781-01A
## ENSG00000000003             1037             1400             4203
## ENSG00000000419             1375             1693             5374
##                 TCGA-60-2709-11A TCGA-22-5472-11A TCGA-77-8140-01A
## ENSG00000000003             2988             2397             4569
## ENSG00000000419             3586             1761             1595
##                 TCGA-18-3410-01A TCGA-O2-A52V-01A TCGA-51-4081-11A
## ENSG00000000003             2209             2794             2582
## ENSG00000000419             2033             1659             2209
##                 TCGA-90-7767-11A TCGA-43-6770-01A TCGA-66-2800-01A
## ENSG00000000003             3286             8317             3827
## ENSG00000000419             2272             4023             3010
##                 TCGA-92-8064-01A TCGA-85-A4QR-01A TCGA-56-7222-11A
## ENSG00000000003              963             2508             1424
## ENSG00000000419              880             1656             1893
##                 TCGA-XC-AA0X-01A TCGA-77-7335-11A TCGA-22-4609-01A
## ENSG00000000003             2175             1765             1811
## ENSG00000000419             1646             1507             2058
##                 TCGA-43-6143-11A TCGA-43-6647-11A TCGA-66-2793-01A
## ENSG00000000003             1486             1685             2039
## ENSG00000000419             1404             1846             2352
##                 TCGA-85-8351-01A
## ENSG00000000003             3012
## ENSG00000000419             2872
##
## $samples
##                    group lib.size norm.factors           sample condition
## TCGA-85-A4QQ-01A    case 58571703    0.8869267 TCGA-85-A4QQ-01A      case
## TCGA-94-7557-01A    case 70493279    1.0680208 TCGA-94-7557-01A      case
## TCGA-98-A53A-01A    case 44094066    1.1527778 TCGA-98-A53A-01A      case
## TCGA-NC-A5HR-01A    case 41067788    1.0903034 TCGA-NC-A5HR-01A      case
## TCGA-56-7730-11A control 55292582    0.9336756 TCGA-56-7730-11A   control
## 95 more rows ...
##
## $genes
##                 ensembl_gene_id external_gene_name length
## ENSG00000000003 ENSG00000000003             TSPAN6  12883
## ENSG00000000419 ENSG00000000419               DPM1  24273
```

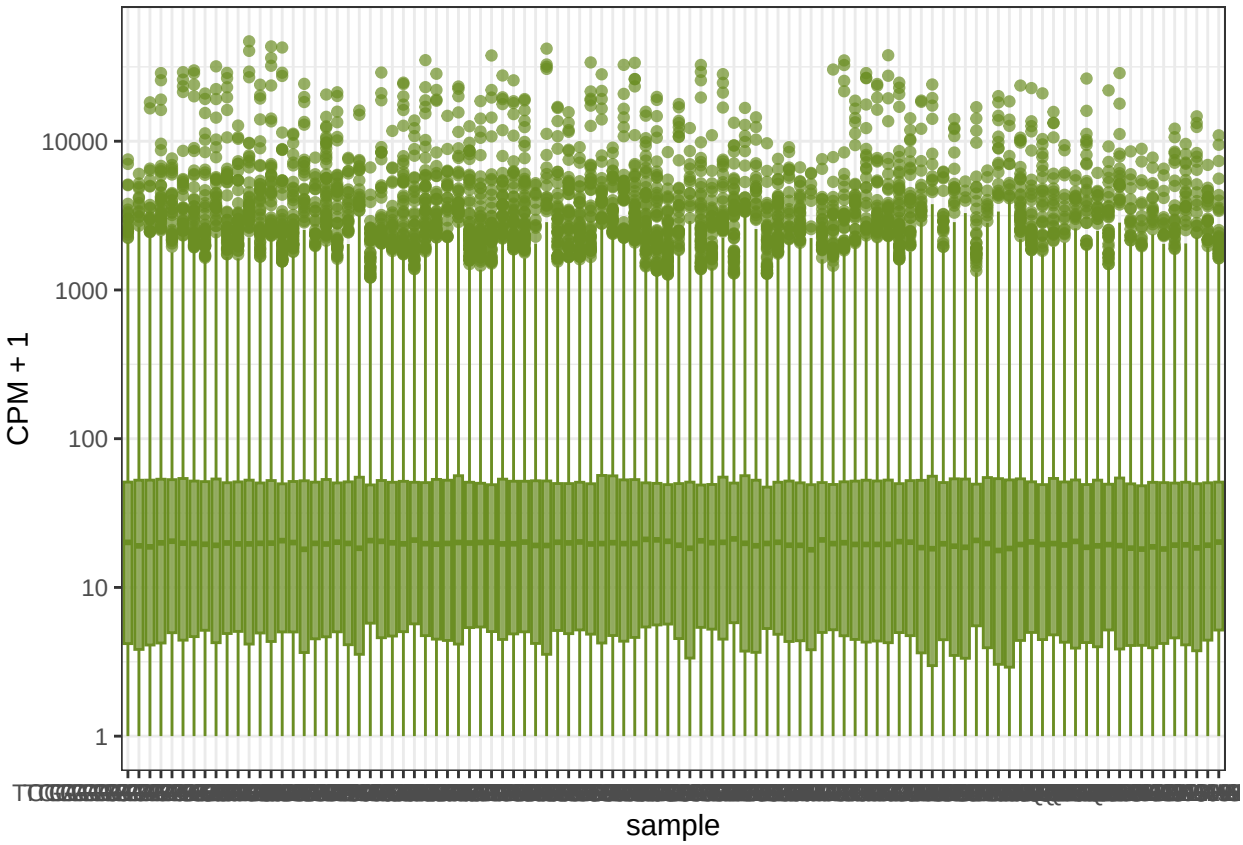
```
cpm_table <- as.data.frame(round(cpm(edge_n),2))
head(cpm_table,2)
```

```
##                 TCGA-85-A4QQ-01A TCGA-94-7557-01A TCGA-98-A53A-01A
## ENSG00000000003            33.36            32.99            50.97
## ENSG00000000419            47.39            42.00            35.73
```

```
##                 TCGA-NC-A5HR-01A TCGA-56-7730-11A TCGA-33-6737-11A
## ENSG00000000003            45.05            40.37            41.88
## ENSG00000000419            51.61            22.66            15.26
##                 TCGA-22-5491-11A TCGA-66-2744-01A TCGA-22-5482-11A
## ENSG00000000003            24.41            74.36            29.02
## ENSG00000000419            21.79            48.07            19.00
##                 TCGA-77-7338-11A TCGA-56-A4BW-01A TCGA-43-3394-11A
## ENSG00000000003            26.14            39.72            31.10
## ENSG00000000419            29.86            43.80            23.35
##                 TCGA-56-8083-11A TCGA-66-2768-01A TCGA-NC-A5HP-01A
## ENSG00000000003            20.76            54.95            33.83
## ENSG00000000419            20.21            32.02            42.47
##                 TCGA-39-5040-11A TCGA-77-8007-11A TCGA-56-8201-11A
## ENSG00000000003            22.99            25.18            19.17
## ENSG00000000419            50.32            35.93            21.72
##                 TCGA-56-7731-11A TCGA-56-7582-11A TCGA-43-3394-01A
## ENSG00000000003            25.86            30.52            55.63
## ENSG00000000419            29.11            21.81            38.21
##                 TCGA-34-7107-11A TCGA-21-5786-01A TCGA-77-8148-01A
## ENSG00000000003            18.84            75.73            56.56
## ENSG00000000419            19.39            38.74            37.09
##                 TCGA-77-8144-01A TCGA-90-6837-11A TCGA-33-4587-11A
## ENSG00000000003            31.78            25.99            31.43
## ENSG00000000419            60.32            15.45            40.96
##                 TCGA-56-8628-01A TCGA-22-5478-11A TCGA-63-A5MR-01A
## ENSG00000000003            71.25            15.28            21.31
## ENSG00000000419            23.12            17.93            38.04
##                 TCGA-98-8022-01A TCGA-LA-A7SW-01A TCGA-56-8623-11A
## ENSG00000000003            25.29            57.79            31.55
## ENSG00000000419            28.99            52.58            23.36
##                 TCGA-56-8082-11A TCGA-22-5481-11A TCGA-43-6771-11A
## ENSG00000000003            24.60            26.55            14.39
## ENSG00000000419            32.34            19.59            24.31
##                 TCGA-34-8454-11A TCGA-L3-A524-01A TCGA-34-5239-01A
## ENSG00000000003            18.88            54.95            22.37
## ENSG00000000419            21.35            70.27            33.80
##                 TCGA-98-A539-01A TCGA-85-7710-11A TCGA-43-7657-11A
## ENSG00000000003            23.72            18.97            32.56
## ENSG00000000419            42.55            27.47            22.40
##                 TCGA-92-7340-11A TCGA-33-AASL-01A TCGA-77-7337-11A
## ENSG00000000003            30.31            31.36            23.81
## ENSG00000000419            25.53            44.36            26.42
##                 TCGA-56-7580-11A TCGA-43-6773-11A TCGA-60-2697-01A
## ENSG00000000003            17.58            31.56            36.91
## ENSG00000000419            20.95            18.63            32.20
##                 TCGA-92-8063-01A TCGA-43-7658-11A TCGA-58-8386-11A
## ENSG00000000003            44.09            38.72            30.64
## ENSG00000000419            27.16            24.51            26.85
##                 TCGA-56-A4BX-01A TCGA-79-5596-01A TCGA-51-4080-11A
## ENSG00000000003            73.94            67.73            16.71
## ENSG00000000419            36.45            18.89            16.85
##                 TCGA-37-4133-01A TCGA-56-7579-11A TCGA-39-5031-01A
## ENSG00000000003           171.61            33.77            55.94
## ENSG00000000419            74.04            27.85            26.63
```

```
##                 TCGA-85-7710-01A TCGA-77-8008-11A TCGA-60-2723-01A
## ENSG00000000003            26.41            13.95            89.70
## ENSG00000000419            48.13            23.23            27.95
##                 TCGA-43-5670-11A TCGA-60-2715-01A TCGA-56-8309-11A
## ENSG00000000003            57.88            29.47            60.04
## ENSG00000000419            24.20            21.97            22.87
##                 TCGA-51-6867-01A TCGA-NC-A5HE-01A TCGA-18-4721-01A
## ENSG00000000003            79.89            31.03            59.15
## ENSG00000000419            54.56            44.62            49.01
##                 TCGA-22-4609-11A TCGA-51-4079-11A TCGA-22-5483-11A
## ENSG00000000003            21.32            23.48            22.19
## ENSG00000000419            24.69            18.39            23.13
##                 TCGA-68-A59I-01A TCGA-92-8065-01A TCGA-22-5471-11A
## ENSG00000000003            72.35            51.59            38.23
## ENSG00000000419            35.31            39.74            20.90
##                 TCGA-22-4593-11A TCGA-37-4132-01A TCGA-J1-A4AH-01A
## ENSG00000000003            17.47           135.89            67.93
## ENSG00000000419            25.62            48.68            39.45
##                 TCGA-90-A59Q-01A TCGA-63-A5MV-01A TCGA-22-5489-11A
## ENSG00000000003            28.69            56.38            39.18
## ENSG00000000419            29.58            41.30            19.28
##                 TCGA-77-7142-11A TCGA-77-7138-11A TCGA-66-2781-01A
## ENSG00000000003            20.30            20.70            58.76
## ENSG00000000419            26.92            25.03            75.13
##                 TCGA-60-2709-11A TCGA-22-5472-11A TCGA-77-8140-01A
## ENSG00000000003            40.71            25.29           146.62
## ENSG00000000419            48.86            18.58            51.18
##                 TCGA-18-3410-01A TCGA-O2-A52V-01A TCGA-51-4081-11A
## ENSG00000000003            34.33            47.29            20.69
## ENSG00000000419            31.59            28.08            17.70
##                 TCGA-90-7767-11A TCGA-43-6770-01A TCGA-66-2800-01A
## ENSG00000000003             36.6            81.87            69.00
## ENSG00000000419             25.3            39.60            54.27
##                 TCGA-92-8064-01A TCGA-85-A4QR-01A TCGA-56-7222-11A
## ENSG00000000003            28.59            48.98            21.00
## ENSG00000000419            26.13            32.34            27.92
##                 TCGA-XC-AA0X-01A TCGA-77-7335-11A TCGA-22-4609-01A
## ENSG00000000003            45.10            46.46            26.05
## ENSG00000000419            34.13            39.67            29.60
##                 TCGA-43-6143-11A TCGA-43-6647-11A TCGA-66-2793-01A
## ENSG00000000003            20.30            17.50            40.71
## ENSG00000000419            19.18            19.18            46.96
##                 TCGA-85-8351-01A
## ENSG00000000003            60.76
## ENSG00000000419            57.94
```

```
long_cpm_df <- gather(cpm_table, key = "sample", value = "CPM")

ggplot(data=long_cpm_df,aes(sample,CPM+1)) +
   geom_boxplot(colour="olivedrab",fill="olivedrab",alpha=0.7)+
   theme_bw()+
   scale_y_log10()
```



In order to verify that the normalization process worked, we have checked the values inside the CPM table, following instructions from https://combine-australia.github.io/RNAseq-R/06-rnaseq-day1.html (*Filtering lowly expressed genes* section).

```
# Which values in cpm_table are greater than 0.25?
# (As a general rule, a good threshold can be chosen by identifying the CPM
# that corresponds to a count of 10)
thresh <- cpm_table > 0.25
# This produces a logical matrix with TRUEs and FALSEs
head(thresh,2)
```

```
##                 TCGA-85-A4QQ-01A TCGA-94-7557-01A TCGA-98-A53A-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-NC-A5HR-01A TCGA-56-7730-11A TCGA-33-6737-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-22-5491-11A TCGA-66-2744-01A TCGA-22-5482-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-77-7338-11A TCGA-56-A4BW-01A TCGA-43-3394-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-56-8083-11A TCGA-66-2768-01A TCGA-NC-A5HP-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
```

```
##                 TCGA-39-5040-11A TCGA-77-8007-11A TCGA-56-8201-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-56-7731-11A TCGA-56-7582-11A TCGA-43-3394-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-34-7107-11A TCGA-21-5786-01A TCGA-77-8148-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-77-8144-01A TCGA-90-6837-11A TCGA-33-4587-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-56-8628-01A TCGA-22-5478-11A TCGA-63-A5MR-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-98-8022-01A TCGA-LA-A7SW-01A TCGA-56-8623-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-56-8082-11A TCGA-22-5481-11A TCGA-43-6771-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-34-8454-11A TCGA-L3-A524-01A TCGA-34-5239-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-98-A539-01A TCGA-85-7710-11A TCGA-43-7657-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-92-7340-11A TCGA-33-AASL-01A TCGA-77-7337-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-56-7580-11A TCGA-43-6773-11A TCGA-60-2697-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-92-8063-01A TCGA-43-7658-11A TCGA-58-8386-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-56-A4BX-01A TCGA-79-5596-01A TCGA-51-4080-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-37-4133-01A TCGA-56-7579-11A TCGA-39-5031-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-85-7710-01A TCGA-77-8008-11A TCGA-60-2723-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-43-5670-11A TCGA-60-2715-01A TCGA-56-8309-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-51-6867-01A TCGA-NC-A5HE-01A TCGA-18-4721-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-22-4609-11A TCGA-51-4079-11A TCGA-22-5483-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
```

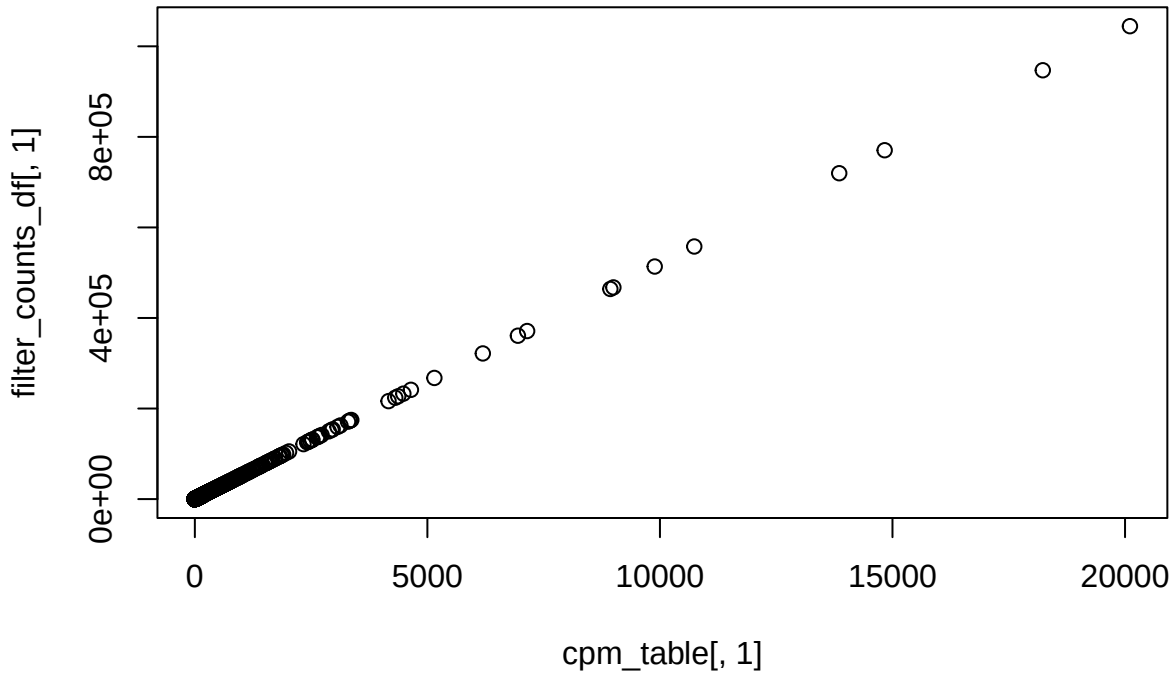
```
##                 TCGA-68-A59I-01A TCGA-92-8065-01A TCGA-22-5471-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-22-4593-11A TCGA-37-4132-01A TCGA-J1-A4AH-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-90-A59Q-01A TCGA-63-A5MV-01A TCGA-22-5489-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-77-7142-11A TCGA-77-7138-11A TCGA-66-2781-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-60-2709-11A TCGA-22-5472-11A TCGA-77-8140-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-18-3410-01A TCGA-O2-A52V-01A TCGA-51-4081-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-90-7767-11A TCGA-43-6770-01A TCGA-66-2800-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-92-8064-01A TCGA-85-A4QR-01A TCGA-56-7222-11A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-XC-AA0X-01A TCGA-77-7335-11A TCGA-22-4609-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-43-6143-11A TCGA-43-6647-11A TCGA-66-2793-01A
## ENSG00000000003             TRUE             TRUE             TRUE
## ENSG00000000419             TRUE             TRUE             TRUE
##                 TCGA-85-8351-01A
## ENSG00000000003             TRUE
## ENSG00000000419             TRUE
```

```
# Summary of how many TRUEs there are in each row
# There are 13036 genes that have TRUEs in all 100 samples.
table(rowSums(thresh))
```

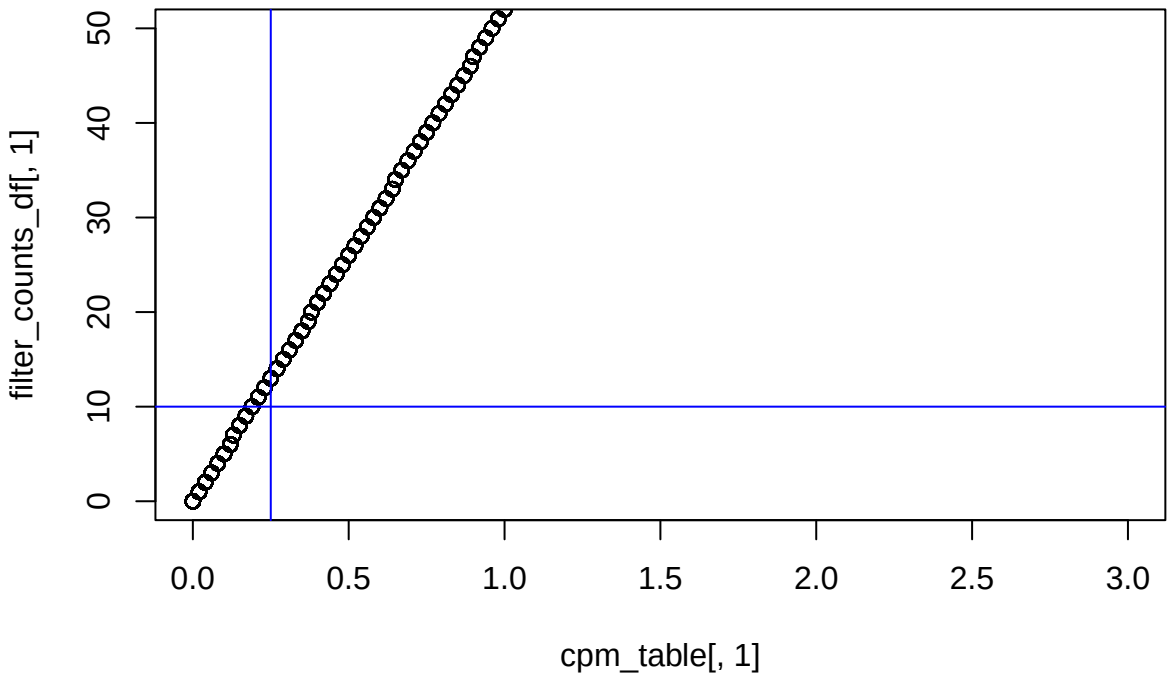
```
##
##     4     5     6     7     8     9    10    11    12    13    14    15    16
##     2    13    21    25    39    41    36    38    48    37    44    48    29
##    17    18    19    20    21    22    23    24    25    26    27    28    29
##    29    29    26    35    40    30    38    39    31    28    25    31    30
##    30    31    32    33    34    35    36    37    38    39    40    41    42
##    31    24    33    28    28    25    31    39    19    32    33    25    26
##    43    44    45    46    47    48    49    50    51    52    53    54    55
##    21    37    37    32    29    24    25    35    25    29    18    31    30
##    56    57    58    59    60    61    62    63    64    65    66    67    68
##    31    25    28    32    29    35    33    26    25    26    31    36    32
##    69    70    71    72    73    74    75    76    77    78    79    80    81
##    38    37    38    36    44    35    35    38    44    35    44    34    42
##    82    83    84    85    86    87    88    89    90    91    92    93    94
##    41    41    39    44    53    47    69    72    72    75    86    90    97
##    95    96    97    98    99   100
```

