Project Bioinformatic Resources

Annalisa Xamin, Nicola Perotti

Introduction

The following analysis focus on RNA-seq count data extracted from different cancer datasets from the Cancer Genome Atlas (TCGA). From the original TCGA data 50 cases (tumor samples) and 50 controls (normal samples) were randomly selected.

Analysis

Task 1: Load data

In particular, we consider data coming from **Lung adenocarcinoma**.

```
load("./RData/Lung_adenocarcinoma.RData")
```

After loading the data, the following three data-frames are available:

- `raw_counts_df` which contains the raw RNA-seq counts;
- `c_anno_df`, which contains sample name and condition (case and control);
- `r_anno_df`, which contains the ENSEMBL genes ids, the length of the genes and the genes symbols.

Task 2: Filter protein coding genes

To extract only protein coding genes from `raw_counts_df` and `r_anno_df`, we use the `biomaRt` package.

First, we retrieve the information about the protein coding genes from Ensembl.

```
database <- useMart("ensembl")
datasetHuman <- useDataset("hsapiens_gene_ensembl", mart = database)
query <- getBM(attributes = c("ensembl_gene_id", "external_gene_name", "gene_biotype"),
    filters = c("ensembl_gene_id"), values = r_anno_df$ensembl_gene_id, mart = datasetHuman)

query_protein_coding <- query[which(query$gene_biotype == "protein_coding"), ]
```

Then, we filter the data frames containing the raw counts and the annotation to keep only the protein coding genes.

```
indexes_r_anno_df <- which(r_anno_df$ensembl_gene_id %in% query_protein_coding$ensembl_gene_id)
r_anno_df_protein_coding <- r_anno_df[indexes_r_anno_df, ]

indexes_raw_counts_df <- which(rownames(raw_counts_df) %in% query_protein_coding$ensembl_gene_id)
raw_counts_df_protein_coding <- raw_counts_df[indexes_raw_counts_df, ]
```

Task 3: Differential expression analysis

To perform the differential expression analysis, we will use the `edgeR` package.

It is important to remove genes with low signal that will have low statistical power. Since, we want to focus on the transcripts we can use to perform the analysis, we can filter raw counts data using a threshold of raw count > 20 in at least 5 replicates.

```
# count threshold
count_thr <- 20

# number of replicates with more counts than the count threshold
repl_thr <- 5

filter_vec <- apply(raw_counts_df_protein_coding, 1, function(y) max(by(y, c_anno_df$condition,
    function(x) sum(x >= count_thr))))  # groups the values on each condition and sum all the values ab
```

Then, we filter the previously updated data frames.

```
filter_counts_df <- raw_counts_df_protein_coding[filter_vec >= repl_thr, ]
dim(filter_counts_df)  # 17289
```

```
## [1] 17289   100
```

```
filter_anno_df <- r_anno_df_protein_coding[rownames(filter_counts_df), ]
dim(filter_anno_df)  # 17289
```

```
## [1] 17289     3
```

Now we check the library size of each sample (how many reads we have sequenced for each experiment)

```
size_df <- data.frame(sample = colnames(filter_counts_df), read_millions = colSums(filter_counts_df)/1e

ggplot(data = size_df, aes(sample, read_millions)) + geom_bar(stat = "identity",
    fill = "indianred", colour = "indianred", width = 0.7, alpha = 0.7) + coord_flip() +
    theme_bw()
```

Then, we visualize a boxplot of gene counts.

```
long_counts_df <- gather(as.data.frame(filter_counts_df), key = "sample", value = "read_number")

ggplot(data = long_counts_df, aes(sample, read_number + 1)) + geom_boxplot(colour = "deeppink4",
    fill = "deeppink4", alpha = 0.7) + theme_bw() + scale_y_log10()
```