```
##    113    130    176    270    491 13036
```

```
# Let's have a look and see whether our threshold of 0.25 does indeed correspond
# to a count of about 10-15
# We will look at the first sample
plot(cpm_table[,1],filter_counts_df[,1])
```

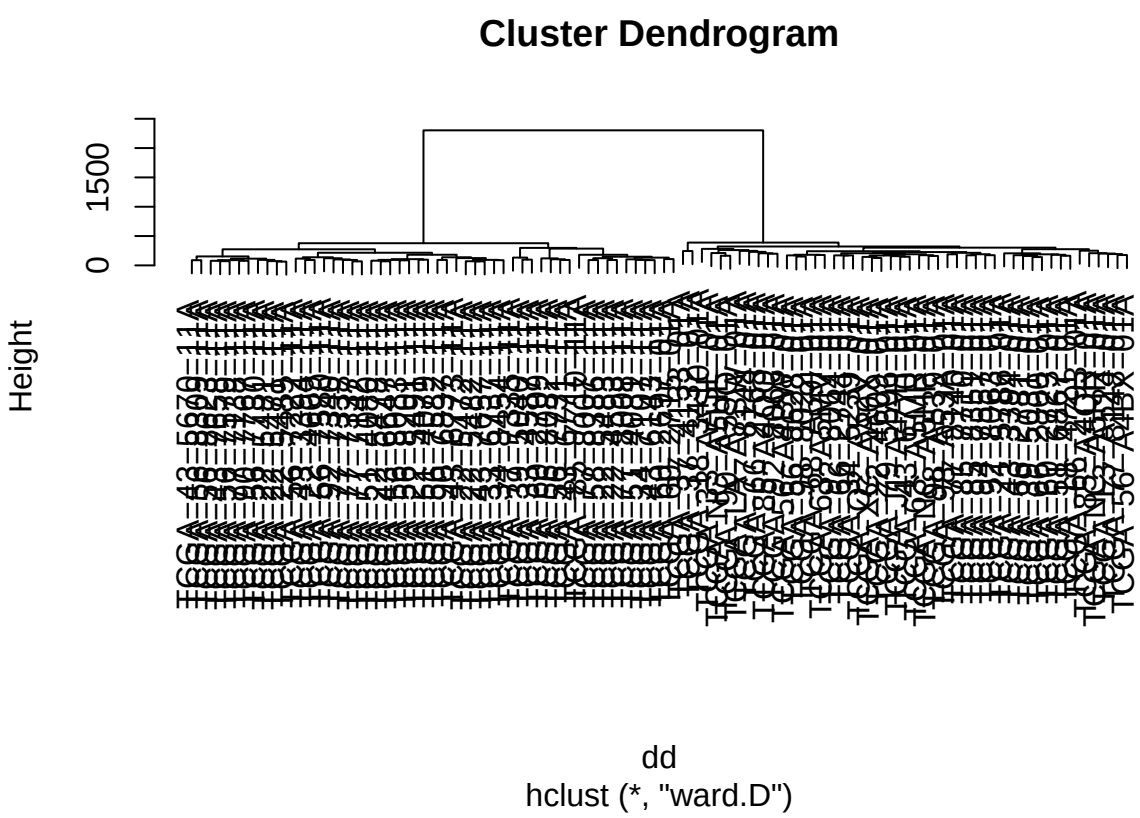



```
# Let us limit the x and y-axis so we can actually look to see what is happening at the smaller counts
plot(cpm_table[,1],filter_counts_df[,1],ylim=c(0,50),xlim=c(0,3))
# Add a vertical line at 0.5 CPM
abline(v=0.25, col="blue")
abline(h=10, col="blue")
```



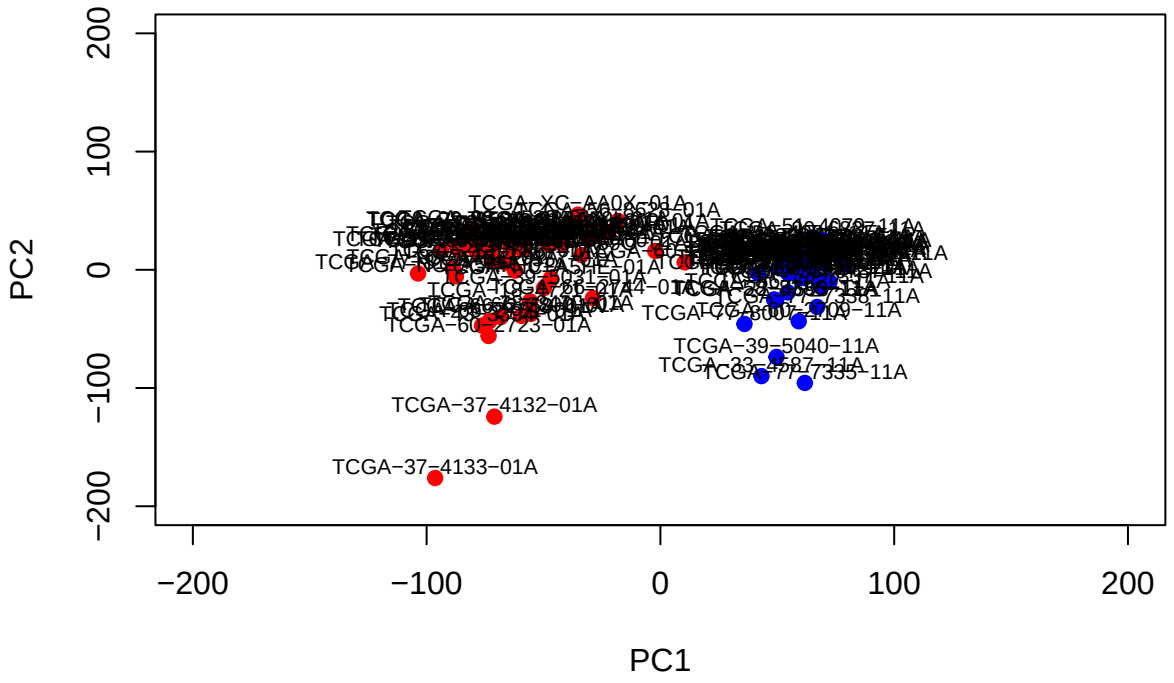
We have then applied unsupervised methods to gather preliminary information about the possibility of distinction between case and control samples, by applying hierarchical clustering with Ward's minimum variance criterion to the distance matrix, which results in the dendogram *hc*.

```
clu_data <- t(scale(t(cpm_table)))
dd <- dist(t(clu_data), method = "euclidean")
hc <- hclust(dd, method="ward.D")
plot(hc)
```

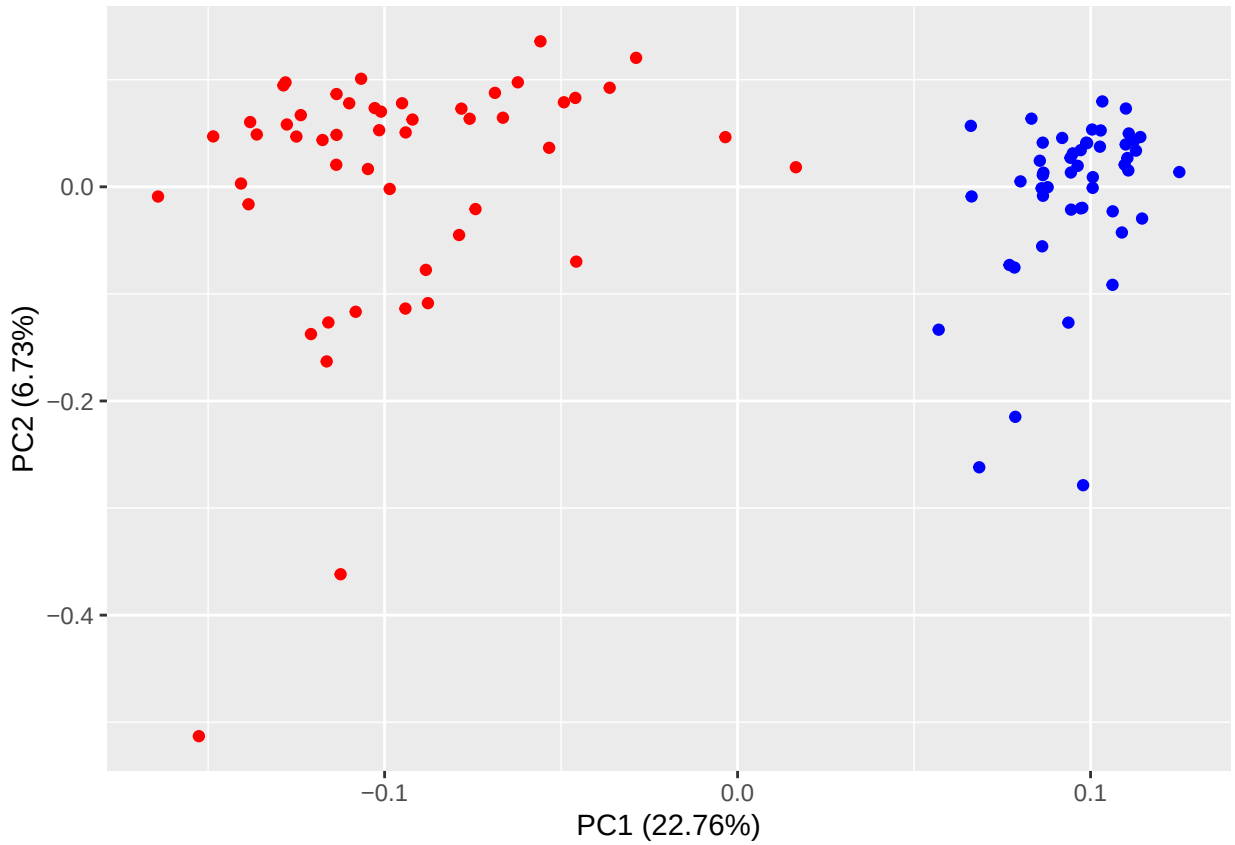


We have also performed Principal Component Analysis.

```
data.matrix <- cpm_table
color <- c(rep("red", ncol(cpm_table))) # case samples
color[grep('11A',colnames(data.matrix))]<-'blue' # control samples
data.PC <- prcomp(t(data.matrix),scale.=TRUE)
plot(data.PC$x[,1:2],xlim=c(-200,200),ylim=c(-200,200),col=color,pch=19)
text(data.PC$x[,1],data.PC$x[,2]+10,colnames(cpm_table),cex=0.7)
```




```
library(ggfortify)
pca_res <- prcomp(t(data.matrix), scale. = TRUE)
autoplot(pca_res, colour=color) #label=TRUE, label.size=3
```



We can clearly see that the two clusters have been identified and separated (with a slight overlap at their central border, then canceled by a zoomed in visualization).

Linear modeling and differential expression analysis in *edgeR* requires a design matrix to be specified. The design matrix records which treatment conditions were applied to each sample, and it also defines how the experimental effects are parametrized in the linear models. This design matrix simply links each group to the samples that belong to it. Each row of the design matrix corresponds to a sample whereas each column represents a coefficient corresponding to one of the 2 groups.

```
design <- model.matrix(~0+group, data=edge_n$samples)
colnames(design) <- levels(edge_n$samples$group)
rownames(design) <- edge_n$samples$sample
head(design,5)
```

```
##                  control case
## TCGA-85-A4QQ-01A       0    1
## TCGA-94-7557-01A       0    1
## TCGA-98-A53A-01A       0    1
## TCGA-NC-A5HR-01A       0    1
## TCGA-56-7730-11A       1    0
```

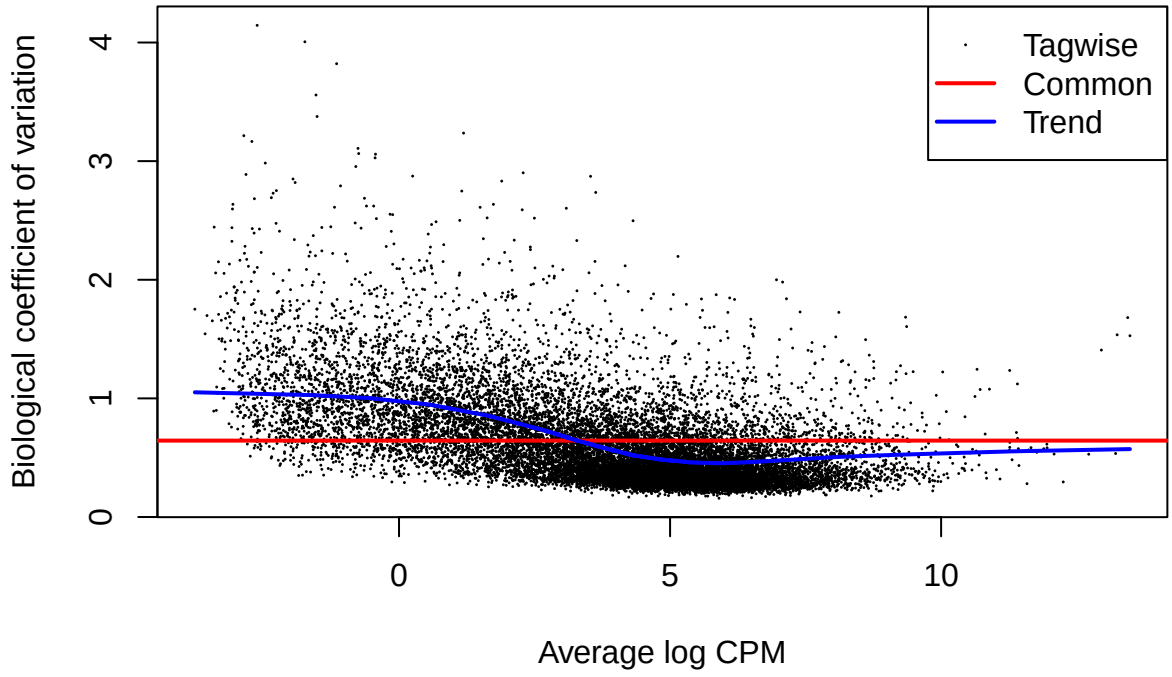
As explained by Chen et al. (2016, doi: 10.12688/f1000research.8987.2):

> edgeR uses the negative binomial (NB) distribution to model the read counts for each gene in each sample. The dispersion parameter of the NB distribution accounts for variability between biological replicates. The NB model can be extended with quasi-likelihood (QL) methods to account for gene-specific variability from both biological and technical sources.

Under the QL framework, the NB dispersion trend is used to describe the overall biological variability across all genes, and gene-specific variability above and below the overall level is picked up by the QL dispersion.

The estimation of QL dispersion is performed using the *glmQLFit()* function.

```
# calculate dispersion and fit with edgeR (necessary for differential expression analysis)
edge_d <- estimateDisp(edge_n,design)

# the vertical axis of the plot shows square-root dispersion (or BCV)
plotBCV(edge_d)
```



The NB dispersions tend to be higher for genes with very low counts. The dispersion trend (blue curve) tends to decrease smoothly with abundance and to asymptotic to a constant value (red line) for genes with larger counts. The asymptotic value for the BCV seems to be greater than 0.5, which is acceptable for human subjects.

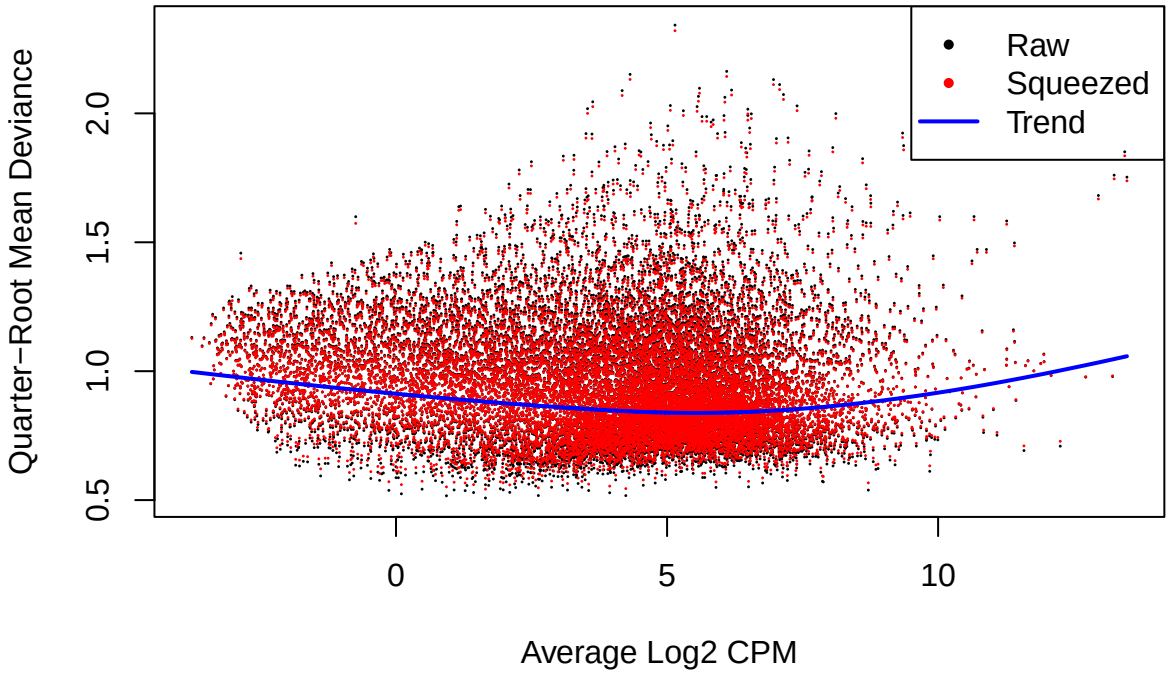
```
edge_f <- glmQLFit(edge_d,design) #robust=TRUE
head(edge_f$coefficients,5)
```

```
##                     control      case
## ENSG00000000003 -10.495627  -9.80463
## ENSG00000000419 -10.609647 -10.10007
## ENSG00000000457 -11.022128 -10.67398
## ENSG00000000460 -11.851276 -10.73640
## ENSG00000000938  -9.340072 -10.96196
```

```
summary(edge_f$df.prior)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.775   3.775   3.775   3.775   3.775   3.775
```

```
#visualization of QL dispersion
plotQLDisp(edge_f)
```



The plot of the quarter-root QL dispersion against the average abundance of each gene shows that there is not a difference between the raw (before EB moderation) and squeezed (after EB moderation) dispersions' estimates.

The next step is to test for differential expression between the experimental groups. The contrast corresponding to case and control samples are constructed using the *makeContrasts()* function:

```
contro <- makeContrasts("case-control", levels=design)
```

For example, a positive log2-fold-change (logFC) will indicate an up-regulated gene in case samples with respect to control ones, whereas a negative logFC will indicate a less expressed gene in carcinoma than normal cells. We will use QL F-tests as they give stricter error rate control by accounting for the uncertainty in dispersion estimation. The top DE genes can be viewed with *topTags()*.

```
# fit the model with generalized linear models
edge_t <- glmQLFTest(edge_f,contrast=contro)
# ordered by p-value
topTags(edge_t,5)
```

```
## Coefficient:  -1*control 1*case
##                  ensembl_gene_id external_gene_name length    logFC   logCPM
## ENSG00000117394 ENSG00000117394             SLC2A1  33540 5.572380 8.860388
## ENSG00000085999 ENSG00000085999             RAD54L  32019 3.528082 3.327960
## ENSG00000137807 ENSG00000137807              KIF23  34181 3.671980 4.438705
## ENSG00000112378 ENSG00000112378               PERP  18914 3.384865 9.381354
## ENSG00000110400 ENSG00000110400            NECTIN1 105792 4.613867 8.117906
##                        F       PValue          FDR
## ENSG00000117394 685.4443 5.191256e-47 5.626499e-43
## ENSG00000085999 682.1535 6.426613e-47 5.626499e-43
## ENSG00000137807 674.0412 1.091957e-46 6.373388e-43
## ENSG00000112378 664.1679 2.097384e-46 9.181300e-43
## ENSG00000110400 647.7528 6.326635e-46 2.215587e-42
```

```
is.de <- decideTestsDGE(edge_t)
summary(is.de)
```

```
##        -1*control 1*case
## Down                5859
## NotSig              3560
## Up                  8091
```

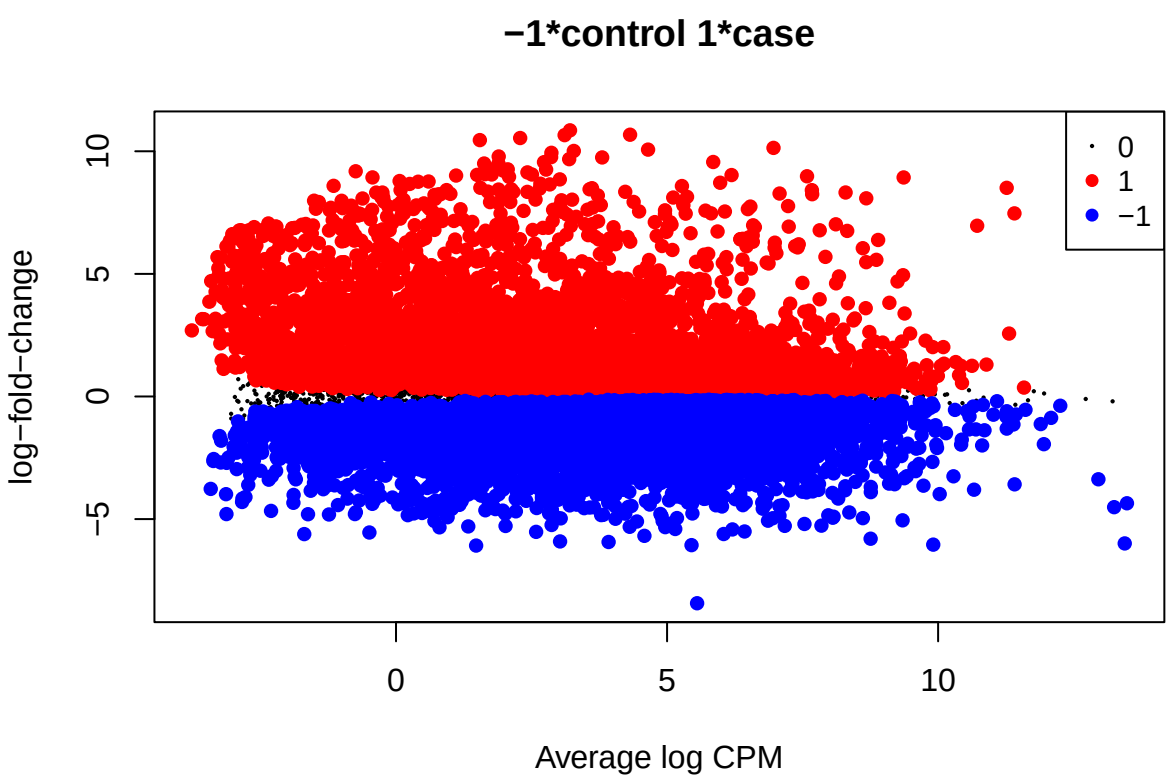
The top DE gene SLC2A1 has a large positive logFC, showing that it is far more highly expressed in the lung squamous carcinoma cells than normal. This gene is indeed known to be involved in a variety of cell death modalities and has been found to be associated with the prognosis and immune microenvironment of a variety of tumors. In particular, it is found to be highly expressed in LUSC (Wang et al., 2022, doi: 10.3389/fgene.2022.1068462).