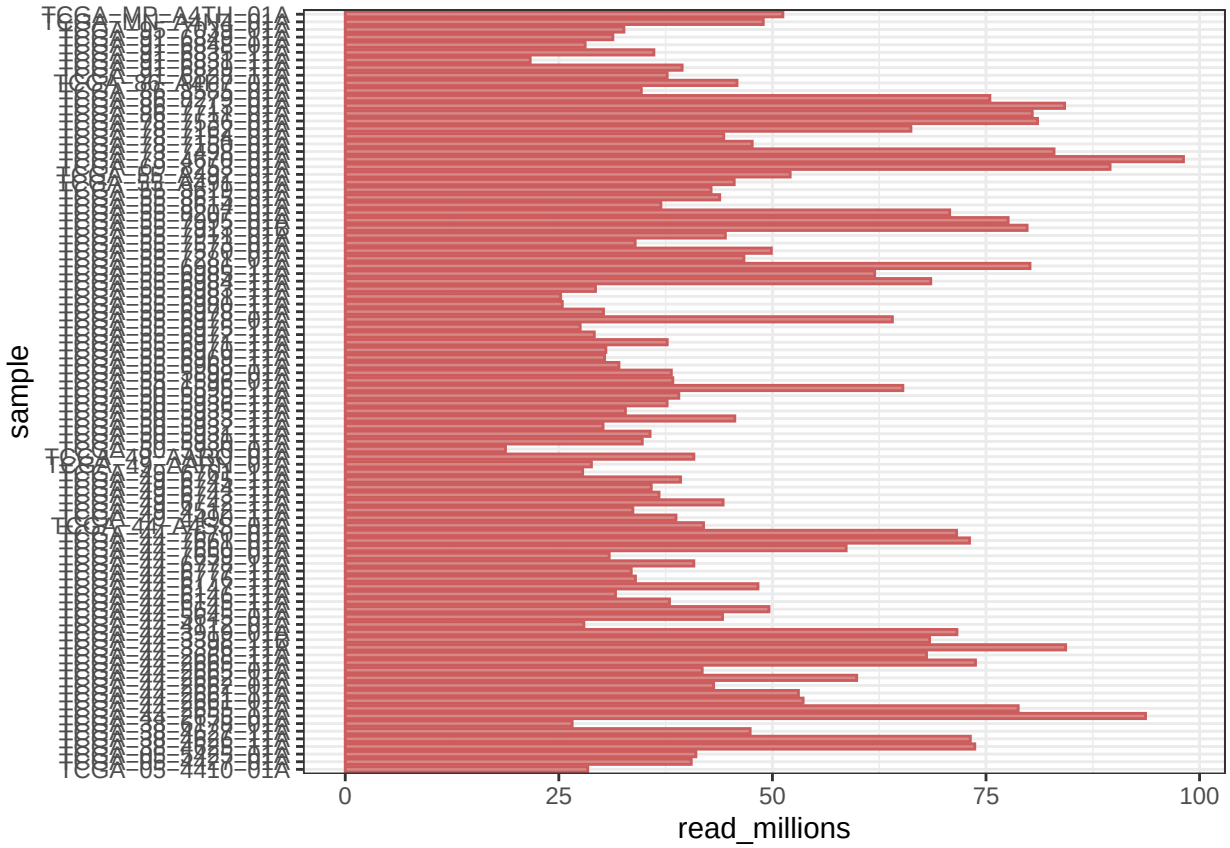


Figure 1: Library size of each sample.

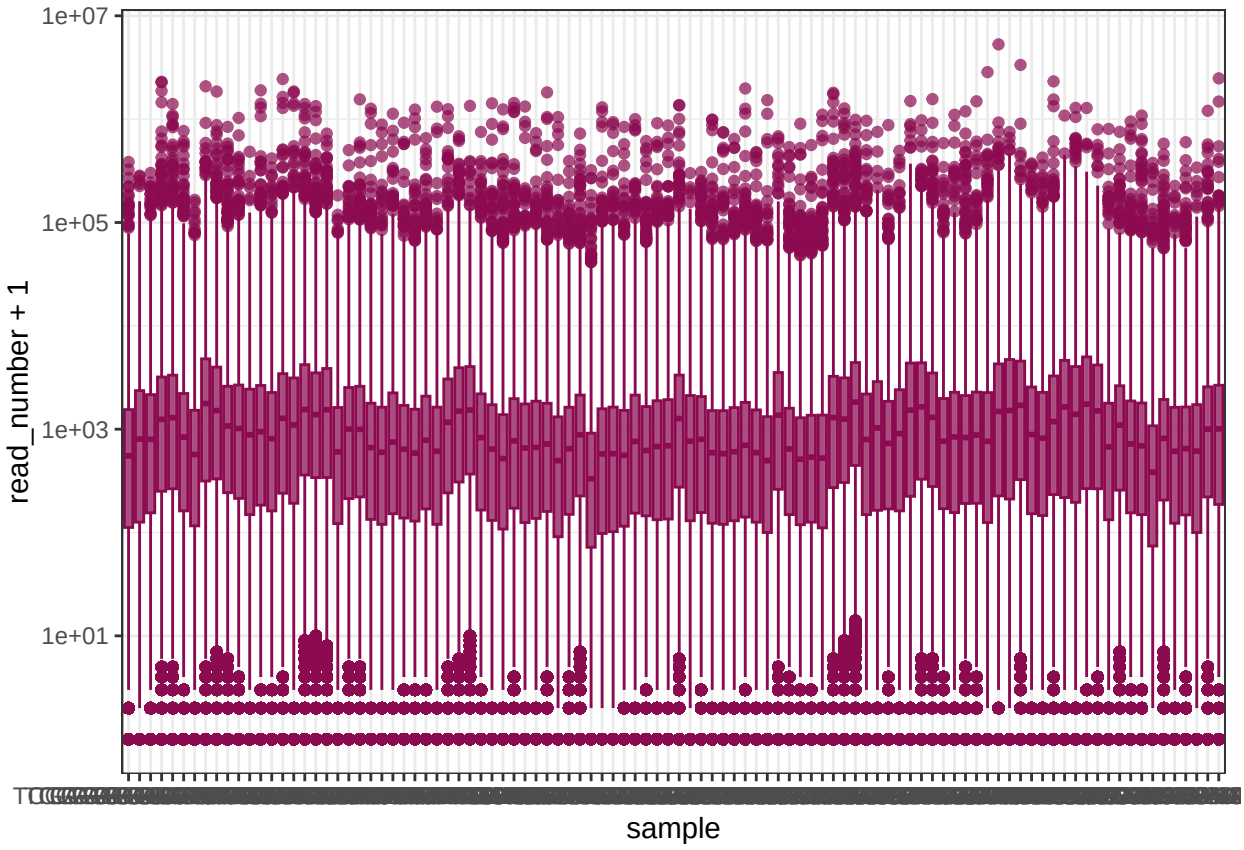


Figure 2: Boxplot of gene counts before normalization.

As we can see from the plots, there is a significant variability across samples in term of library sizes (reads per million). We have to take into account this aspect because the expected size of each count is the product of the relative abundance of that gene in that sample but also of the library size.

As we can see from the boxplot, we need to normalize our data before testing for differential expression. Normalization can be obtained using different methodologies. Among them, TMM (the default method) is a method that consider in the normalization also variables related to the library size.

To perform the DEG analysis, we first need to create a DGRList object containing information about counts, annotation, samples and genes. To do that, we use the function `DGEList` to create the input for the following normalization step. This object contains information about counts, samples and genes.

The normalization intra- and inter-sample is done using the function `calcNormFactors` and specifying `method = "TMM"`, which is the weighted trimmed mean of M-values approach.

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

# computing norm factors for TMM normalization
edge_n <- calcNormFactors(edge_c, method = "TMM")
```

Then, we create a cpm table containing the normalized expression values for each transcript expressed as counts per million (CPM).

```
# create a cpm table (normalized expression values)
cpm_table <- as.data.frame(round(cpm(edge_n), 2))
```

To see the effect of the normalization, we look at the boxplot distribution of gene expression signals after normalization.