```
degs_example <- as.data.frame(topTags(edge_t,n=20, p.value = 0.01,sort.by = "logFC"))
head(degs_example,5)
```

```
##                  ensembl_gene_id external_gene_name length    logFC   logCPM
## ENSG00000267978 ENSG00000267978            MAGEA9B   5806 10.85332 3.209478
## ENSG00000104371 ENSG00000104371               DKK4   3166 10.67517 4.317621
## ENSG00000123584 ENSG00000123584             MAGEA9   5807 10.65832 3.110258
## ENSG00000177243 ENSG00000177243           DEFB103B   1460 10.54035 2.290118
## ENSG00000006059 ENSG00000006059             KRT33A   4736 10.45522 1.546511
##                         F       PValue          FDR
## ENSG00000267978 235.35897 3.188515e-28 7.620210e-27
## ENSG00000104371  96.12085 2.256596e-16 1.114926e-15
## ENSG00000123584 229.85905 7.389562e-28 1.663126e-26
## ENSG00000177243 108.66022 9.613693e-18 5.566659e-17
## ENSG00000006059 148.75315 1.257354e-21 1.170550e-20
```

If we look at the results, sorting by logFC, the top DE gene is MAGEA9B, which is know to be expressed in many tumors of several types, including melanoma, head and neck squamous cell carcinoma, lung carcinoma and breast carcinoma,but not in normal tissues (genecards.org).

```
plotMD(edge_t, status=is.de, values=c(1,-1), col=c("red","blue"),legend="topright")
```



The MD plot shows the log-fold change and average abundance of each gene. Significantly up and down DE genes are highlighted in red and blue, respectively.

Let us know apply the proposed filtering on logCPM, logFC and FDR. We have set a p-value cutoff of 0.01, a log fold change ratio > 1.5 for up-regulated genes and < -1.5 for down-regulated genes and a logCPM >1.

```
# keep all the DE genes with a pvalue lower than 0.01
DEGs_clean <- as.data.frame(topTags(edge_t, n=nrow(edge_t), p.value=0.01))
DEGs_clean <- DEGs_clean[DEGs_clean$logCPM > 1, ] # 10940 obs. of 8 variables

DEGs_clean$class <- "="
DEGs_clean$class[which(DEGs_clean$logCPM>1&DEGs_clean$logFC>(1.5)&DEGs_clean$FDR<0.01)] <- "+"
DEGs_clean$class[which(DEGs_clean$logCPM>1&DEGs_clean$logFC<(-1.5)&DEGs_clean$FDR<0.01)] <-  "-"
DEGs_clean <- DEGs_clean[order(DEGs_clean$logFC,decreasing = T),]
head(DEGs_clean,5)
```

```
##                 ensembl_gene_id external_gene_name length    logFC   logCPM
## ENSG00000267978 ENSG00000267978            MAGEA9B   5806 10.85332 3.209478
## ENSG00000104371 ENSG00000104371               DKK4   3166 10.67517 4.317621
## ENSG00000123584 ENSG00000123584             MAGEA9   5807 10.65832 3.110258
## ENSG00000177243 ENSG00000177243           DEFB103B   1460 10.54035 2.290118
## ENSG00000006059 ENSG00000006059             KRT33A   4736 10.45522 1.546511
##                         F       PValue          FDR class
## ENSG00000267978 235.35897 3.188515e-28 7.620210e-27     +
## ENSG00000104371  96.12085 2.256596e-16 1.114926e-15     +
## ENSG00000123584 229.85905 7.389562e-28 1.663126e-26     +
```

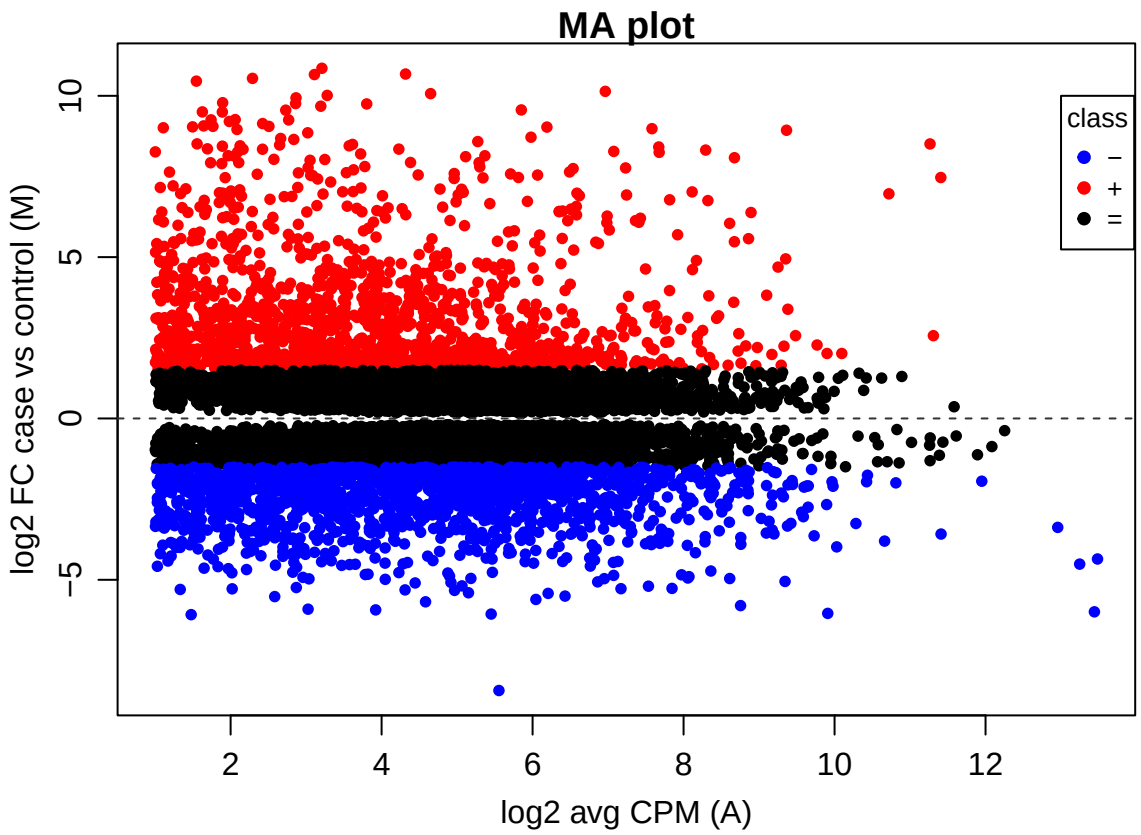
```
## ENSG00000177243 108.66022 9.613693e-18 5.566659e-17     +
## ENSG00000006059 148.75315 1.257354e-21 1.170550e-20     +
```

```
table(DEGs_clean$class)
```

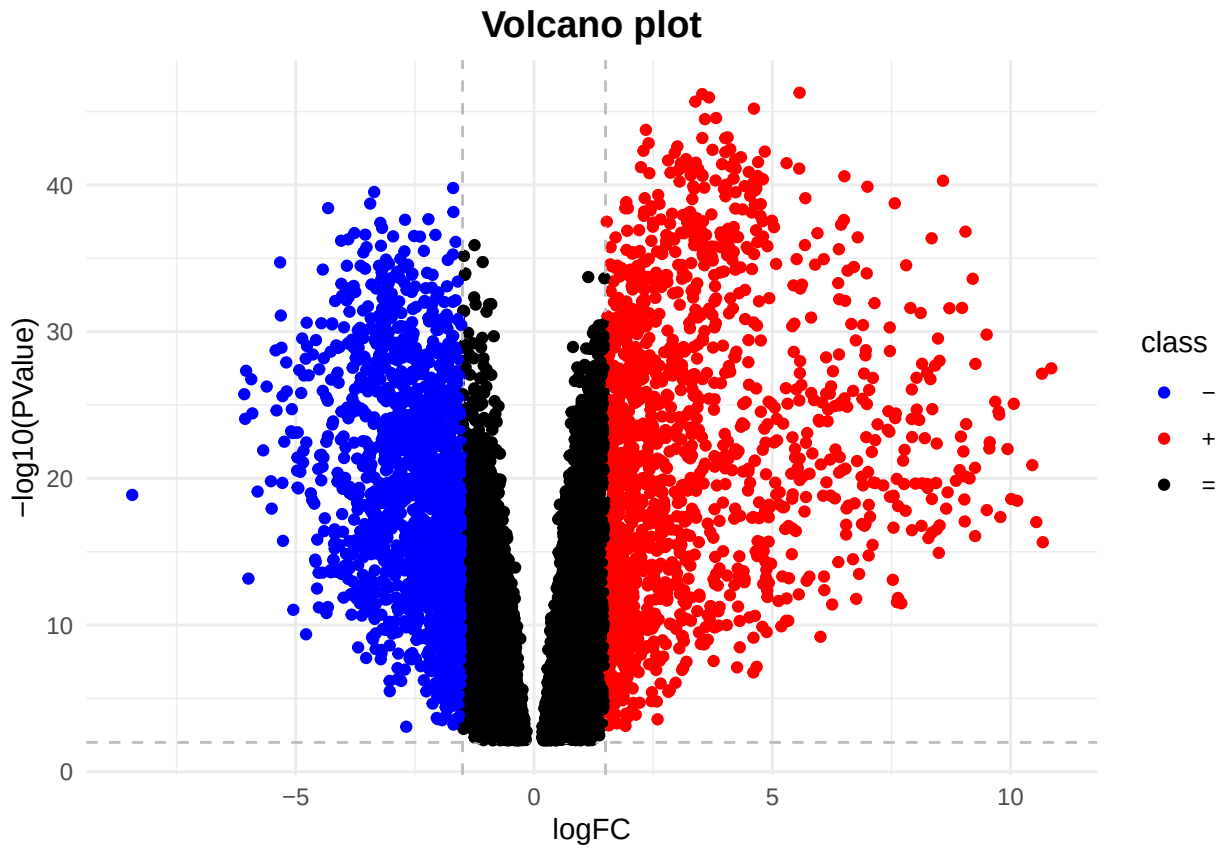
```
##
##    -    +    =
## 1625 1504 7811
```

We have decided not to relax the threshold since the number of up-regulated and down-regulated is enough to proceed with the analysis. We have decided to display both the volcano plot and the MA plot, since they can be compared on the basis of the shared log FC values, respectively on the y-axis and on the x-axis. The MA plot does not provide information about the magnitude of the p-value, while the volcano plot does not include the information about average expression.

```
# display MA plot
input_df_MA <- DEGs_clean
xlabel <- "log2 avg CPM (A)"
ylabel <- "log2 FC case vs control (M)"

palette(c("blue", "red", "black"))
par(fig=c(0,1,0,1), mar=c(4,4,1,2), mgp=c(2, 0.75, 0))
plot(input_df_MA$logCPM, input_df_MA$logFC, xlab=xlabel, ylab=ylabel,
     col= factor(input_df_MA$class), frame.plot=TRUE, cex=0.8, pch=16, main="MA plot", )
abline(h=0,lty=2,col="grey20")
legend(13, 10, legend=c("-", "+", "="),
       col=c("blue", "red", "black"), cex=0.8, title="class", pch=c(19,19,19))
```

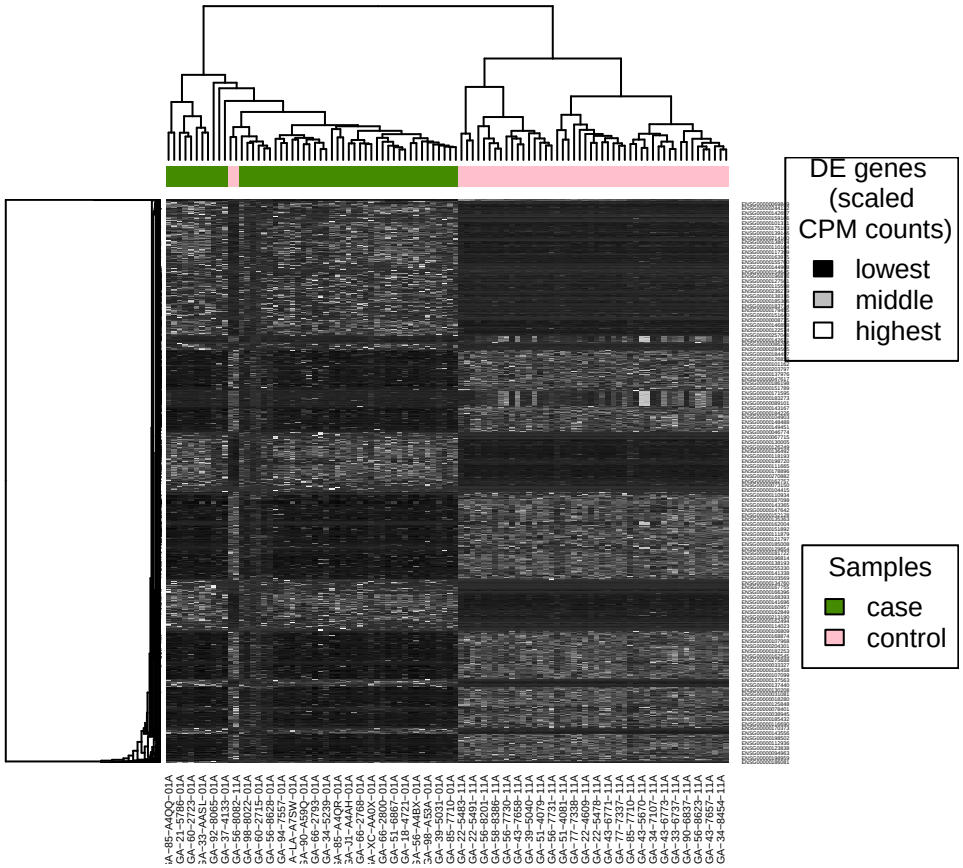



```
# Display volcano plot
pVO <- ggplot(data = input_df_MA, aes(x = logFC, y = -log10(PValue), col=class, label="")) +
  ggtitle("Volcano plot") +
  geom_point() +
  theme_minimal() +
  scale_color_manual(values = c("blue", "red", "black")) +
  geom_vline(xintercept = c(-1.5, 1.5), col = "gray", linetype = 'dashed') +
  geom_hline(yintercept = -log10(0.01), col = "gray", linetype = 'dashed') + geom_point(size = 1)
pVO + theme(plot.title = element_text(color="black", size=14, face="bold",hjust = 0.5))
```



Then, we have created an annotated heatmap, focusing only on up and down-regulated genes.

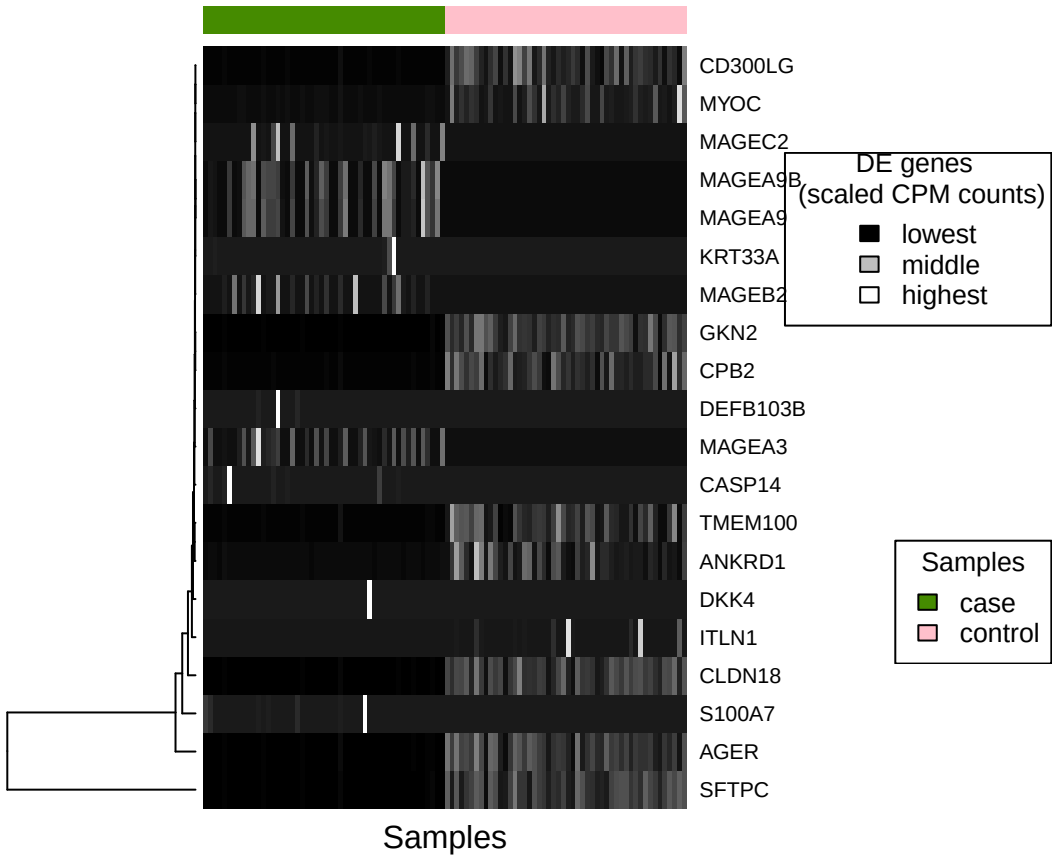
```
cols <- c(rep("chartreuse4",100))
cols[grep('11A',colnames(cpm_table))]<-'pink' # case - control
pal <- c("black", "grey", "white")
pal <- colorRampPalette(pal)(100)

# Heatmap representing all DE genes

heatmap(as.matrix(cpm_table[which(rownames(cpm_table) %in%
        DEGs_clean$ensembl_gene_id[which(DEGs_clean$class!="=")]),]), ColSideColors = cols,
        cexCol = 0.5,margins = c(4,4),col=pal,cexRow = 0.2, Rowv = TRUE)
legend("topright", legend=c("lowest", "middle", "highest"), fill = c("black", "grey", "white"),
       cex = 0.8, title="DE genes \n (scaled \n CPM counts)")
legend("bottomright", legend=c("case", "control"), fill = c("chartreuse4", "pink"),
       cex = 0.8, title="Samples")
```



We note that the *Colv* parameter has its default value, so the column dendogram is reordered and a control sample gets included in a cluster belonging to the case side. Moreover, we cannot actually verify that the heatmap is properly constructed, due to the high number of genes. So, we have decided to make a reduced version of this heatmap, focusing only on the top 20 differentially expressed genes, 10 from the up-regulated subset and 10 from the down-regulated subset, respectively. First, we subset the DEGs dataset into two different ones (*upreg_df* and *downreg_df*), respectively for up and down-regulated genes. Second, we order them according to logFC (from maximum positive to minimum positive value for up-regulated genes, from minimum negative to maximum negative for down-regulated genes). Then, we have to order the samples of *c_anno_df* to split case and control as two subgroups. We have selected the top 10 DE genes (by *ensembl_gene_id*) from *upreg_df* and *downreg_df*, respectively. In order to properly configure the heatmap matrix, we had to sort the *cpm_table* columns (samples), matching the two subgroups (case-control) of *ordered_c_anno_df*. Finally, we have displayed the reduced version of the heatmap, replacing ensemble gene IDs with gene names.