```
# look at the boxplot distribution of gene expression signals after
# normalization
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()
```

We notice that, with respect to the previous boxplot, after the normalization, the distributions are comparable. That means that now our data is ready to be tested for DE analysis.

We define the experimental design matrix, later needed to calculate the dispersion and fit. This matrix is based on the experimental design, so we define the conditions we want to test (case VS control).

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

Once we normalized the data and the design, we proceed by calculating the dispersion fit.

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

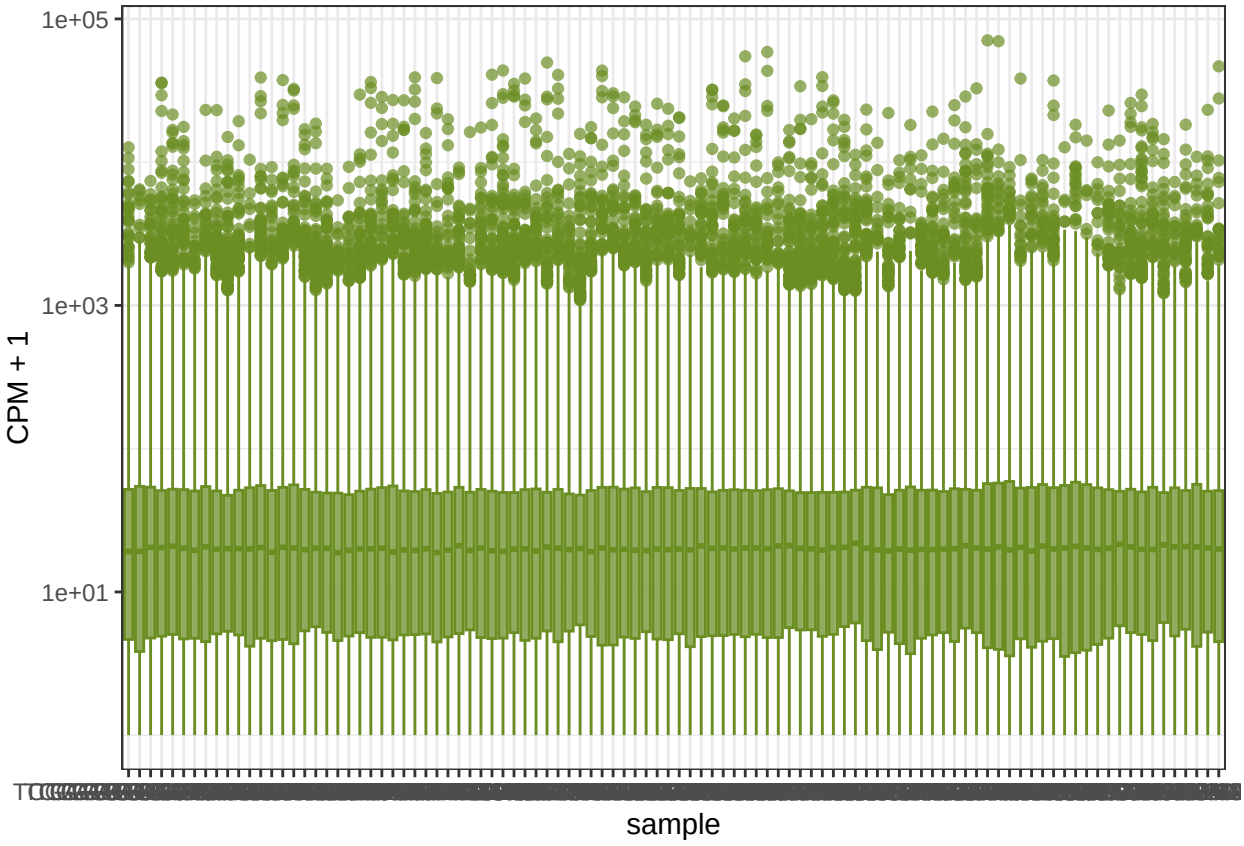


Figure 3: Boxplot distribution of gene expression signals after normalization.

The `estimateDisp` function tries to estimate the variability at different levels: in the data, inter sample and intra sample. The over dispersion of counts across the samples can be modeled as a Poisson distribution, which can be approximated using a negative binomial distribution by using `glmQLFit` function.

We define the contrasts (conditions to be compared).

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

We performed a test through function `glmQLFTest` to determine which genes are differentially expressed. Then, we selected genes based on a 0.01 p-value cutoff and ordered them based on the log2 fold change.

```
# fit the model with generalized linear models
edge_t <- glmQLFTest(edge_f, contrast = contro)
DEGs <- as.data.frame(topTags(edge_t, n = 20000, p.value = 0.01, sort.by = "logFC"))
```

Then, we improve the selection, considering not only the p-value, but also the average expression of the genes (logCPM). We used the logFC value to assign genes to different classes:

- up-regulated genes if $logFC > 1.5$
- down-regulated genes if $logFC < -1.5$
- not significant genes (unchanged) otherwise.

```
DEGs$class <- "="
DEGs$class[which(DEGs$logCPM > 1 & DEGs$logFC > 1.5 & DEGs$FDR < 0.25)] = "+"  # upregulated genes
DEGs$class[which(DEGs$logCPM > 1 & DEGs$logFC < (-1.5) & DEGs$FDR < 0.25)] = "-"  # downregulated genes
DEGs <- DEGs[order(DEGs$logFC, decreasing = T), ]
```

```
head(DEGs)
```

```
##                 ensembl_gene_id external_gene_name length    logFC    logCPM
## ENSG00000179914 ENSG00000179914              ITLN1   8640 6.684901 5.8661078
## ENSG00000108576 ENSG00000108576             SLC6A4  41683 6.084705 6.6338451
## ENSG00000180440 ENSG00000180440             SERTM1  23819 5.663190 3.2093449
## ENSG00000108342 ENSG00000108342               CSF3   2452 5.588520 5.2512424
## ENSG00000034971 ENSG00000034971               MYOC  17271 5.397462 1.5811486
## ENSG00000178084 ENSG00000178084              HTR3C   7626 5.325742 0.5517057
##                        F       PValue          FDR class
## ENSG00000179914  75.57703 6.388727e-14 4.459213e-13     +
## ENSG00000108576 122.16543 3.992071e-19 7.518401e-18     +
## ENSG00000180440 181.47838 2.383659e-24 1.325115e-22     +
## ENSG00000108342  90.91727 8.943402e-16 8.552128e-15     +
## ENSG00000034971 170.30732 1.866103e-23 8.293843e-22     +
## ENSG00000178084 120.55392 5.781068e-19 1.046585e-17     =
```

```
table(DEGs$class)  # 1193-, 877+, 14949=
```

```
##
##    -    +    =
## 1199  884 9258
```

From the results, we saw that 1193 genes are down-regulated, 877 genes are up-regulated and 14949 genes have no changes in their regulation.

We then create the volcano plot.

```
input_df <- DEGs
xlabel <- "log2 FC case vs control"
ylabel <- "-log10 p-value"

par(fig = c(0, 1, 0, 1), mar = c(4, 4, 1, 2), mgp = c(2, 0.75, 0))
plot(input_df$logFC, -log(input_df$PValue, base = 10), xlab = xlabel, ylab = ylabel,
    col = ifelse(input_df$class == "=", "grey70", "olivedrab4"), pch = 20, frame.plot = TRUE,
    cex = 0.8, main = "Volcano plot")
abline(v = 0, lty = 2, col = "grey20")
```

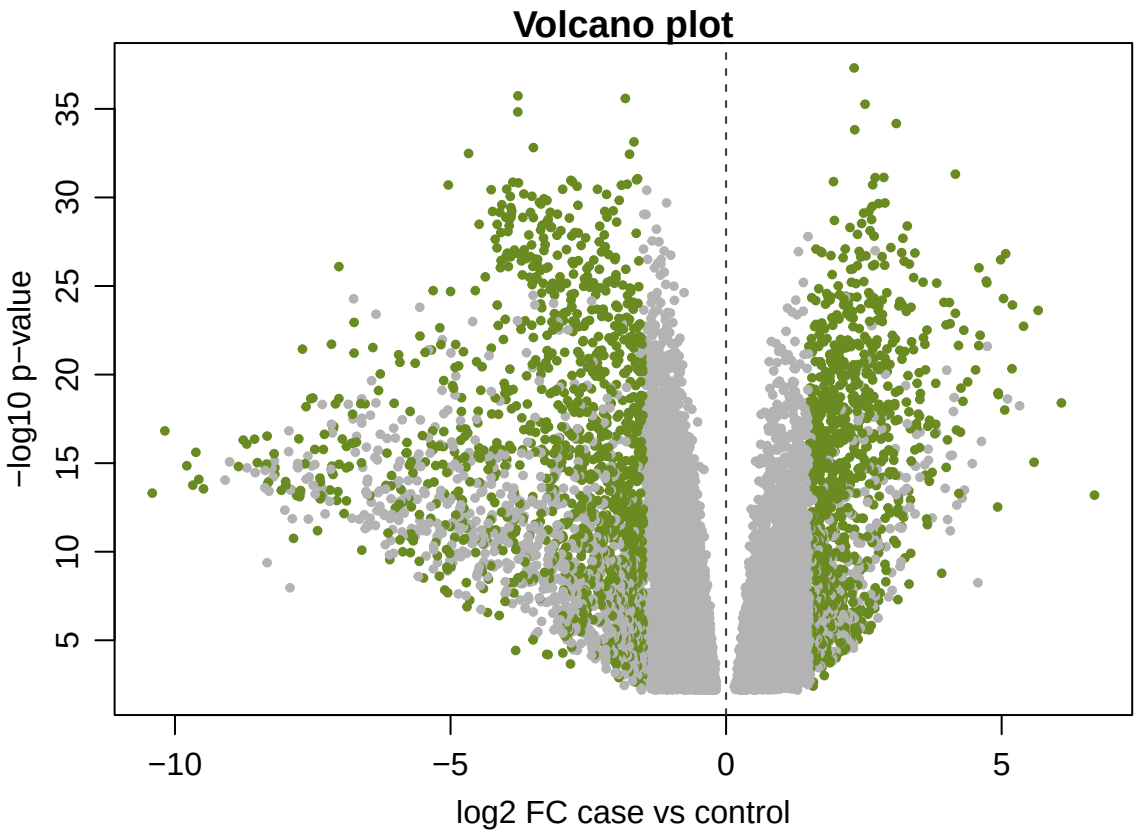


Figure 4: Volcano plot

The volcano plot allows to have a quick visual identification of genes with large fold changes that are also statistically significant.

We also report the heatmap of only up and down-regulated genes.

To create an annotated heatmap focusing only on up- and down-regulated genes we need first of all a matrix in which we select genes with class "+" or "-. Moreover, we also need to assign a color to each sample. In this case, we assign green to the case condition and beige to the control, and save the corresponding color into the variable `cols`. In this way, we can then set the `ColSideColors` parameter in order to have a color code to recognize the sample condition.