```
upreg_df <- DEGs_clean[DEGs_clean$class=="+",]
downreg_df <- DEGs_clean[DEGs_clean$class=="-",]

# Ordering up and down regulated genes by log fold change
upreg_df <- upreg_df[order(-upreg_df$logFC),]
downreg_df <- downreg_df[order(downreg_df$logFC),]

# Ordering samples in case-control for the heatmap
ordered_c_anno_df <- c_anno_df
ordered_c_anno_df$condition <- as.character(ordered_c_anno_df$condition)
ordered_c_anno_df <- ordered_c_anno_df[with(ordered_c_anno_df, order(condition)), ]
```

```r
# Heatmap representing the top 20 differentially expressed genes
# 10 from the up-regulated subset and 10 from the down-regulated sub-set

heatmap_sub_matrix <- as.matrix(cpm_table[rbind(head(upreg_df, 10),
                      head(downreg_df, 10))$ensembl_gene_id,])[, ordered_c_anno_df$sample]
heatmap_sub_matrix <- as.data.frame(heatmap_sub_matrix)
# Substitute ensembl_gene_id with external_gene_name
genename_sub_heatmap <- merge(DEGs_clean, heatmap_sub_matrix, by="row.names")
genename_sub_heatmap <- subset(genename_sub_heatmap, select = -c(length, logFC, logCPM, F,
                        PValue, FDR, class, ensembl_gene_id, Row.names))
row.names(genename_sub_heatmap) <-genename_sub_heatmap$external_gene_name
genename_sub_heatmap <- subset(genename_sub_heatmap, select = -c(external_gene_name))

cols <- c(rep("chartreuse4",50),rep('pink',50))
heatmap(as.matrix(genename_sub_heatmap), ColSideColors = cols, cexCol = 2,
        margins = c(2,13),col = pal, cexRow = 1, Colv = NA, Rowv = TRUE,
        xlab = "Samples", labCol = FALSE)
legend("topright", legend=c("lowest", "middle", "highest"), fill = c("black", "grey", "white"),
       cex = 0.8, title="DE genes \n (scaled CPM counts)")
legend("bottomright", legend=c("case", "control"), fill = c("chartreuse4", "pink"),
       cex = 0.8, title="Samples")
```



By looking at this heatmap, we can clearly distinguish the up-regulated from the down-regulated genes. For example, SFTPC and AGER genes have higher CPM counts in the control samples, confirming their down-regulation from the DE analysis. Also, genes belonging to the 'MAGE' family, found to be up-regulated, have higher CPM counts in the case samples with respect to control ones. For the latter case, we have found

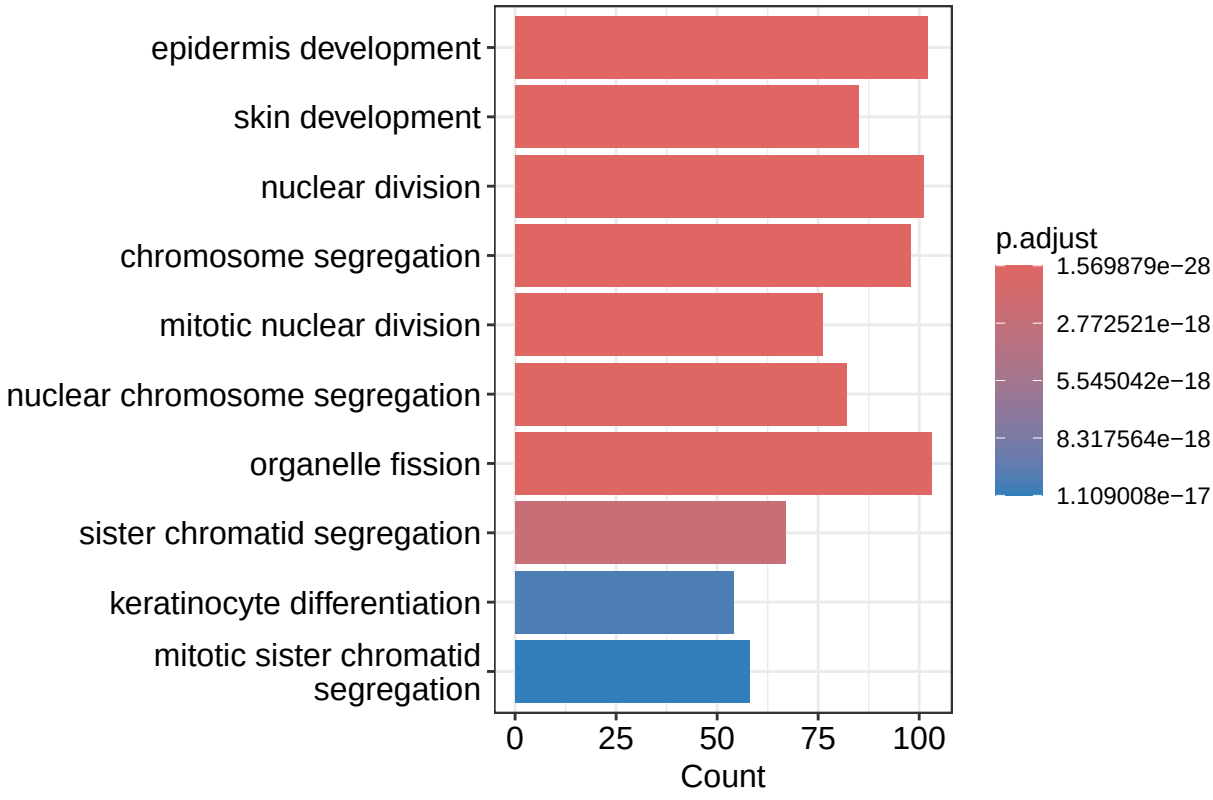
biological validation from literature (Hoang & Landi, 2022, https://doi.org/10.3390/cancers14040961).

4. Perform gene set enrichement analysis

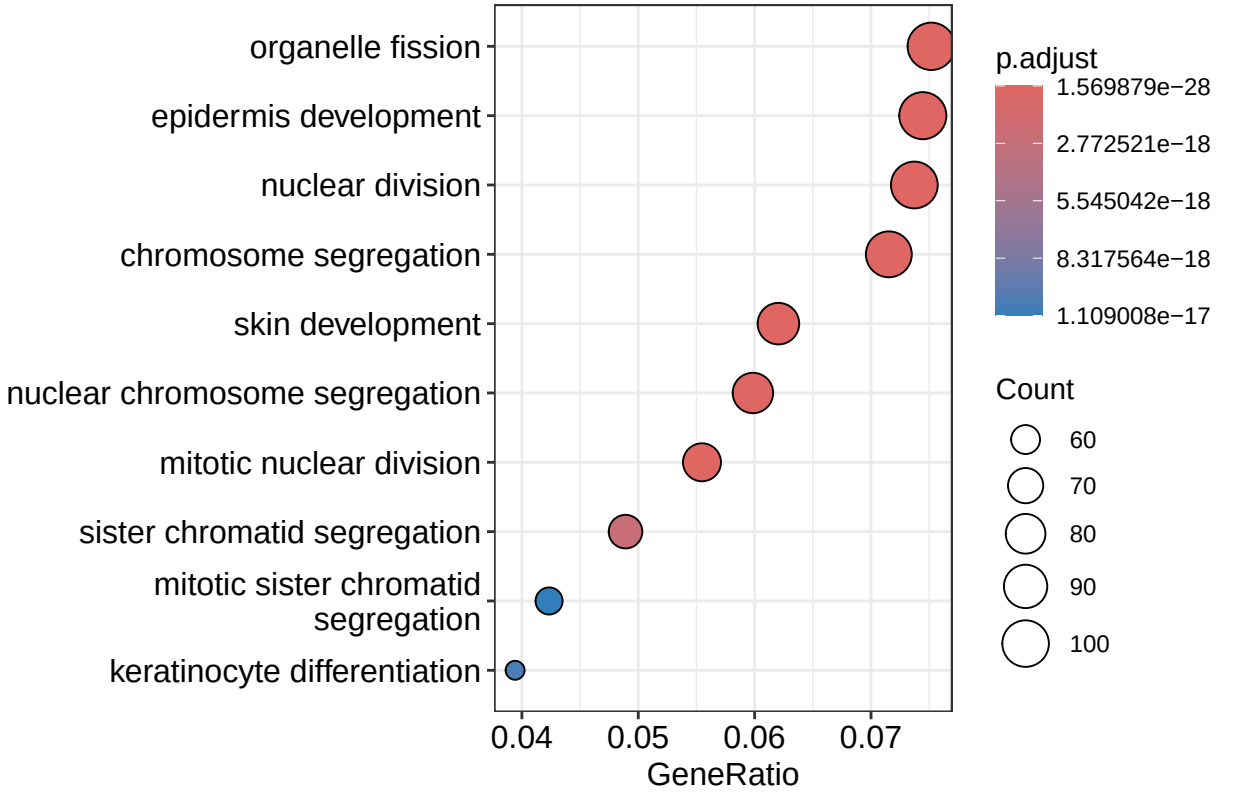
Let's start by performing GO (BP and MF) analysis, focusing on the top 10 enriched GO terms, respectively for the up and down regulated genes.

```
## Perform Gene Ontology enrichment analysis (Biological Process) on up-regulated genes.
ego_BP_up <- enrichGO(gene = upreg_df$external_gene_name,
                      OrgDb = org.Hs.eg.db,
                      keyType = 'SYMBOL',
                      ont = "BP",
                      pAdjustMethod = "BH",
                      pvalueCutoff = 0.05,
                      qvalueCutoff = 0.05)
```

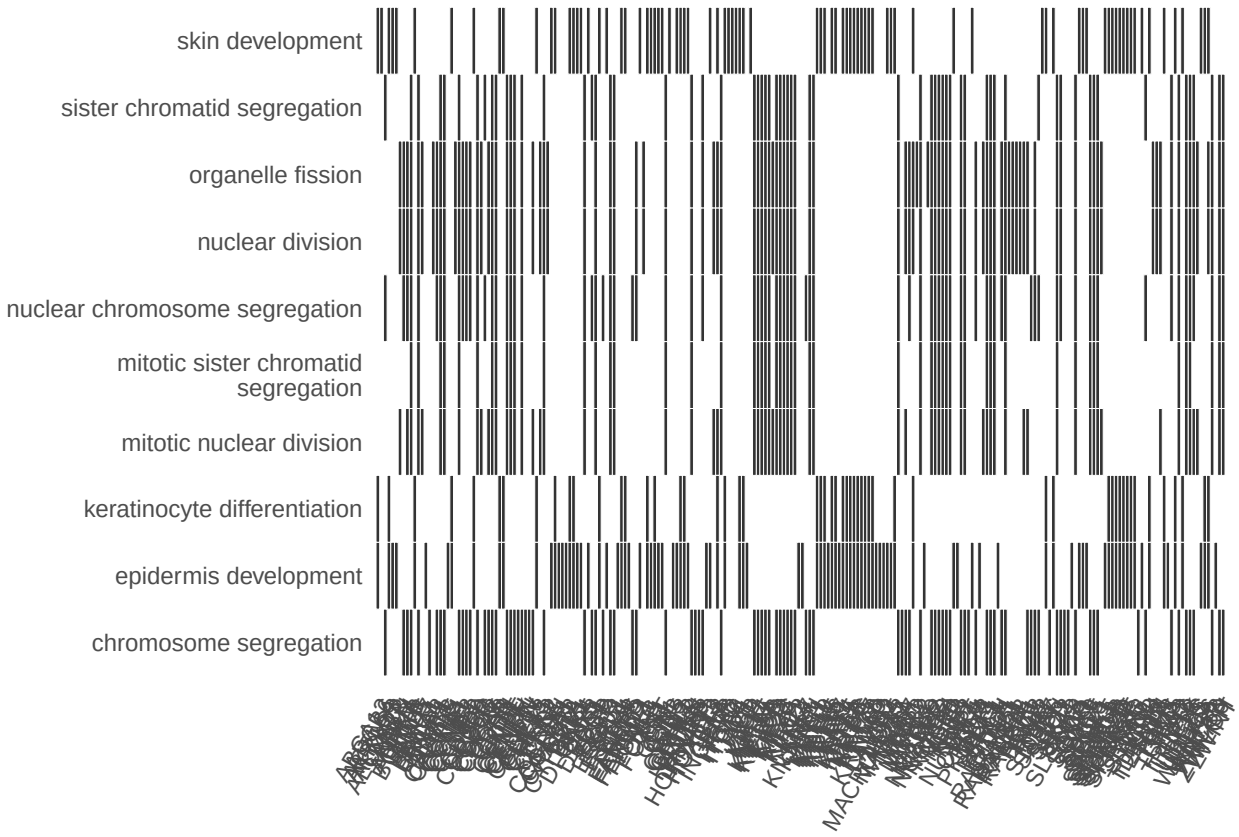
```
## Visualize the top 10 enriched terms
barplot(ego_BP_up, showCategory = 10)
```




```
dotplot(ego_BP_up, showCategory = 10)
```




```
heatplot(ego_BP_up,showCategory =10)
```

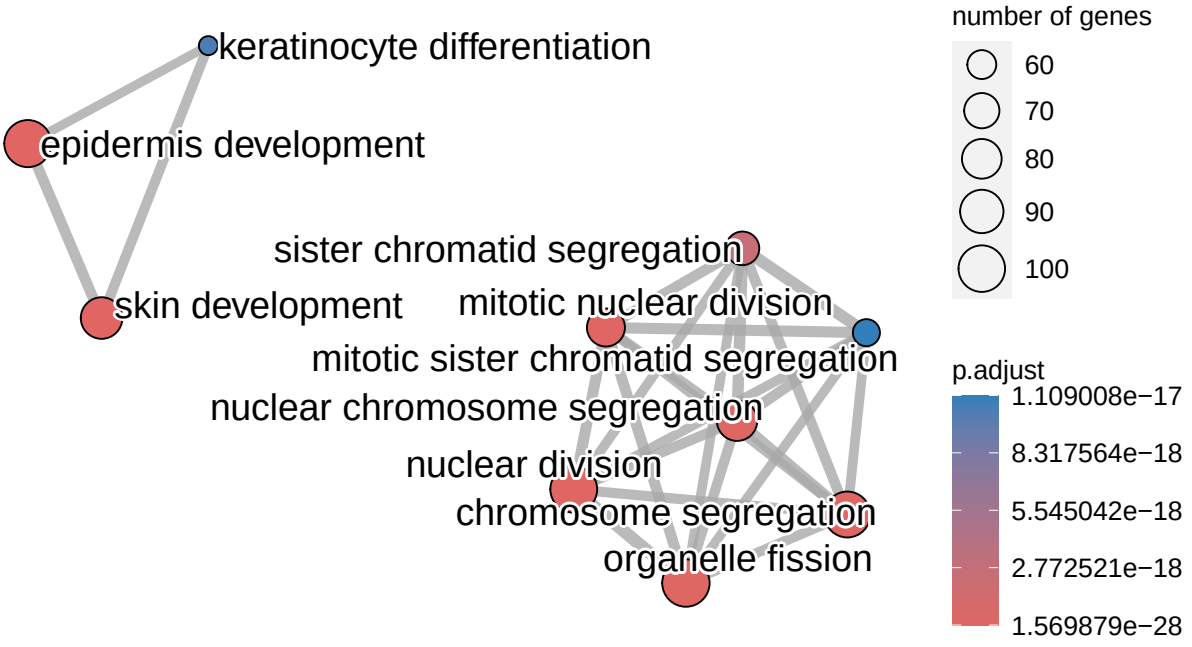


By looking at the first view (barplot) of the GO enrichment analysis performed in the context of Biological Process, we have detected as high ranking terms (by adjusted p-value) epidermis and skin development, along with phenomena related to cellular duplication. Knowing that proliferation is an important part of cancer development and progression, we can unambiguously state that over-expression of genes related to cell cycle is coherent with the provided data, which is about Lung Squamous Cell Carcinoma (LUSC). Additionally, we have found publications that link lung cancer with skin metastasis of the back and hand (Khaja et al., 2019, doi: 10.1159/000501363) and fungating ulcerated skin lesion (Chisenga et al., 2022, https://doi.org/10.1186/s13256-022-03352-4).

In particular, from the second view (dotplot), we can see that the highest number of genes are involved in the process of organelle fission. From literature (Boulton & Caino, 2002, doi: 10.3389/fcell.2022.849962), we have discovered that altered mitochondrial fission and fusion change tumor cell growth, metabolism, motility, and invasion, validating our experiment.

To further analyse the BP enrichment results, we have calculated the pairwise similarities of the enriched term, using the *pairwise_termsim()* function on GO BP for up-regulated genes, which exploits the Jaccard's similarity index (JC).

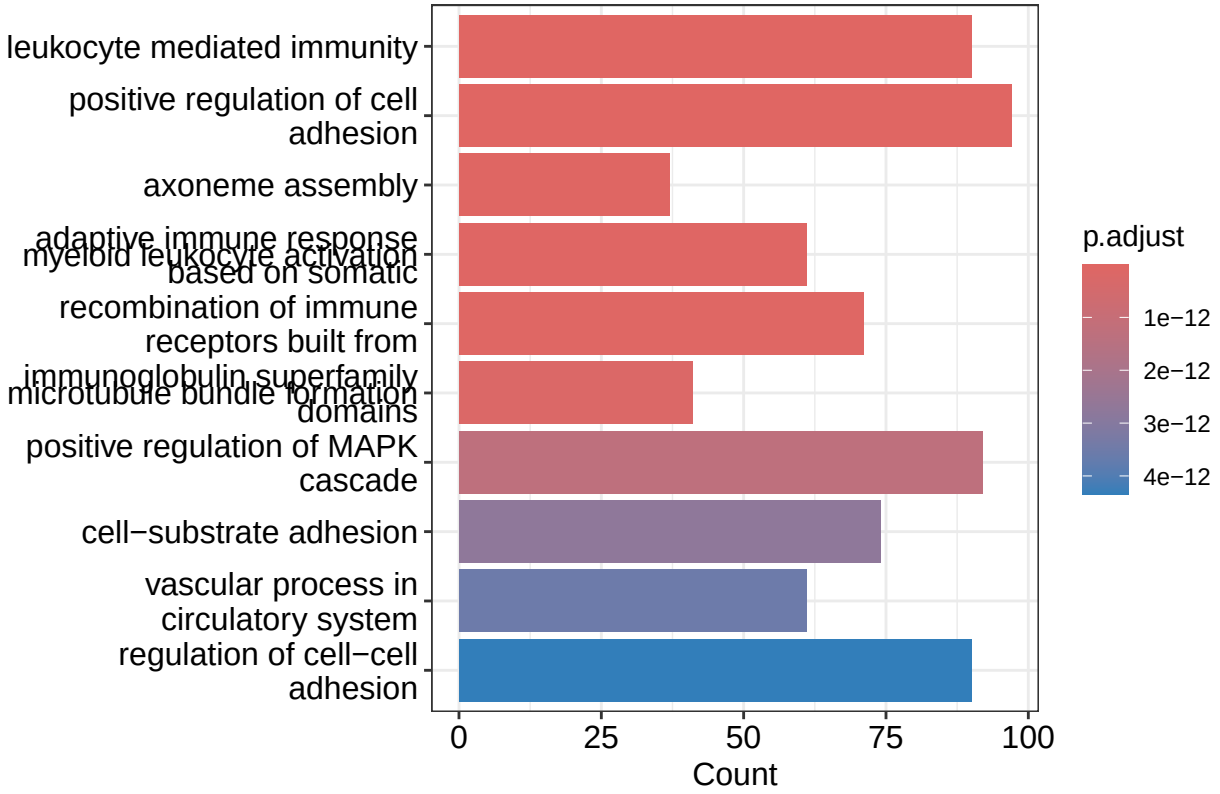
```
x_BP_up <- pairwise_termsim(ego_BP_up)
emapplot(x_BP_up,showCategory = 10)
```



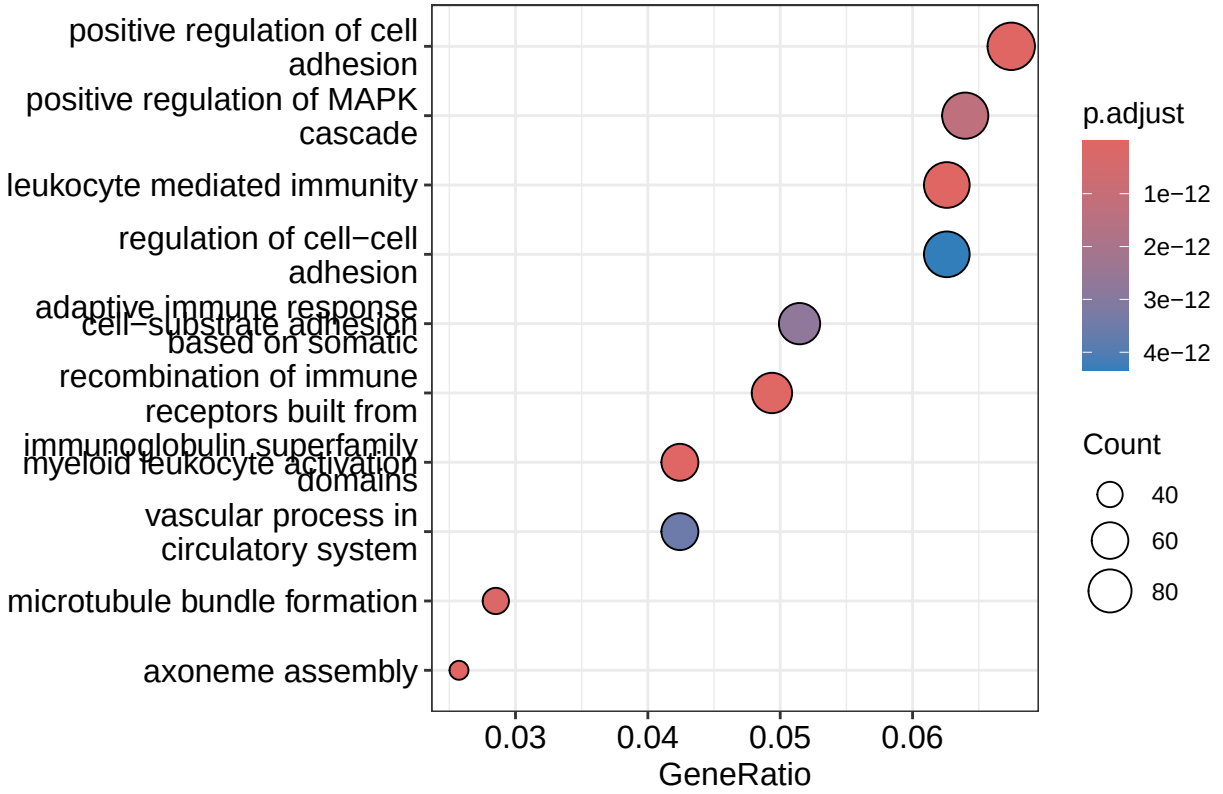
As expected, there are two main connected components, the bigger one regarding cellular duplication and the smaller one (triangle) concerning skin and epidermis development.