```
# plot only up and down regulated genes
cols <- c(ifelse(c_anno_df$condition == "case", "chartreuse4", "burlywood3"))
pal <- c("blue", "white", "red")
pal <- colorRampPalette(pal)(100)
heatmap(as.matrix(cpm_table[which(rownames(cpm_table) %in% DEGs$ensembl_gene_id[which(DEGs$class !=
    "=")]), ]), ColSideColors = cols, cexCol = 0.5, margins = c(4, 4), col = pal,
    cexRow = 0.2)
```

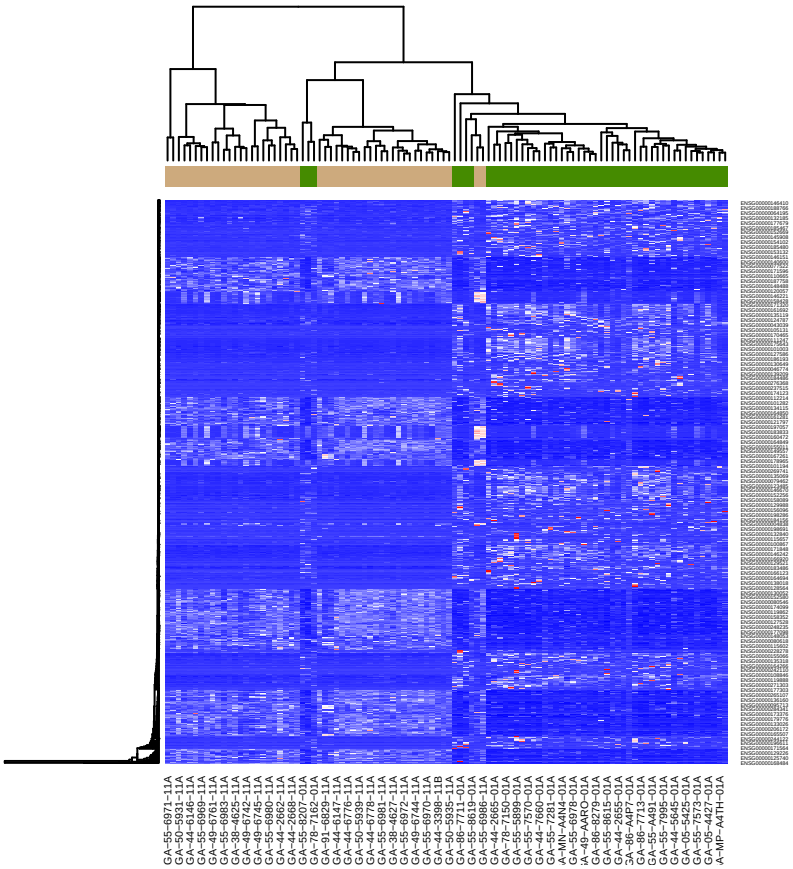


Figure 5: Heatmap up and down-regulated genes.

On the top of the heatmap, we can see the dendrogram indicating how distant our samples are, while on the left the dendrogram related to genes. The dendogram is built by hierarchical clustering, a method based on the concept of dissimilarity.

To simplify the later analysis, we save the list of differentially expressed genes in a text file.

```
up_DEGs <- DEGs[which(DEGs$class == "+"), ]
down_DEGs <- DEGs[which(DEGs$class == "-"), ]

write.table(up_DEGs, file = "output/up_DEGs.txt", row.names = F, col.names = T, sep = "\t",
    quote = F)
write.table(down_DEGs, file = "output/down_DEGs.txt", row.names = F, col.names = T,
    sep = "\t", quote = F)
write.table(DEGs, file = "output/DEGs.txt", row.names = F, col.names = T, sep = "\t",
    quote = F)
```

Task 4: Gene set enrichment analysis

To perform the gene set enrichment analysis, we will use the `clusterProfiler` package.

```
DEGs <- read.table("output/DEGs.txt", header = T, sep = "\t", as.is = T)
table(DEGs$class)
```

```
##
##    -    +    =
## 1199  884 9258
```

We use the `biomaRt` package to retrieve the `entrezgene_id` for all the genes in the `DEGs` dataset.

```
ensembl <- useEnsembl(biomart = "ensembl", dataset = "hsapiens_gene_ensembl")
convert <- getBM(attributes = c("ensembl_gene_id", "entrezgene_id"), filters = c("ensembl_gene_id"),
    values = DEGs$ensembl_gene_id, mart = ensembl)

DEGs <- merge(DEGs, convert, by.x = "ensembl_gene_id", by.y = "ensembl_gene_id")  # include the new inf
DEGs <- DEGs[which(!is.na(DEGs$entrezgene_id)), ]
DEGs <- DEGs[-which(duplicated(DEGs$entrezgene_id)), ]
```

We removed all the NA and duplicates in the dataset DEGs.

We created new lists for the down and up-regulated genes.

```
# list of up-regulated genes
upDEGs <- DEGs %>%
    filter(class == "+")
# list of down-regulated genes
downDEGs <- DEGs %>%
    filter(class == "-")
```

Performing enrichment analysis for GO biological process

```
# biological process up regulated genes
ego_BP_up <- enrichGO(gene = upDEGs$external_gene_name, OrgDb = org.Hs.eg.db, keyType = "SYMBOL",
    ont = "BP", pAdjustMethod = "BH", pvalueCutoff = 0.05, qvalueCutoff = 0.05)
```

```
# biological process down regulated genes
ego_BP_down <- enrichGO(gene = downDEGs$external_gene_name, OrgDb = org.Hs.eg.db,
    keyType = "SYMBOL", ont = "BP", pAdjustMethod = "BH", pvalueCutoff = 0.05, qvalueCutoff = 0.05)
```

We report the top 10 enriched GO terms related to the Biological Process for both up and down regulated genes.
In the barplots we can see that the elements are ordered by adjusted p-value (where the most significant is placed on the top) and on the x-axis we have the gene counts, so the number of elements of our lists were found in the category.

```
barplot(ego_BP_up, showCategory = 10, main = "Up-regulated gene list: top 10 enriched BP terms")
```

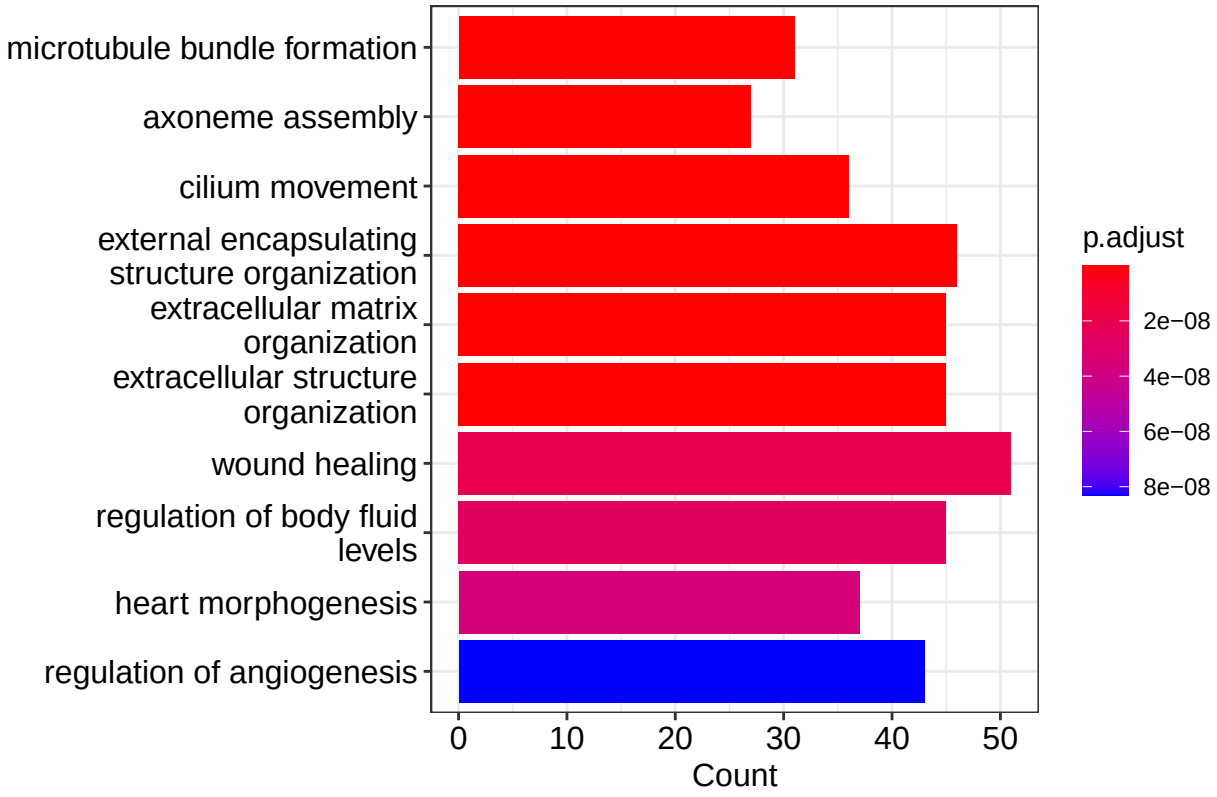


Figure 6: Biological process up regulated genes

Both the first two biological processes are related to the microtubules, which might be explained by the fact that tumour cells have a high growth rate thus the cytoskeleton is assembled over and over again. We also notice that angiogenesis is among the most enriched biological processes as we expected. In fact, angiogenesis is particularly important in tumorigenesis to allow the growth of the tumour tissue, providing nutrients to cells.