```
## Perform Gene Ontology enrichment analysis (Biological Process) on down-regulated genes
ego_BP_down <- enrichGO(gene = downreg_df$external_gene_name,
                    OrgDb = org.Hs.eg.db,
                    keyType = 'SYMBOL',
                    ont = "BP",
                    pAdjustMethod = "BH",
                    pvalueCutoff = 0.05,
                    qvalueCutoff = 0.05)
```

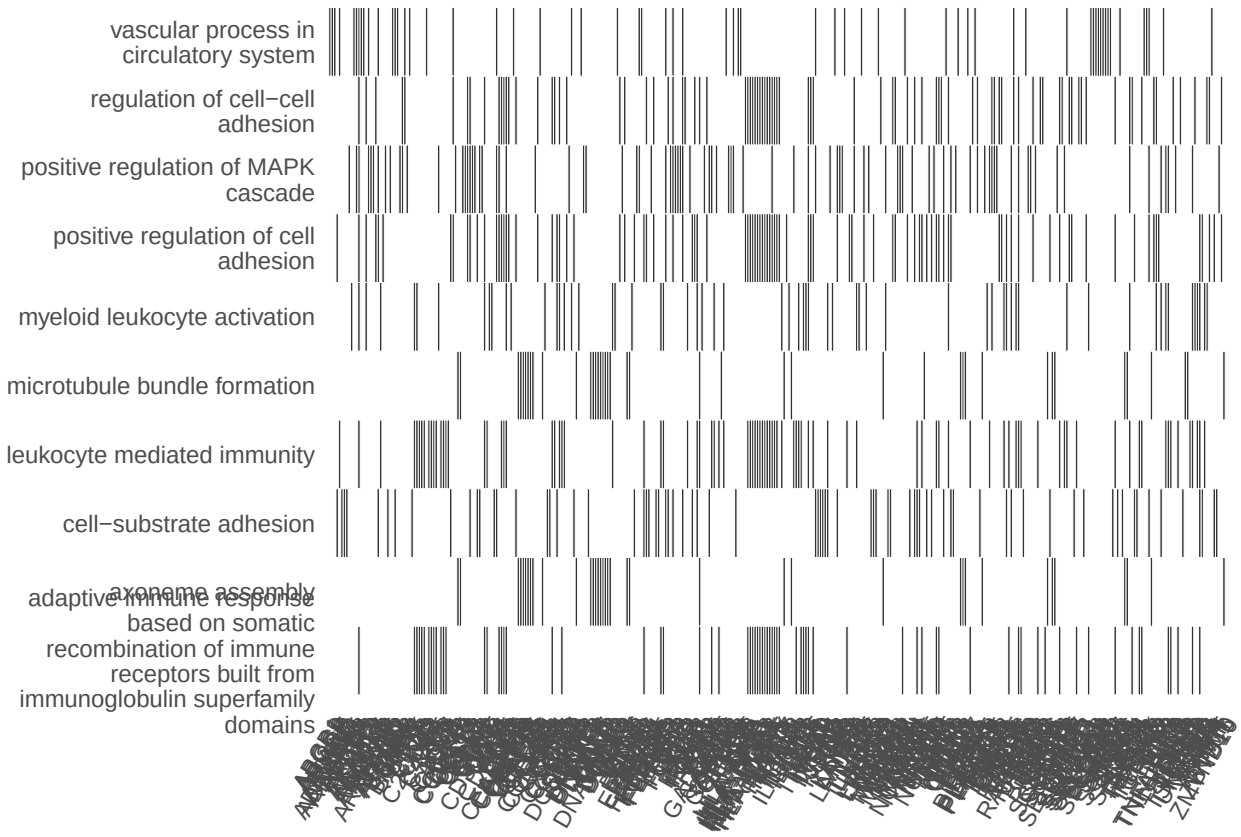
```
## Visualize the top 10 enriched terms
barplot(ego_BP_down, showCategory = 10)
```




```
dotplot(ego_BP_down, showCategory=10)
```



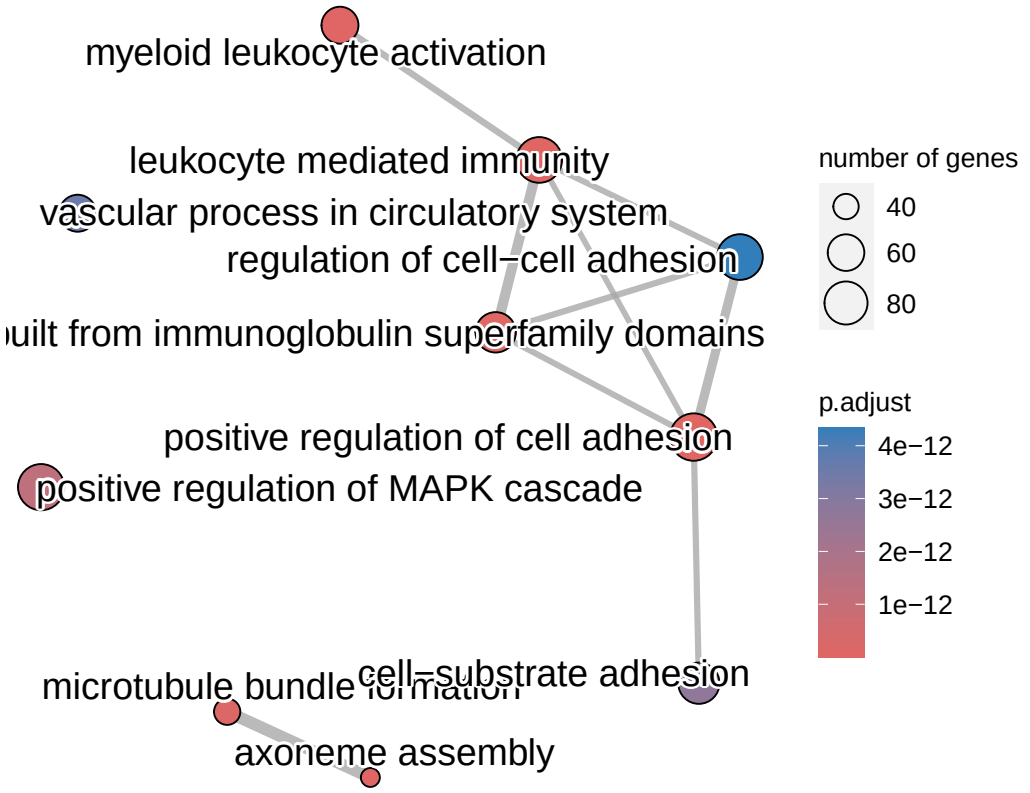

```
heatplot(ego_BP_down,showCategory =10)
```



By looking at the first view (barplot) of the GO enrichment analysis performed in the context of Biological Process, we have detected as high ranking terms (by adjusted p-value) leukocyte mediated immunity and regulation of cell adhesion. Knowing that decreased adhesion strength corresponds to increased metastatic potential (Beri et al., 2020, doi: 10.1158/0008-5472.CAN-19-1794), and that immune response is functionally impacted by cancer proliferation, we can safely state that under-expression of genes related to these processes is consistent with our experimental data.

In particular, from the second view (dotplot), we note that a high number of genes is involved in the positive regulation of MAPK cascade. We know from literature that this pathway has a key role in cell differentiation and apoptosis (Plotnikov et al., 2011, https://doi.org/10.1016/j.bbamcr.2010.12.012) and its regulation is disrupted by several diseases, including lung cancer.

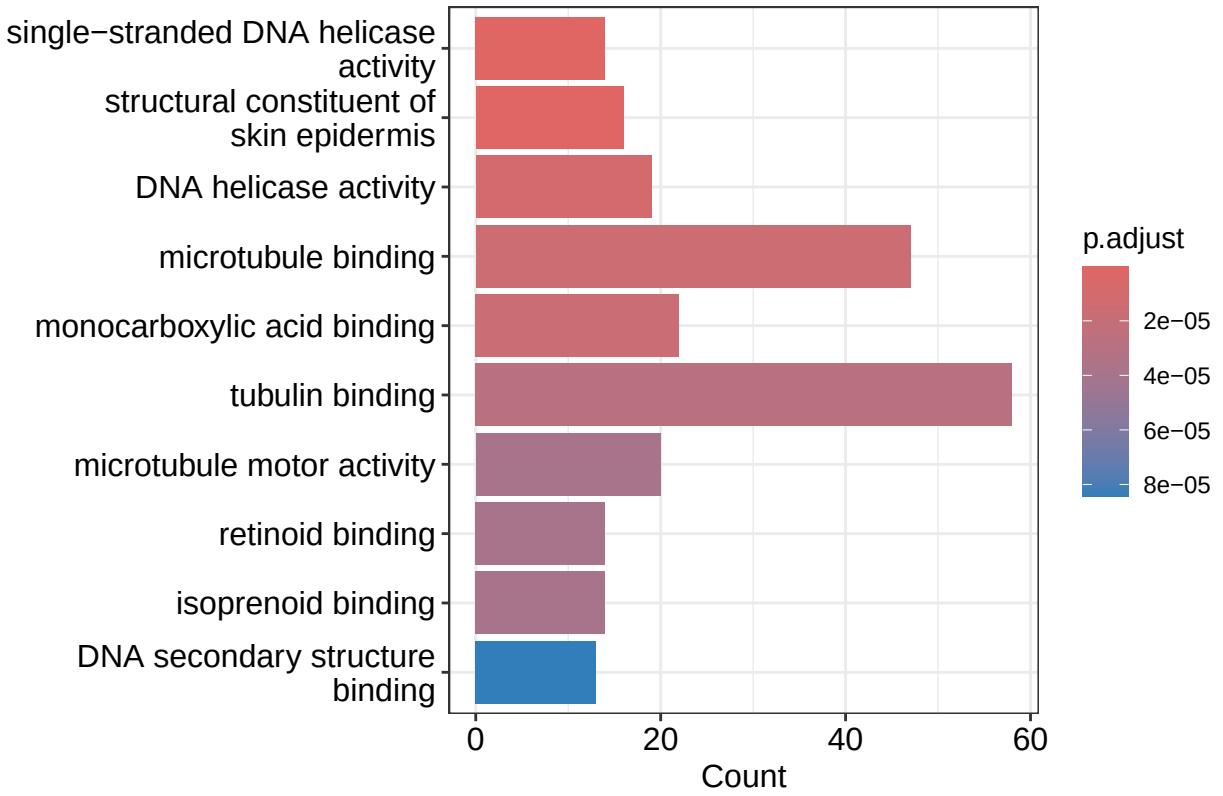
```
x_BP_down<- pairwise_termsim(ego_BP_down)
emapplot(x_BP_down,showCategory = 10)
```



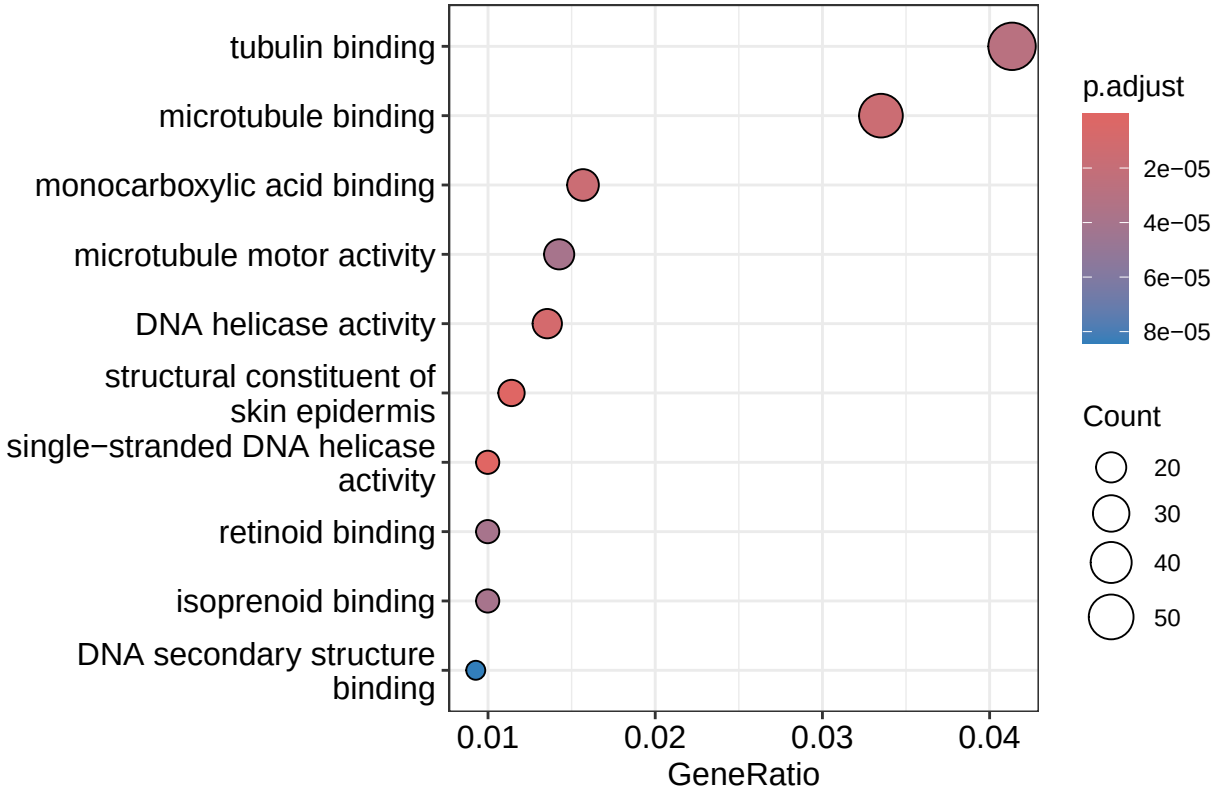
As expected, there is a main connected component, regarding immune response and cell adhesion. MAPK cascade regulation is instead isolated, as well as vascular process in circulatory system. Nevertheless, these two processes remain relevant in the cancer context, according to literature.

```
## Perform Gene Ontology enrichment analysis (Molecular Function) for up-regulated genes
ego_MF_up <- enrichGO(gene = upreg_df$external_gene_name,
                      OrgDb = org.Hs.eg.db,
                      keyType = 'SYMBOL',
                      ont = "MF",
                      pAdjustMethod = "BH",
                      pvalueCutoff = 0.05,
                      qvalueCutoff = 0.05)
```

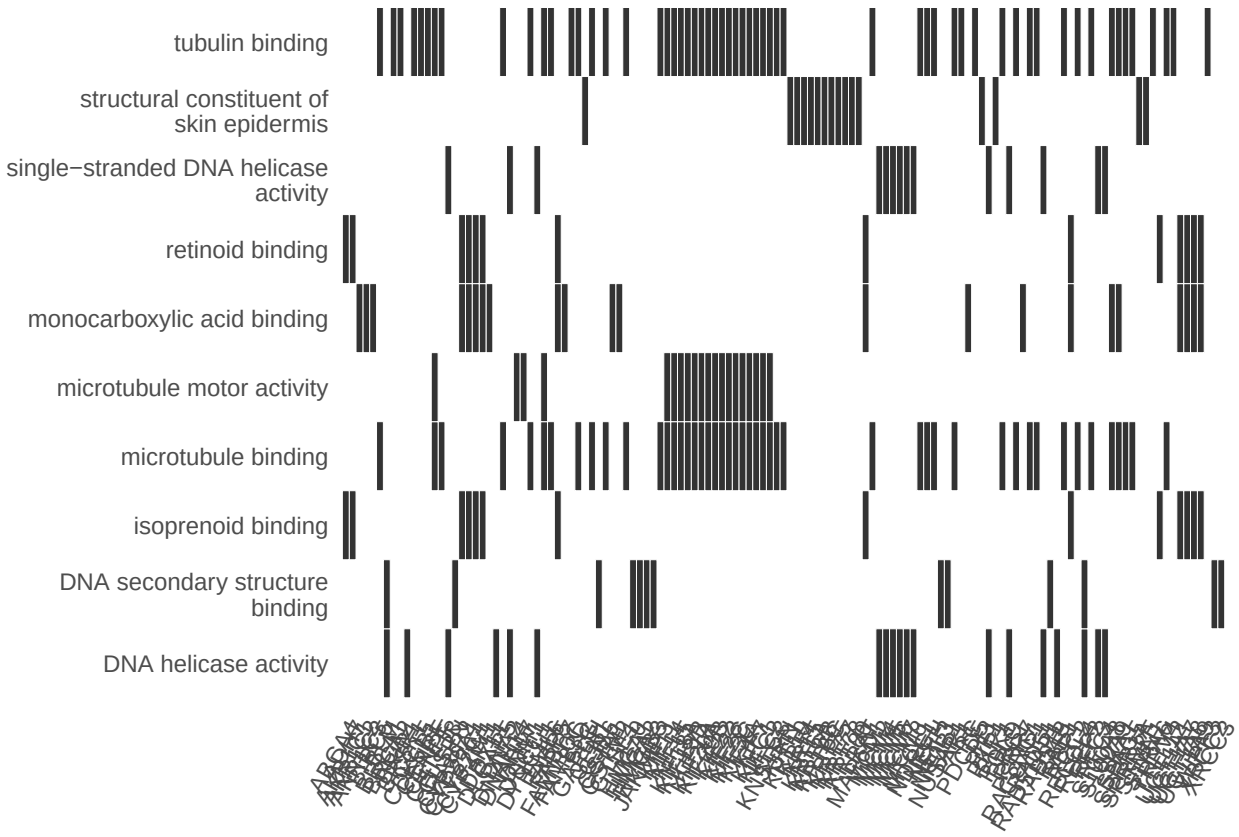
```
## Visualize the top 10 enriched terms
barplot(ego_MF_up, showCategory = 10)
```




```
dotplot(ego_MF_up, showCategory = 10)
```

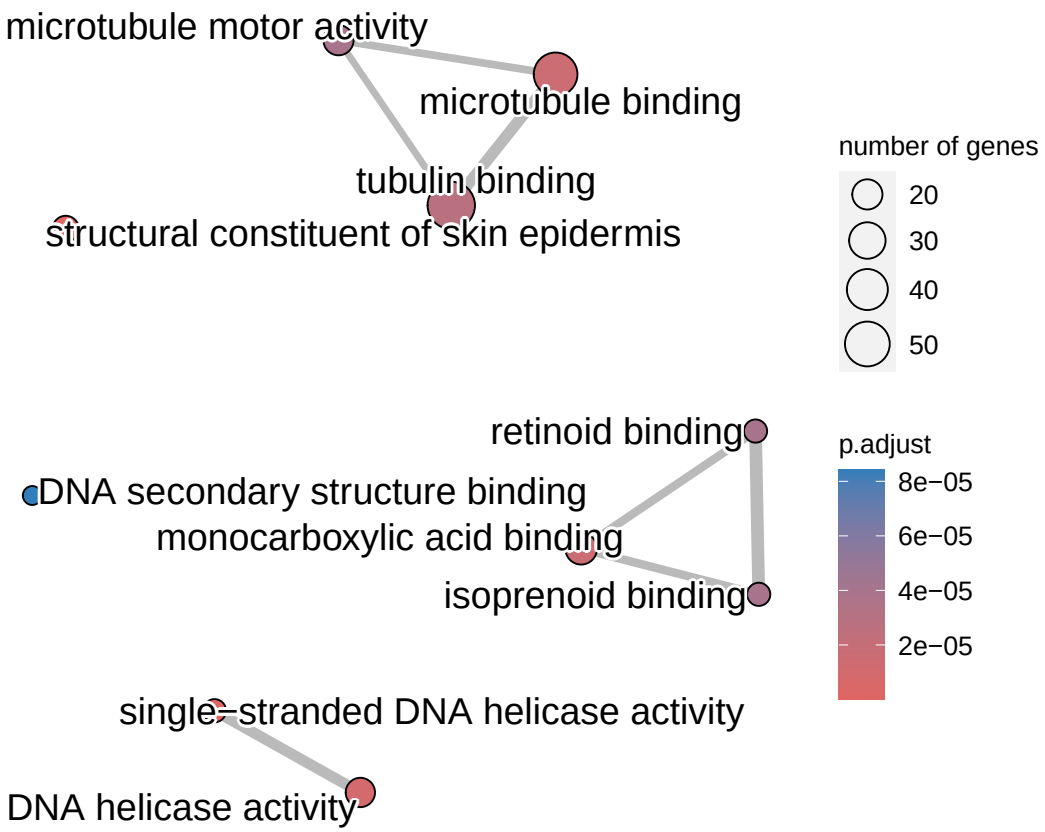



```
heatplot(ego_MF_up,showCategory =10)
```



By looking at the first view (barplot) of the GO enrichment analysis performed in the context of Molecular Function, we have detected as high ranking terms (by adjusted p-value) single-stranded DNA helicase activity and structural constituent of skin epidermis. In particular, from the second view (dotplot), we note that a high number of genes is involved in tubulin and microtubule binding. These functions are consistent with the above discovered biological processes.

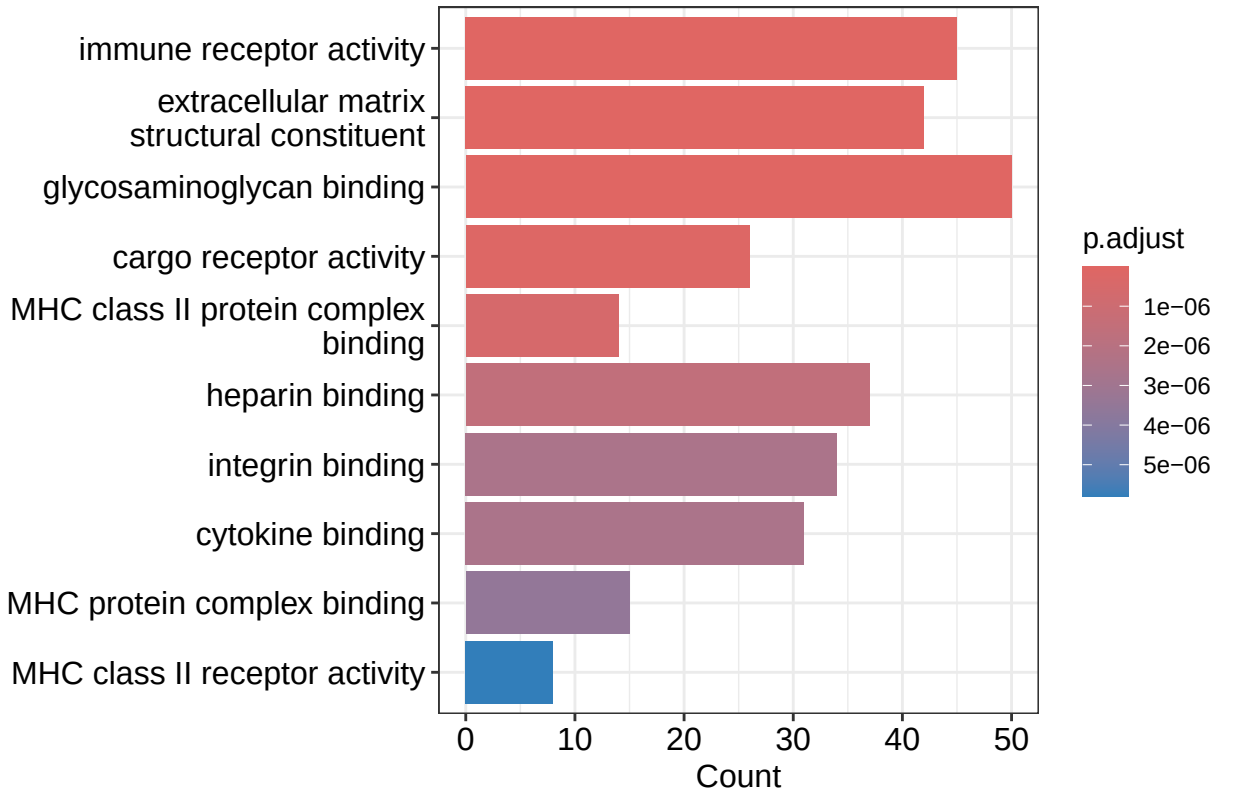
```
x_MF_up<- pairwise_termsim(ego_MF_up)
emapplot(x_MF_up,showCategory = 10)
```



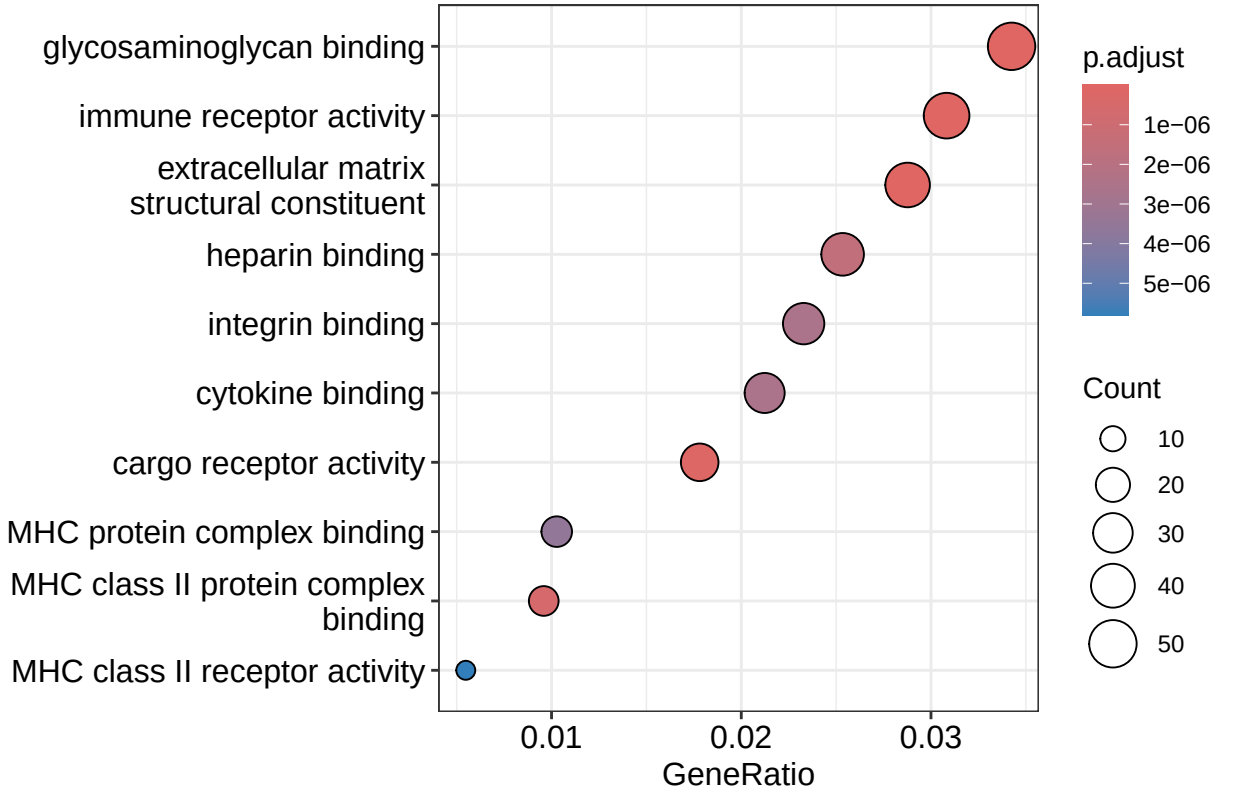
Pairwise-similarity analysis shows three main connected components: microtubule-related, helicase-related terms and a triangle of specific binding, including retinoid and isoprenoid. In particular, the over-expression of retinoid receptors (Tripathi et al., 2019 doi: 10.1016/j.phrs.2019.104331) is contemplated in lung cancer.

```
## Perform Gene Ontology enrichment analysis (Molecular Function) for down-regulated genes
ego_MF_down <- enrichGO(gene = downreg_df$external_gene_name,
                  OrgDb = org.Hs.eg.db,
                  keyType = 'SYMBOL',
                  ont = "MF",
                  pAdjustMethod = "BH",
                  pvalueCutoff = 0.05,
                  qvalueCutoff = 0.05)
```

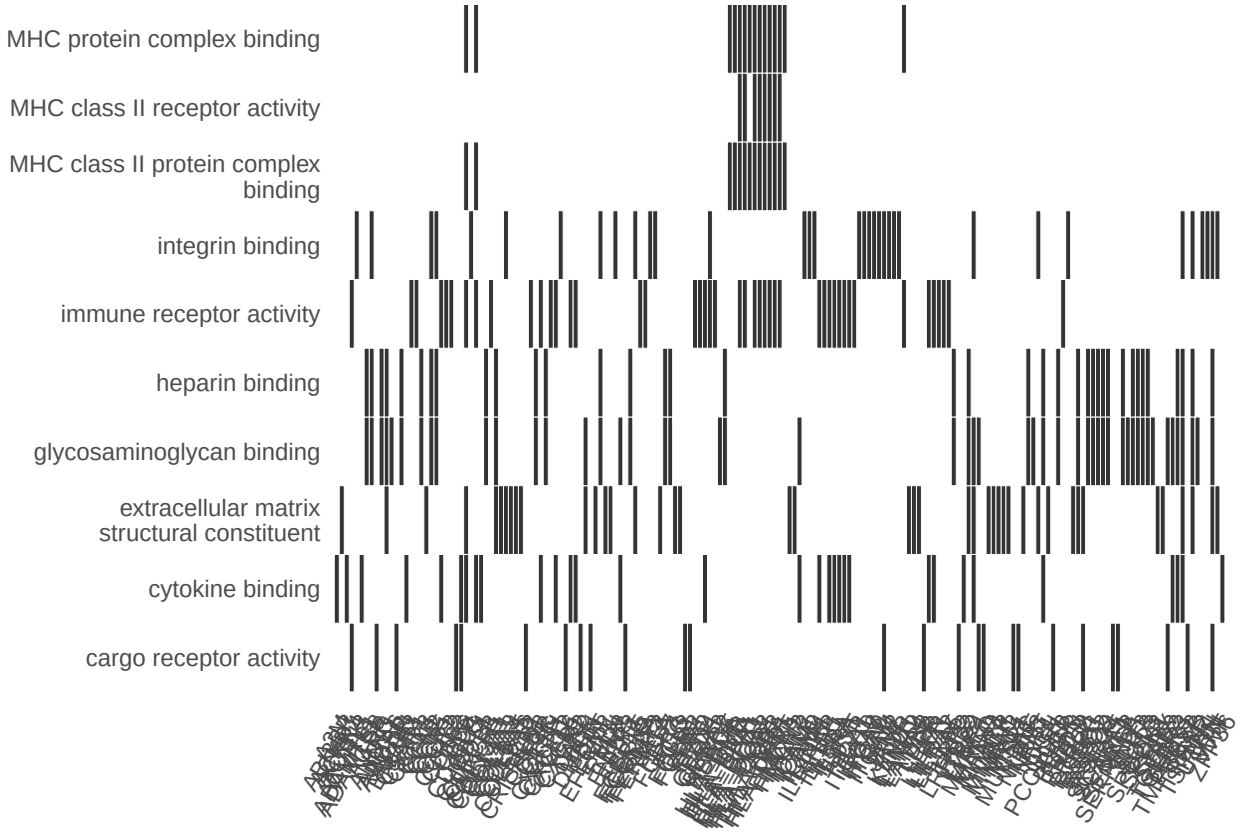
```
## Visualize the top 10 enriched terms
barplot(ego_MF_down, showCategory = 10)
```




```
dotplot(ego_MF_down, showCategory = 10)
```

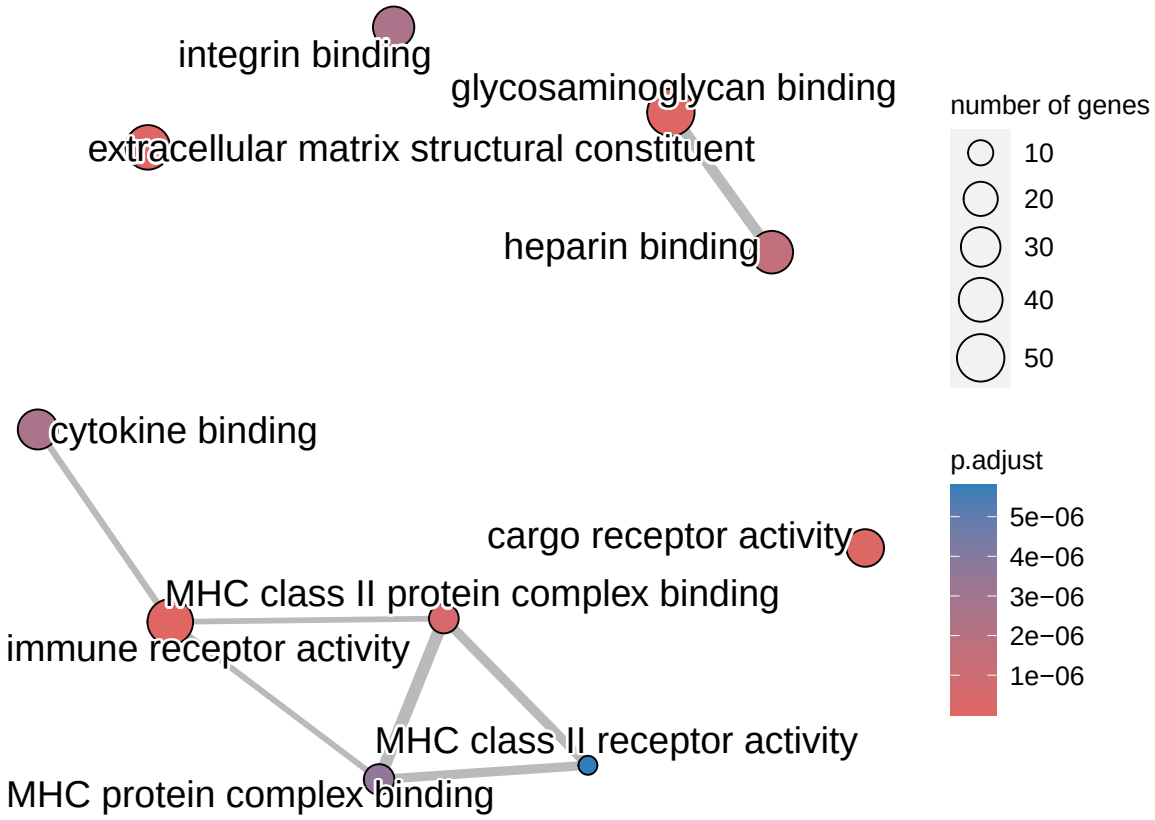



```
heatplot(ego_MF_down,showCategory = 10)
```



By looking at the first view (barplot) of the GO enrichment analysis performed in the context of Molecular Function, we have detected as high ranking terms (by adjusted p-value) immune receptor activity and extracellular matrix structural constituent. In particular, from the second view (dotplot), we note that a high number of genes is involved in glycosaminoglycan binding. The latter protein is involved in cancer development and progression (Wei et al., 2020, doi: 10.3390/ijms21175983) and it is one of the main components of the extracellular matrix, influencing cell behavior, matrix properties and immune supervision. These functions are consistent with the above discovered biological processes (immune response and cell adhesion).

```
x_MF_down<- pairwise_termsim(ego_MF_down)
emapplot(x_MF_down,showCategory = 10)
```



Pairwise-similarity detects a bigger connected component related to extracellular matrix and an edge between heparin and glycosaminoglycan binding. Integrin binding results in a isolated node. Both heparin and glycosaminoglycan are consistent with the above discovered biological processes. Additionally, overexpression of integrin, due to the under-expression of its receptors, may provide a broad advantage to cancer by protecting it from apoptosis. Indeed integrin up-regulation promotes proliferation and invasion (Janes and Watt, 2006, https://doi.org/10.1038/nrc1817).

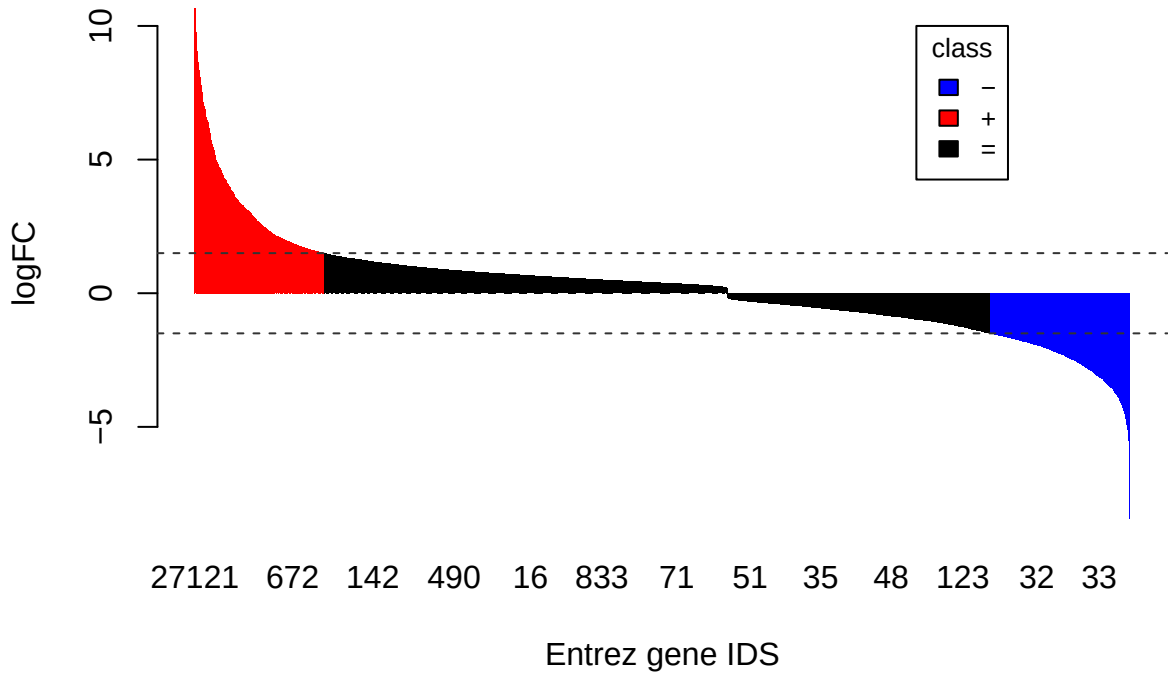
In order to use up-regulated and down-regulated gene subsets for the construction of WP Pathways, we need to include the *Entrez IDs* to the above-produced *DEGs_clean* dataset.

```
# Use biomaRt to map Gene symbols, Entrez IDs and Ensembl gene IDs from DGEList object (DEGs_clean)
convert <- getBM(attributes=c("ensembl_gene_id","entrezgene_id","external_gene_name"),
                 filters=c("ensembl_gene_id"),
                 values=DEGs_clean$ensembl_gene_id,
                 mart = ensembl)
DEGs_entrez <- merge(DEGs_clean,convert,by.x="ensembl_gene_id",by.y="ensembl_gene_id")
DEGs_entrez <- DEGs_entrez[1:nrow(DEGs_entrez),1:(ncol(DEGs_entrez)-1)]
colnames(DEGs_entrez)[2] <- "external_gene_name"
DEGs_entrez <- DEGs_entrez[which(!is.na(DEGs_entrez$entrezgene_id)),]
DEGs_entrez <- DEGs_entrez[-which(duplicated(DEGs_entrez$entrezgene_id)),]
```

```
ranks <- DEGs_entrez[, c("logFC","class")]
# associated each Entrez ID to the corresponding logFC
row.names(ranks) <- DEGs_entrez$entrezgene_id
ranks <- ranks[order(-ranks$logFC),]
head(ranks,5)
```

```
##              logFC class
## 27121     10.67517     +
## 4108      10.65832     +
## 728269    10.65832     +
## 414325    10.54035     +
## 124906744 10.54035     +
```

```
class_cols <- c(rep("red", sum(ranks$class == "+")),rep("black", sum(ranks$class == "=")),
                rep("blue", sum(ranks$class == "-")))

barplot(height = ranks$logFC, names = rownames(ranks), col=class_cols,
        xlab = "Entrez gene IDS", ylab = "logFC", border=NA)
abline(h=min(ranks$logFC[ranks$class == "+"]),lty=2,col="grey20")
abline(h=max(ranks$logFC[ranks$class == "-"]),lty=2,col="grey20")
legend(10000, 10, legend=c("-", "+", "="),
       fill=c("blue", "red", "black"), cex=0.8, title="class")
```



Having the gene *Entrez IDs*, we have to update the up and down-regulated gene subsets.

```
upreg_df_entrez <- DEGs_entrez[DEGs_entrez$class=="+",]
downreg_df_entrez <- DEGs_entrez[DEGs_entrez$class=="-",]
```

We can now perform WP (over-representation) analysis on up and down-regulated genes, reporting the top 10 enriched pathways.

```
eWP_up = enrichWP(gene = upreg_df_entrez$entrezgene_id,
                  organism = "Homo sapiens",
                  pvalueCutoff = 0.05,
                  qvalueCutoff = 0.05)
head(eWP_up, n=5)
```

```
##            ID                                                Description
## WP2446 WP2446                           Retinoblastoma gene in cancer
## WP179   WP179                                                Cell cycle
## WP466   WP466                                             DNA replication
## WP2840 WP2840 Hair follicle development cytodifferentiation part 3 of 3
## WP2361 WP2361                                      Gastric cancer network 1
##        GeneRatio  BgRatio       pvalue     p.adjust       qvalue
## WP2446    32/762  89/8389 2.513516e-12 1.412596e-09 1.299091e-09
## WP179     35/762 120/8389 2.081328e-10 5.848532e-08 5.378590e-08
## WP466     19/762  42/8389 7.738618e-10 1.449701e-07 1.333214e-07
## WP2840    28/762  87/8389 1.151397e-09 1.617712e-07 1.487726e-07
## WP2361    15/762  28/8389 2.511726e-09 2.823181e-07 2.596332e-07
##
## WP2446           1870/54443/4175/4998/24137/8318/8317/5427/1869/4173/898/7272/5947/3925/7153/5111/598
## WP179  10926/1870/4171/4175/4998/23594/8318/990/8317/4174/1869/4173/898/7272/991/9088/5111/8900/891/9
## WP466                                                                                  10926/55388/4171/4175/49
## WP2840                  1747/8626/2736/27121/23581/22943/8456/64065/7474/4488/10804/6662/64919/43847
## WP2361                                                                                              6790/
##        Count
## WP2446    32
## WP179     35
## WP466     19
## WP2840    28
## WP2361    15
```

It is interesting to note that Cell cycle and DNA replication are the top2 and top3 WP pathways, in agreement with the GO enrichment analysis on the up-regulated genes. Even though many types of cancer-related pathways are detected, they are not directly linked to lung. However, taking into account the possibility of tumor progression and metastasis, the involvement of other tissues is not to be a-priori excluded.

```
eWP_down = enrichWP(gene = downreg_df_entrez$entrezgene_id,
                  organism = "Homo sapiens",
                  pvalueCutoff = 0.05,
                  qvalueCutoff = 0.05)
head(eWP_down, n=5)
```

```
##            ID                                                  Description
## WP2806 WP2806                                          Complement system
## WP2328 WP2328                                        Allograft rejection
## WP545   WP545                                       Complement activation
## WP558   WP558                          Complement and coagulation cascades
## WP5090 WP5090 Complement system in neuronal development and plasticity
##        GeneRatio  BgRatio       pvalue     p.adjust       qvalue
## WP2806    38/840  95/8389 8.747957e-15 5.012579e-12 4.502896e-12
## WP2328    32/840  90/8389 4.835272e-11 1.385306e-08 1.244446e-08
## WP545     15/840  23/8389 2.058338e-10 3.931426e-08 3.531675e-08
```

```
## WP558     21/840  58/8389 7.329821e-08 1.025936e-05 9.216176e-06
## WP5090    30/840 107/8389 1.056924e-07 1.025936e-05 9.216176e-06
##
## WP2806 6401/1634/729/4036/2219/2160/3383/727/3384/7448/6404/730/653509/722/2162/718/22918/2159/5199/5
## WP2328                                    732/7431/729/5243/959/727/6387/730/718/6366/185/714/717/71
## WP545
## WP558
## WP5090                                                  732/729/23250/2219/727/7448/730/57088/722/84612/7
##        Count
## WP2806    38
## WP2328    32
## WP545     15
## WP558     21
## WP5090    30
```

The WP Complement system (top1) is linked to immune system and, along with coagulation (top4), is also present in the GO enrichment analysis for down-regulated genes.

For the sake of curiosity, we have also performed enriched KEGG pathways resulting from both up- and down-regulated gene lists.

```
upkegg <- enrichKEGG(gene = upreg_df_entrez$entrezgene_id,
                     organism = 'human',
                     keyType = "kegg",
                     pAdjustMethod = "BH",
                     minGSSize = 10,
                     maxGSSize = 500,
                     pvalueCutoff = 0.05,
                     qvalueCutoff = 0.05,
                     use_internal_data = FALSE)
head(upkegg, n=5)
```

```
##                ID                                     Description GeneRatio
## hsa04110 hsa04110                                      Cell cycle    48/699
## hsa00980 hsa00980 Metabolism of xenobiotics by cytochrome P450    21/699
## hsa04115 hsa04115                          p53 signaling pathway    20/699
## hsa00040 hsa00040     Pentose and glucuronate interconversions    13/699
## hsa03030 hsa03030                                 DNA replication    13/699
##           BgRatio       pvalue     p.adjust       qvalue
## hsa04110 157/8659 1.080871e-16 3.415551e-14 3.037815e-14
## hsa00980  78/8659 5.503321e-07 8.695247e-05 7.733614e-05
## hsa04115  74/8659 9.572793e-07 1.008334e-04 8.968196e-05
## hsa00040  36/8659 2.230885e-06 1.409919e-04 1.253992e-04
## hsa03030  36/8659 2.230885e-06 1.409919e-04 1.253992e-04
##
## hsa04110 10926/1870/1663/9319/4171/4175/10403/4998/23594/8318/990/8317/4174/1869/4173/898/7272/991/9
## hsa00980
## hsa04115
## hsa00040
## hsa03030
##          Count
## hsa04110    48
## hsa00980    21
## hsa04115    20
```

```
## hsa00040    13
## hsa03030    13
```

```
downkegg <- enrichKEGG(gene = downreg_df_entrez$entrezgene_id,
                       organism = 'human',
                       keyType = "kegg",
                       pAdjustMethod = "BH",
                       minGSSize = 10,
                       maxGSSize = 500,
                       pvalueCutoff = 0.05,
                       qvalueCutoff = 0.05,
                       use_internal_data = FALSE)
head(downkegg, n=5)
```

```
##                ID                          Description GeneRatio  BgRatio
## hsa04610 hsa04610 Complement and coagulation cascades    34/736  86/8659
## hsa05150 hsa05150     Staphylococcus aureus infection    33/736  96/8659
## hsa04514 hsa04514             Cell adhesion molecules    42/736 158/8659
## hsa04640 hsa04640          Hematopoietic cell lineage    31/736  99/8659
## hsa05310 hsa05310                              Asthma    15/736  31/8659
##                pvalue     p.adjust       qvalue
## hsa04610 2.738630e-15 8.380206e-13 6.284434e-13
## hsa05150 7.664029e-13 1.172596e-10 8.793464e-11
## hsa04514 1.018900e-11 1.039278e-09 7.793689e-10
## hsa04640 6.032558e-11 4.614907e-09 3.460783e-09
## hsa05310 6.269317e-09 3.836822e-07 2.877287e-07
##
## hsa04610                                                                                                7035/732/729/1361/2160/305
## hsa05150                                                                                               3683/3383/727/6404/718/5648/
## hsa04514 3683/6401/920/933/51208/8516/5788/6614/3383/959/64115/3384/6404/3680/90952/58494/22854/9075/
## hsa04640                                                                                                      920/933/3568/951/94
## hsa05310
##          Count
## hsa04610    34
## hsa05150    33
## hsa04514    42
## hsa04640    31
## hsa05310    15
```

KEGG enrichment looks more consistent for the up-regulated genes than for the down-regulated genes, given the presence of many heterogeneous diseases, not necessarily related to cancer.

5. Visualize an enriched pathway from the up-regulated gene list

In order to use the function *pathview()*, we have to create a vector, containing logFC values related to the corresponding Entrez gene IDs, belonging to the up-regulated gene subset. Since the function only accepts KEGG enrichment pathway results as input, we have chosen the top1 Cell cycle pathway (*ID: hsa04110*), which is also present in WP pathway results, ranking second.

```
logFC <- upreg_df_entrez$logFC
names(logFC) <- upreg_df_entrez$entrezgene_id

pT <- pathview(gene.data = logFC,
         pathway.id = 'hsa04110',
```

```
        species = "human",
        kegg.native = T,
        same.layer = F)

head(pT$plot.data.gene,5)
```

```
##   kegg.names labels               all.mapped type   x   y width height
## 4       1029 CDKN2A                    1029 gene 532 218    46     17
## 5      51343   FZR1                         gene 981 630    46     17
## 6       4171   MCM2 4171,4173,4174,4175,4176 gene 553 681    46     17
## 7       4998   ORC1               4998,23594 gene 494 681    46     17
## 8        996  CDC27                         gene 981 392    46     17
##    mol.data mol.col
## 4  4.502265 #FF0000
## 5        NA #FFFFFF
## 6 11.765436 #FF0000
## 7  7.755551 #FF0000
## 8        NA #FFFFFF
```

```
str(pT)
```

```
## List of 2
##  $ plot.data.gene:'data.frame':  112 obs. of  10 variables:
##   ..$ kegg.names: chr [1:112] "1029" "51343" "4171" "4998" ...
##   ..$ labels    : chr [1:112] "CDKN2A" "FZR1" "MCM2" "ORC1" ...
##   ..$ all.mapped: chr [1:112] "1029" "" "4171,4173,4174,4175,4176" "4998,23594" ...
##   ..$ type      : chr [1:112] "gene" "gene" "gene" "gene" ...
##   ..$ x         : num [1:112] 532 981 553 494 981 981 188 432 780 873 ...
##   ..$ y         : num [1:112] 218 630 681 681 392 613 613 285 562 392 ...
##   ..$ width     : num [1:112] 46 46 46 46 46 46 46 46 46 46 ...
##   ..$ height    : num [1:112] 17 17 17 17 17 17 17 17 17 17 ...
##   ..$ mol.data  : num [1:112] 4.5 NA 11.77 7.76 NA ...
##   ..$ mol.col   : chr [1:112] "#FF0000" "#FFFFFF" "#FF0000" "#FF0000" ...
##  $ plot.data.cpd : NULL
```

```
pF <- pathview(gene.data = logFC,
        pathway.id = 'hsa04110',
        species = "human",
        kegg.native = F,
        same.layer = F
)
```

```
##      [,1] [,2]
## [1,] "9"  "300"
## [2,] "9"  "306"
```

```
head(pF$plot.data.gene,5)
```

```
##   kegg.names labels               all.mapped type   x   y width height
## 4       1029 CDKN2A                    1029 gene 532 218    46     17
```

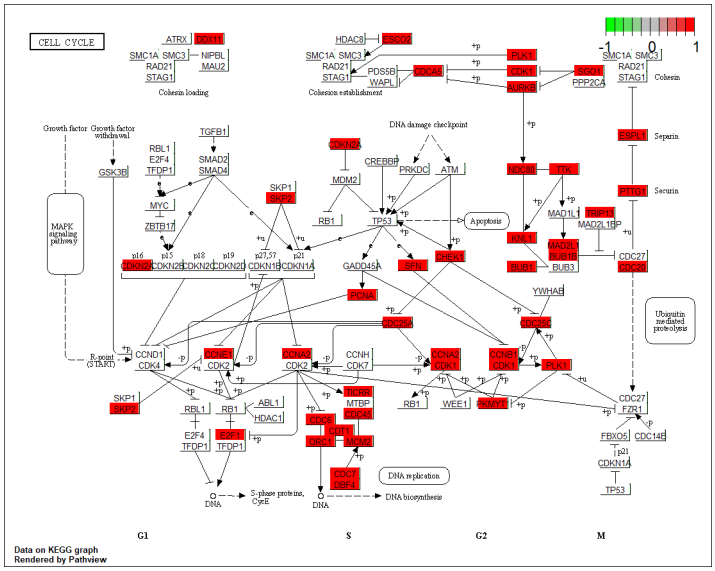


Figure 1: pathview() output for KEGG native results.

```
## 5      51343   FZR1                          gene 981 630    46     17
## 6       4171   MCM2 4171,4173,4174,4175,4176 gene 553 681    46     17
## 7       4998   ORC1               4998,23594 gene 494 681    46     17
## 8        996  CDC27                          gene 981 392    46     17
##    mol.data mol.col
## 4  4.502265 #FF0000
## 5        NA #FFFFFF
## 6 11.765436 #FF0000
## 7  7.755551 #FF0000
## 8        NA #FFFFFF
```

```
str(pF)
```

```
## List of 2
##  $ plot.data.gene:'data.frame':  112 obs. of  10 variables:
##   ..$ kegg.names: chr [1:112] "1029" "51343" "4171" "4998" ...
##   ..$ labels    : chr [1:112] "CDKN2A" "FZR1" "MCM2" "ORC1" ...
##   ..$ all.mapped: chr [1:112] "1029" "" "4171,4173,4174,4175,4176" "4998,23594" ...
##   ..$ type      : chr [1:112] "gene" "gene" "gene" "gene" ...
##   ..$ x         : num [1:112] 532 981 553 494 981 981 188 432 780 873 ...
##   ..$ y         : num [1:112] 218 630 681 681 392 613 613 285 562 392 ...
##   ..$ width     : num [1:112] 46 46 46 46 46 46 46 46 46 46 ...
##   ..$ height    : num [1:112] 17 17 17 17 17 17 17 17 17 17 ...
##   ..$ mol.data  : num [1:112] 4.5 NA 11.77 7.76 NA ...
##   ..$ mol.col   : chr [1:112] "#FF0000" "#FFFFFF" "#FF0000" "#FF0000" ...
##  $ plot.data.cpd : NULL
```

Depending on the *kegg.native* parameter setting, *pathview()* outputs either two png images of the Cell cycle annotated pathway (corresponding to the *pathway.id* parameter provided by KEGG enrichment) or a pdf image exploiting the Graphviz view, which contains a detailed legend of edges and nodes. The latter visualization helps the reader (biologist) explore the pair-wise relationships between genes. In particular, the

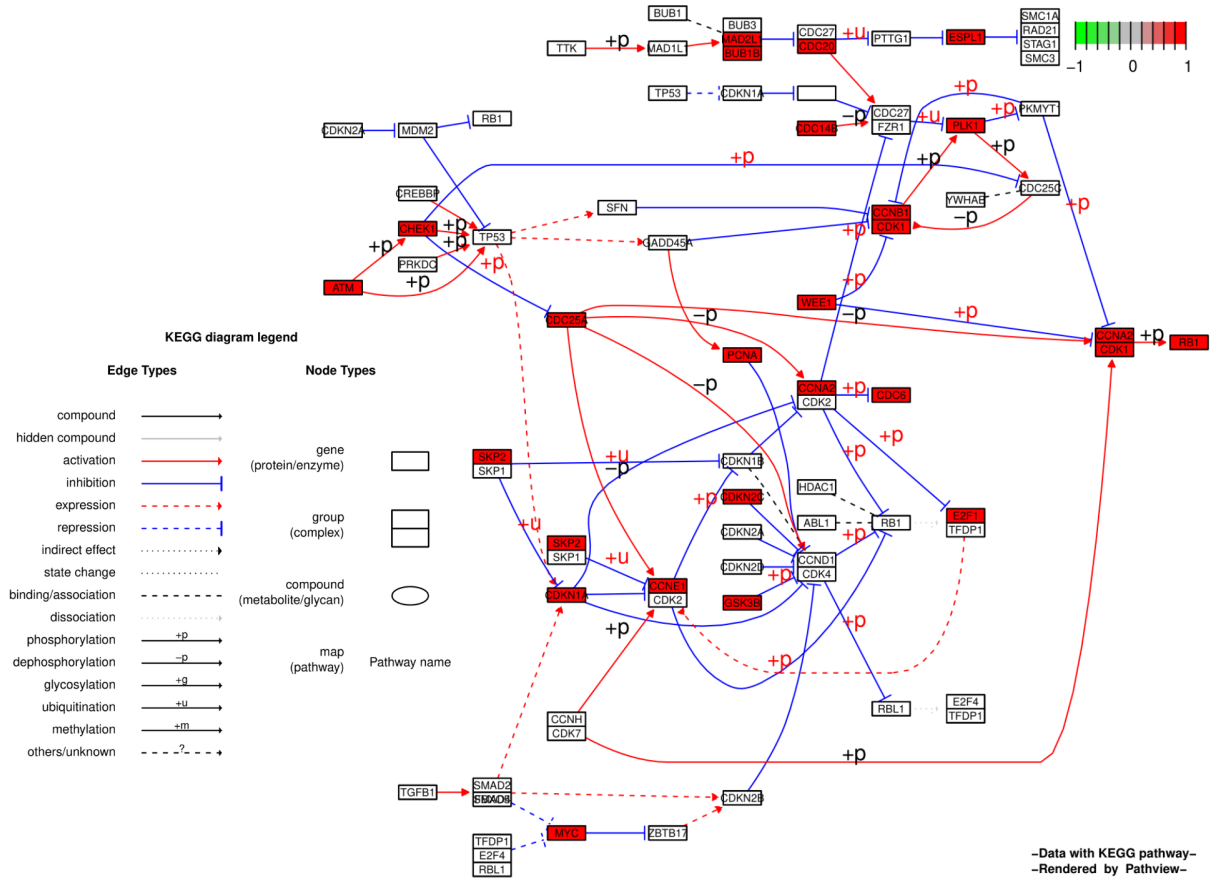


Figure 2: pathview() output for Graphviz (KEGG) results.

red nodes represent the overlapping between the Cell cycle pathway and our subset of up-regulated genes. Indeed, they have a positive score, based on the logFC we have provided.

6. Identify Transcription Factors with enriched scores in the promoters of all up-regulated genes

To proceed in the analysis, we have decided to keep the *upreg_df_entrez* subset, to have as much consistency as possible between gene identifications and for reusability purposes. Then, we have used the *getSequence()* function to extract the promoter sequences (flanking sequences) of all the included up-regulated genes (by ensembl IDs), with a window of 500 nucleotides upstream each gene.

Next, we are going to determine whether the regulatory sequences (e.g., promoters) of the set of up-regulated genes have significantly higher than expected affinity for a regulatory protein or microRNA for which the DNA-binding motif is known (through the PWM matrices provided by *PWMLogn.hg19.MotifDb.Hsap* for each candidate), using the *motifEnrichment()* function (the lognormal affinity background distribution is applied). Such a motif will be considered "enriched" in the set of sequences. The regulatory proteins whose motifs are most enriched in the set of regulatory sequences (of the up-regulated genes) are candidate transcriptional regulators for some or all of the genes and this result is calculated by the *groupReport()* function. In particular, by setting *by.top.motifs* = *T*, we ranks motifs by breadth of enrichment, which is calculated by comparing enrichment between motifs in individual sequences. We then select the first 5 (supposed) Transcription Factors corresponding to the top enriched motifs, trough the plot function, using a p-value of 0.05 as a threshold on the *groupReport()* output (*report* variable).

```
ensembl <-useMart(biomart = 'ensembl', dataset = 'hsapiens_gene_ensembl')
promoter_seq <- getSequence(id = upreg_df_entrez$ensembl_gene_id,
                            type = 'ensembl_gene_id',
                            seqType = 'gene_flank',
                            upstream = 500,
                            mart = ensembl
)

# Transform each gene flank into a DNA String object for efficient storage
# and manipulation of long DNA sequences
sequences <- lapply(promoter_seq$gene_flank, function(x) DNAString(x))

data("PWMLogn.hg19.MotifDb.Hsap")
enriched_TFs <- motifEnrichment(sequences, PWMLogn.hg19.MotifDb.Hsap, score = "affinity")
```

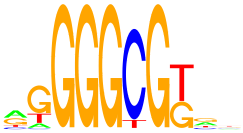
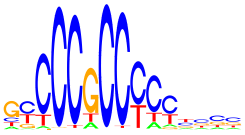
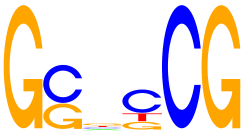
```
## Calculating motif enrichment scores ...
```

```
report <- groupReport(enriched_TFs, by.top.motifs = T)
report[report$p.value < 0.05]
```

```
## An object of class 'MotifEnrichmentReport':
##      rank  target                          id        raw.score
## 1       1   PGAM2                       PGAM2 7.78068187009726
## 2       2    JUND                  M4506_1.02 7.15261882442738
## 3       3     SP2                         SP2 110.520700993076
## 4       4     NNT                         NNT 2.02961493166642
## 5     5.5 BHLHA15                  M5304_1.02 2.62582181097686
## 6     5.5    ODC1                        ODC1 2.60985137861985
## 7       7     SP1                         SP1 31.2288241681127
## 8       8    ATF4                  M5292_1.02 2.52807117487391
## 9       9    POLI                        POLI 1.26875056538632
```

```
## 10     10   CEBPB                 M4556_1.02 2.39516746856572
## ...   ...     ...                        ...               ...
## 2030 2026   NR2F1 Hsapiens-jolma2013-NR2F1-4 1.45776819229328
##                    p.value     top.motif.prop
## 1    1.75991883548339e-126  0.196078431372549
## 2    1.62952159705964e-120  0.195402298850575
## 3     3.7423876760892e-117  0.187964841108857
## 4    2.19504828245645e-126  0.187288708586883
## 5    9.46089990413147e-113  0.181879648411089
## 6    6.66160103396835e-120  0.181879648411089
## 7     2.14637690922085e-97  0.181203515889114
## 8    1.97335070022343e-113  0.178498985801217
## 9    7.97200278970552e-112  0.177822853279243
## 10    1.7403306524467e-123  0.177146720757268
## ...                    ...                ...
## 2030   0.00619397658734275 0.0223123732251521
```

```
plot(report[1:5], fontsize = 7, id.fontsize=5)
```

Rank	Target	PWM	Motif ID	Raw score	P−value	In top motifs
1	PGAM2		PGAM2	7.78	1.76e−126	20 %
2	JUND		M4506_1.02	7.15	1.63e−120	20 %
3	SP2		SP2	111	3.74e−117	19 %
4	NNT		NNT	2.03	2.2e−126	19 %
5.5	BHLHA15		M5304_1.02	2.63	9.46e−113	18 %

7.Select one top enriched TF, compute the empirical distributions of scores for all the matching MOtifDB PWMs to find the 99.75% threshold cutoff.

Now that we have identified which supposed TFs have the top enriched scores in the promoters of some (or all) up-regulated genes, we have selected the top2 enriched gene, JUND, which encodes for the transcription factor JunD, a member of the JUN family and a functional component of the AP1 transcription factor complex. It is known to be a proto-oncogene and has been proposed to protect cells from p53-dependent

senescence and apoptosis. Additionally, JunD, in absence of c-Jun was found to be involved in Ras-induced lung cancer (Ruiz et al., 2021, 10.1172/jci.insight.124985).

We have ruled out PGAM2 (top1 enriched motif), since it encodes for a muscle-specific Phosphoglycerate Mutase, so it is a protein but not a TF (genecards.org & https://www.ncbi.nlm.nih.gov).

```
# select JUND (TF)
TF <- report$target[2]

# extract all PWMs in MotifDB that match the selected TF
TF_motifs <- subset(MotifDb, organism == 'Hsapiens' & geneSymbol == TF)
PWM = toPWM(as.list(TF_motifs))
names(PWM) = sapply(names(PWM),function(x) strsplit(x,"-")[[1]][3])
```

The function *motifEcdf()* computes the empirical cumulative distribution function of the scores (for each motif/PWM), that are obtained by *motifScores()* applied to the PWMs (18) against all the promoters of the human genome, that have different lengths, to find overall affinity.

```
# calculate empirical score distributions for the JUND motifs
# (compute the empirical distributions of scores for all the detected PWMs)
# against all genome promoters (of different lengths)
ecdf <- motifEcdf(PWM,organism = "hg19",quick=TRUE)

# calculate empirical score distribution for the JUND motifs against our set of promoters (background)
# if raw.scores=TRUE, returns a list of raw score values (before cutoff)
scores <- motifScores(sequences, PWM, raw.scores = T, verbose=T)

# compute threshold cutoff on the empirical score distributions for all the PWS
# (against all genome promoters)
threshold <- lapply(ecdf, function(x) quantile(x, 0.9975))
```

8. Identify which up-regulated genes have a region in their promoter with binding scores for JUND above the computed threshold

```
# calculate empirical score distribution for the JUND motifs against our set of promoters (background)
#  if raw.scores=FALSE and cutoff=threshold(0.9975), returns a list of score values after cutoff
scores = motifScores(sequences, PWM, raw.scores = FALSE, cutoff = unlist(threshold))
scores_sign <- which(apply(scores,1,sum) > 0)
# subset of up-regulated genes with score above threshold
enriched_jund <- upreg_df_entrez[scores_sign, 2]
length(enriched_jund)
```

```
## [1] 1345
```

Since the number of up-regulated genes with binding affinity for JUND is 1345 over 1492 (the total number of up-regulated genes), about the 90%. This result is not informative. Hence we have decided to raise the threshold to 99.9%.

```
threshold_update <- lapply(ecdf, function(x) quantile(x, 0.999))
scores_update = motifScores(sequences, PWM, raw.scores = FALSE, cutoff = unlist(threshold_update))
scores_sign_update <- which(apply(scores_update,1,sum) > 0)
# subset of up-regulated genes with score above threshold
enriched_jund_update <- upreg_df_entrez[scores_sign_update, 2]
length(enriched_jund_update)
```

```
## [1] 579
```

The number of up-regulated genes with binding affinity for JUND has now become 579 over 1492, about 39%, which seems more reasonable.

9. Find PPI interactions among differentially expressed genes and export the network in TSV format

```
# Resume up and down-regulated gene subsets, ordering by p-value
upreg_STRING <- upreg_df[order(upreg_df$PValue),]
downreg_STRING <- downreg_df[order(downreg_df$PValue),]
# Merge up and down-regulated gene subsets to get a list of all the DE genes (ordered by p-value)
all_de_STRING <- rbind(upreg_STRING,downreg_STRING)
all_de_STRING <- all_de_STRING[order(all_de_STRING$PValue),]

# Export the above produced lists for STRING website (cutting the lists for readability purposes)
write(upreg_STRING$external_gene_name[1:100],"100_upreg_gene_names.txt")
write(downreg_STRING$external_gene_name[1:100], "100_downreg_gene_names.txt")
write(all_de_STRING$external_gene_name[1:200], "200_DE_gene_names.txt")
#STRING maximum input of genes
write(all_de_STRING$external_gene_name[1:2000], "2000_DE_gene_names.txt")
write(all_de_STRING$external_gene_name, "all_DE_gene_names.txt")
```

We are supposed to upload the generated *txt* files on the STRING web interface. Alternatively, we can connect to STRING APIs through the package *rbioapi*, avoiding the intermediate steps of *txt* files' production and use of the web interface.

First we get the PNG images of the PPI interactions' networks through the function *rba_string_network_image*, to have a qualitative assessment of the results.

```
# PNG images of the PPI interactions output networks from STRING corresponding to
# up-regulated (first 100), down-regulated (first 100), all DE (first 200 and 2000) lists.
# We chose to cut the lists for readability purposes and STRINGdb constraints.

PPI_graph_up <- rba_string_network_image(ids=upreg_STRING$external_gene_name[1:100],
                image_format = "image", species = 9606, network_flavor = "evidence")
```

```
PPI_graph_down <- rba_string_network_image(ids=downreg_STRING$external_gene_name[1:100],
                  image_format = "image", species = 9606, network_flavor = "evidence")
```

```
PPI_graph_DE_200 <- rba_string_network_image(ids=all_de_STRING$external_gene_name[1:200],
                    image_format = "image", species = 9606, network_flavor = "evidence")
```

```
PPI_graph_DE_2000 <- rba_string_network_image(ids=all_de_STRING$external_gene_name[1:2000],
                     image_format = "image", species = 9606, network_flavor = "evidence")
```

10. Import the network in R and use *igraph* to identify and plot the largest connected component

If we used the web interface to get the PPI interactions' networks, we would have to export the corresponding TSV files and then reload them into RStudio. Instead, we have decided to employ the function

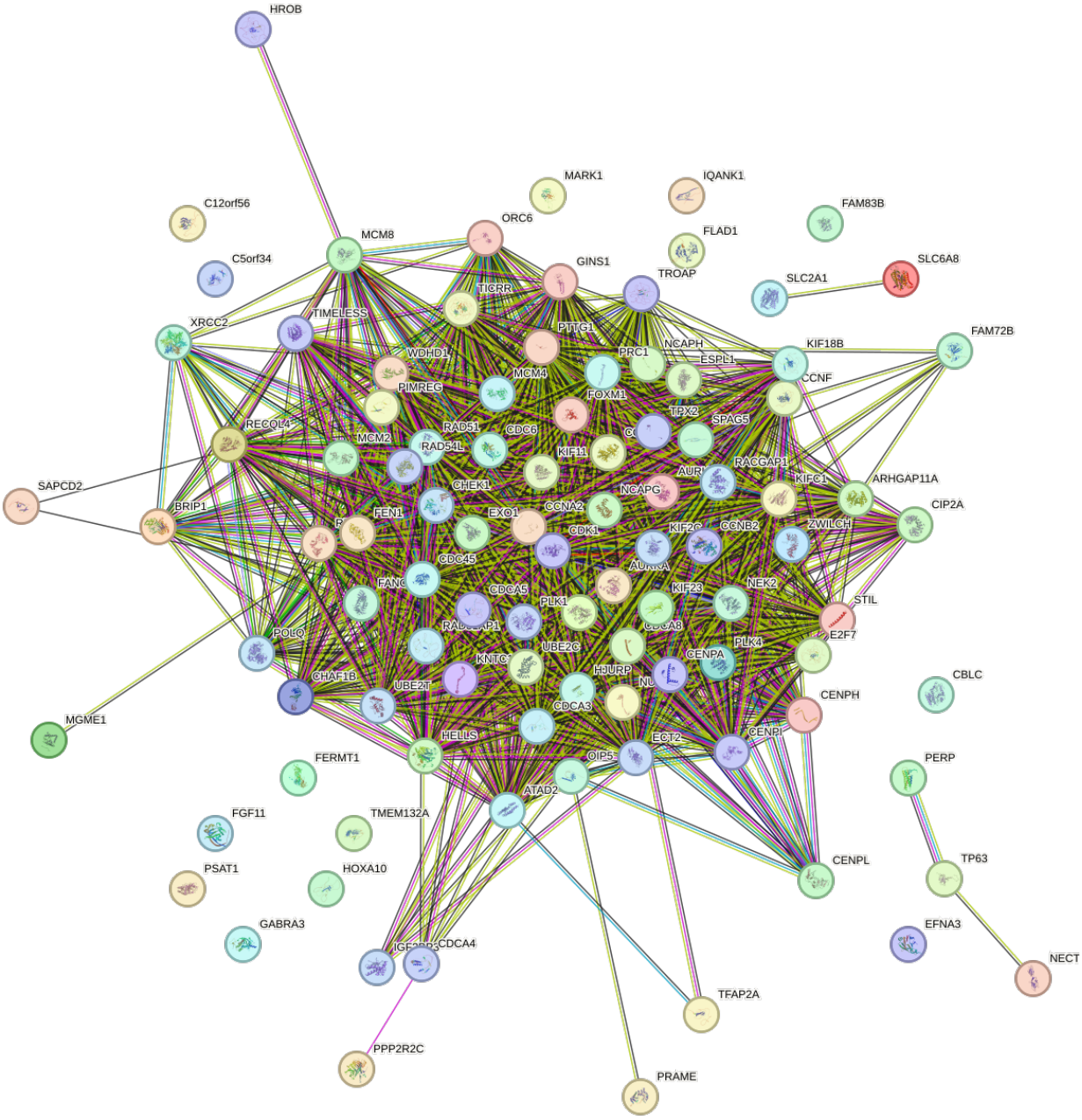


Figure 3: STRING PPI interactions' network for the first 100 up-regulated genes.

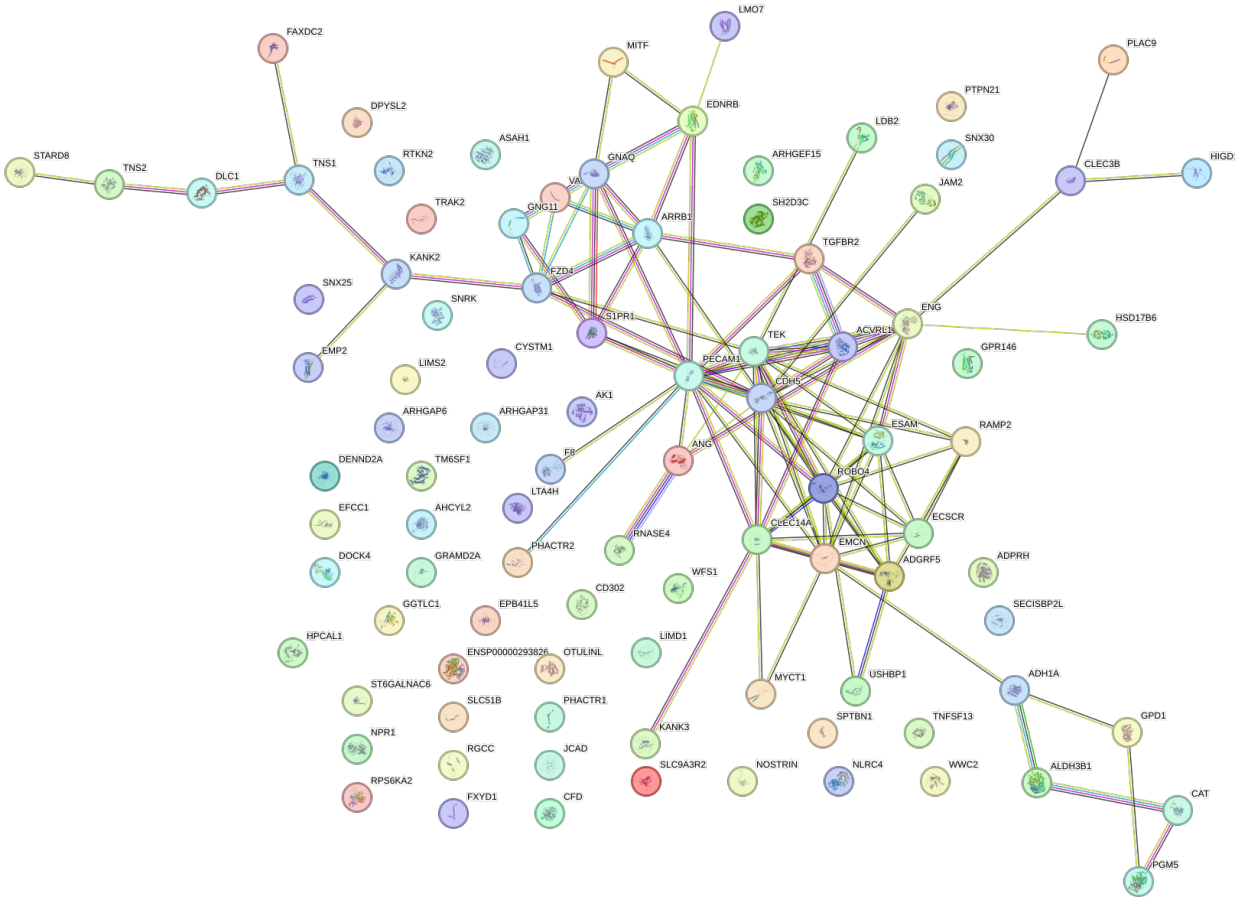


Figure 4: STRING PPI interactions' network for the first 100 down-regulated genes.

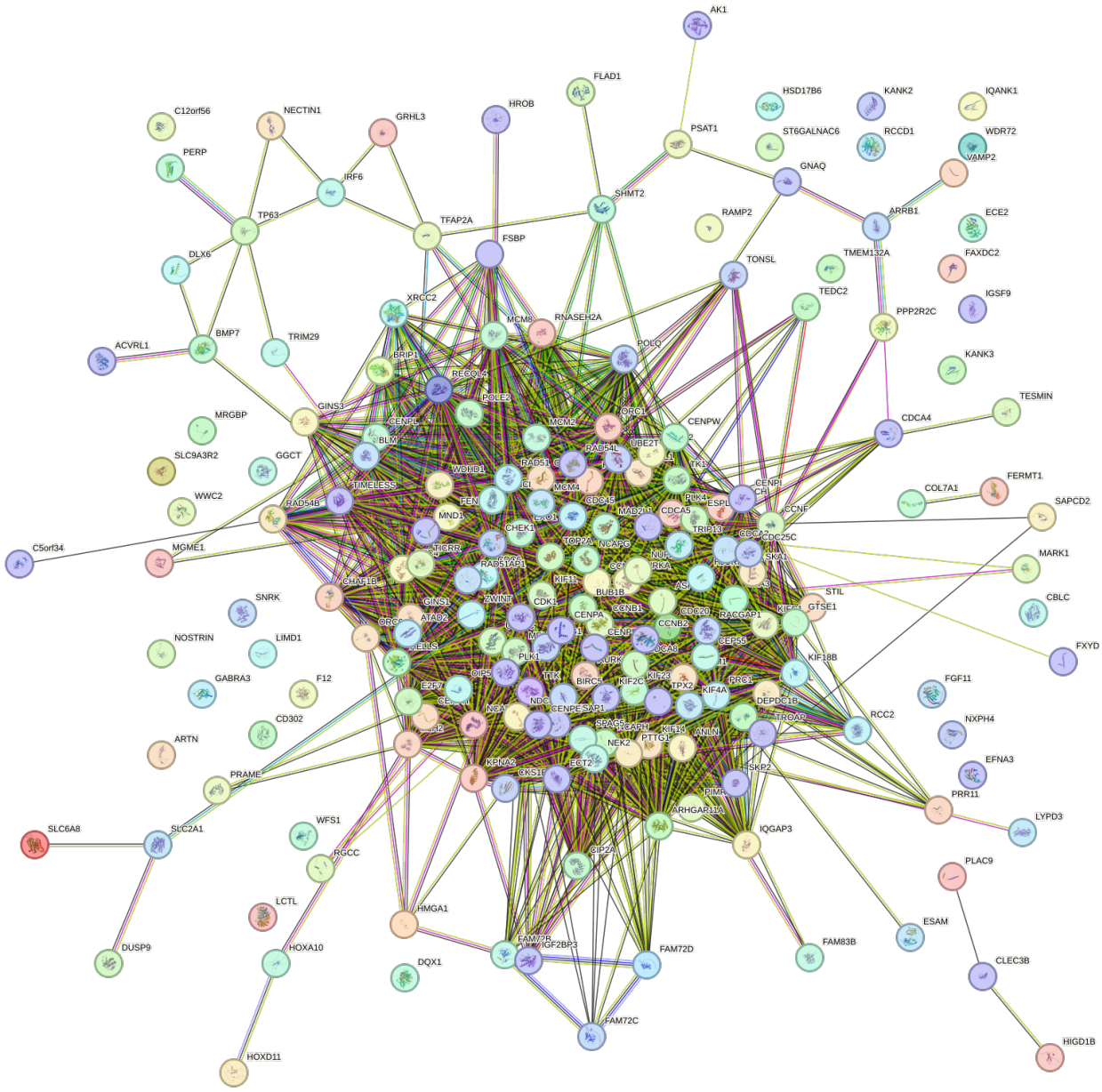


Figure 5: STRING PPI interactions' network for the first 200 up and down-regulated genes (according to p-value).

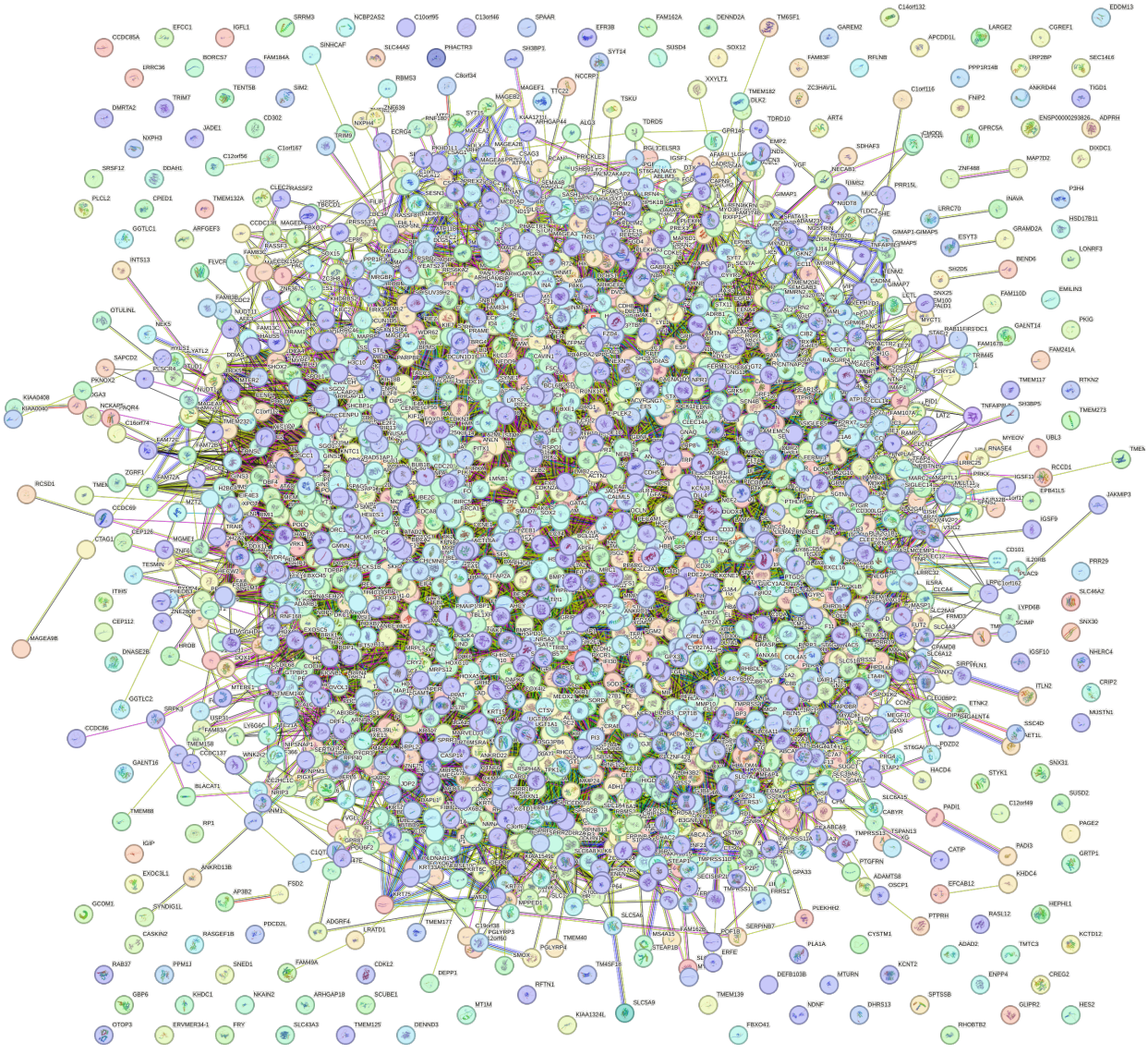


Figure 6: STRING PPI interactions' network for the first 2000 up and down-regulated genes (ordered by p-value).

rba_string_interactions_network() from *rbioapi* to get the PPI interactions network information as it is contained in the TSV files (exportable from the STRING web interface). We are going to use both methods for the reduced up-regulated genes' subset to verify that their results are consistent (the same). If so, we will perform the LCC (Largest Connected Component) identification only with the *rbioapi* package, since it is quicker.

```
# Alternative to tsv file upload (rbioapi)
PPI_net_up <- rba_string_interactions_network(ids=upreg_STRING$external_gene_name[1:100],
              species=9606, required_score = 400)
# function called in date: 13/01/2024 at 11:06

# Deleting duplicate rows (for undirected edges)
PPI_net_up <- distinct(PPI_net_up)
# formatting output to match igraph constraints
PPI_net_up <- PPI_net_up[, -c(1,2)]

# tsv file upload (for consistency check)
PPI_tsv_up <-  read.delim("100_upreg_STRING_PPI_interactions.tsv")
# file downloaded in date: 13/01/2024 at 11:06

# PPI network comparison

# ordering gene pairs (rbioapi)
PPI_net_up <- transform(PPI_net_up, preferredName_A = pmin(preferredName_A, preferredName_B),
                        preferredName_B=pmax(preferredName_A, preferredName_B))
# ordering alphabetically by first column
PPI_net_up <- PPI_net_up[order(PPI_net_up$preferredName_A),]
# resetting row names
rownames(PPI_net_up) <- 1:nrow(PPI_net_up)

# ordering gene pairs (tsv file)
PPI_tsv_up <- transform(PPI_tsv_up, X.node1 = pmin(X.node1, node2), node2=pmax(X.node1, node2))
# ordering alphabetically by first column
PPI_tsv_up <- PPI_tsv_up[order(PPI_tsv_up$X.node1),]
# resetting row names
rownames(PPI_tsv_up) <- 1:nrow(PPI_tsv_up)

identical(PPI_net_up$preferredName_A, PPI_tsv_up$X.node1) # TRUE in date: 13/01/2024 at 11:06
```

```
## [1] TRUE
```

```
identical(PPI_net_up$preferredName_B, PPI_tsv_up$node2) # TRUE in date: 13/01/2024 at 11:06
```

```
## [1] TRUE
```

We have verified that the PPI interactions' network produced by *rbioapi* and STRING web interface are the same (15/01/2024, 11:08), so we can proceed with the analysis using the first tool for all the gene lists.

```
PPI_net_down <- rba_string_interactions_network(ids=downreg_STRING$external_gene_name[1:100],
                species=9606, required_score = 400)
# Deleting duplicate rows (for undirected edges)
PPI_net_down <- distinct(PPI_net_down)
```

```
# formatting output to match igraph constraints
PPI_net_down <- PPI_net_down[, -c(1,2)]

PPI_net_DE_200 <- rba_string_interactions_network(ids=all_de_STRING$external_gene_name[1:200],
                  species=9606, required_score = 400)
PPI_net_DE_200 <- distinct(PPI_net_DE_200)
PPI_net_DE_200 <- PPI_net_DE_200[, -c(1,2)]

PPI_net_DE_2000 <- rba_string_interactions_network(ids=all_de_STRING$external_gene_name[1:2000],
                   species=9606, required_score = 400)
PPI_net_DE_2000 <- distinct(PPI_net_DE_2000)
PPI_net_DE_2000 <- PPI_net_DE_2000[, -c(1,2)]
```

We have designed the *find_largest_conn_comp()* function that returns and plots (as a graph) the LCC, following the steps explained in the code comments.

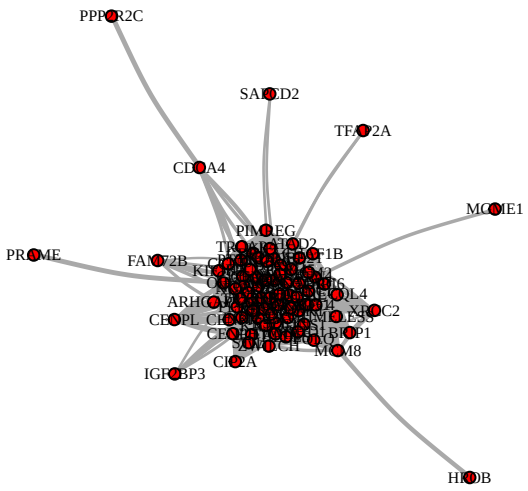
```
find_largest_conn_comp <- function(PPI_net){
  # isolate unique gene names from the node columns (of the PPI interactions network dataframe)
  gene_names <- unique(unlist(PPI_net[, c(1,2)]), use.names = FALSE)
  # get info from dataframe to generate corresponding igraph graph
  graph_from_PPI <- graph_from_data_frame(PPI_net,gene_names,directed=FALSE)
  # check if the graph is connected
  cat("\n Is the graph connected? ", is_connected(graph_from_PPI))
  # count how many (strongly) connected component are there in the graph
  cat("\n How many (strongly) connected components are there? ",
      count_components(graph_from_PPI, mode = "strong"))
  # find the maximal (strongly) connected components of the PPI graph (as clusters)
  conn_comp <- components(graph_from_PPI, mode = "strong")
  # extract largest connected component number (1, associated to largest cluster)
  # by csize parameter (numeric vector giving the sizes of the clusters)
  largest_conn_comp <- which.max(conn_comp$csize)
  cat("\n Largest connected component size: ", conn_comp$csize[largest_conn_comp])
  # create subgraph of the original graph, by specifying vertices (with V() function)
  # having value 1 in the membership parameter of components() output.
  lcc <- induced_subgraph(graph_from_PPI,
         V(graph_from_PPI)[conn_comp$membership == largest_conn_comp])
  # plot lcc subgraph
  plot(lcc,
       # edge thickness based on score values
       edge.width= E(lcc)$score*3,
       vertex.color="red",
       vertex.size=5,
       vertex.label.color="black",
       vertex.label.cex=0.5,
       edge.curved=0.1,
       )

  # Alternative 3D plotting (considered experimental by igraph package authors)
  # rglplot(lcc,
  #       edge.width= E(lcc)$score*3,
  #       vertex.color="red",
  #       vertex.size=5,
  #       vertex.label.color="black",
```

```
  #       vertex.label.cex=0.5,
  #       edge.curved=0.1
  #       # rescale=FALSE
  #       )
  return(lcc)
}
```

We can now use the *find_largest_conn_comp()* function on all four PPI networks, starting from the up-regulated genes.

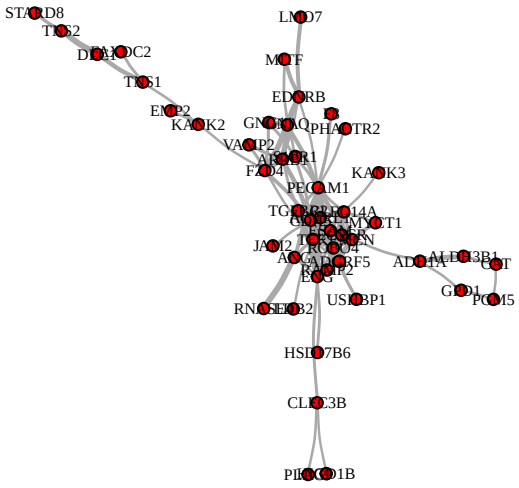
```
#Identify and plot the largest connected component in the up-regulated genes PPI network
lcc_up_100 <- find_largest_conn_comp(PPI_net_up)
```

```
##
##  Is the graph connected?  FALSE
##  How many (strongly) connected components are there?  3
##  Largest connected component size:  80
```



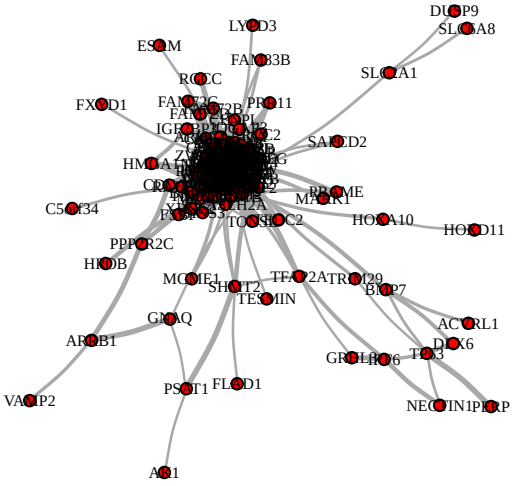

```
# Identify and plot the largest connected component in the down-regulated genes PPI network
lcc_down_100 <- find_largest_conn_comp(PPI_net_down)
```

```
##
##  Is the graph connected?  TRUE
##  How many (strongly) connected components are there?  1
##  Largest connected component size:  47
```



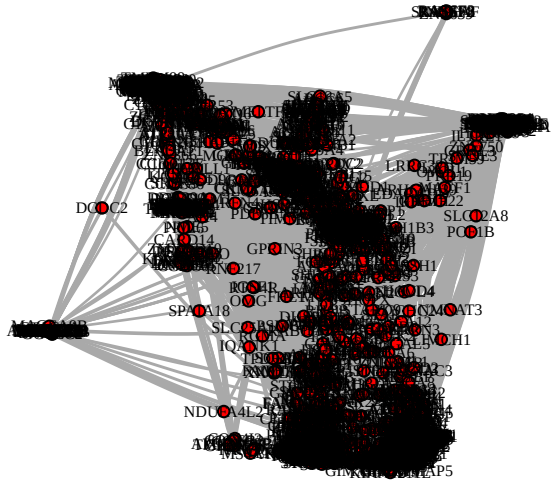

```
# Identify and plot the largest connected component in the top 200 DE genes PPI network
lcc_up_down_200 <- find_largest_conn_comp(PPI_net_DE_200)
```

```
##
##  Is the graph connected?  FALSE
##  How many (strongly) connected components are there?  3
##  Largest connected component size:  159
```




```
#Identify and plot the largest connected component in the top 2000 DE genes PPI network
lcc_up_down_2000 <- find_largest_conn_comp(PPI_net_DE_2000)
```

```
##
##  Is the graph connected?  FALSE
##  How many (strongly) connected components are there?  8
##  Largest connected component size:  1817
```


SLC2A5
DCDC2
SLC2A8
PCDH1B
SPATA18
NDUFA4L2