```
barplot(ego_BP_down, showCategory = 10, main = "Down-regulated gene list: top 10 enriched BP terms")
```

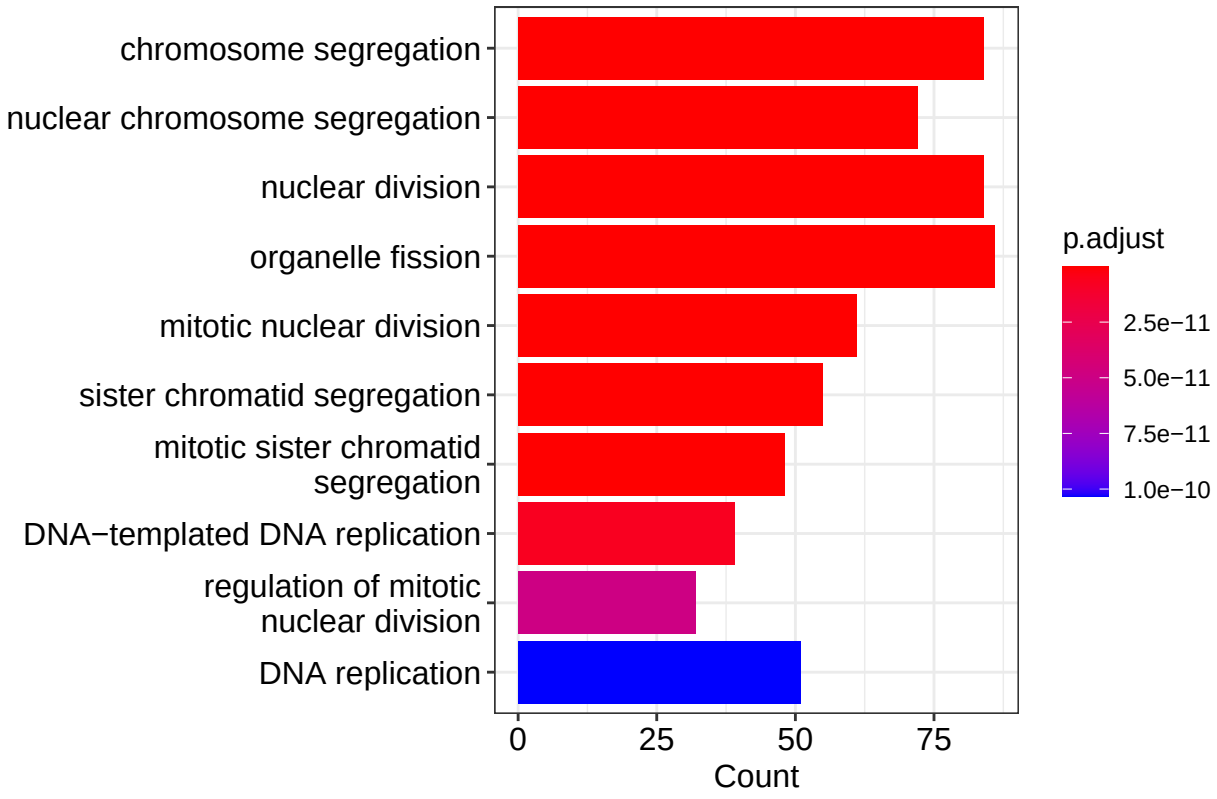


Figure 7: Biological process down regulated genes

We would expect most of these biological processes to be enriched in the up-regulated genes since tumour cells tend to replicate more than normal cells. A possible explaination is that tumour cells genes accumulate numerous mutations usually ending up in loss of function or downregulation. These mutations might occur on transcription factors binding sites or RNA binding sites, reducing their expression.

Performing enrichment analysis for GO molecular function

The same analysis was done also for the molecular function.

```
# molecular function up regulated genes
ego_MF_up <- enrichGO(gene = upDEGs$external_gene_name, OrgDb = org.Hs.eg.db, keyType = "SYMBOL",
    ont = "MF", pAdjustMethod = "BH", pvalueCutoff = 0.05, qvalueCutoff = 0.05)
```

```
# molecular function down regulated genes
ego_MF_down <- enrichGO(gene = downDEGs$external_gene_name, OrgDb = org.Hs.eg.db,
    keyType = "SYMBOL", ont = "MF", pAdjustMethod = "BH", pvalueCutoff = 0.05, qvalueCutoff = 0.05)
```

```
barplot(ego_MF_up, showCategory = 10, main = "Up-regulated gene list: top 10 enriched MP terms")
```

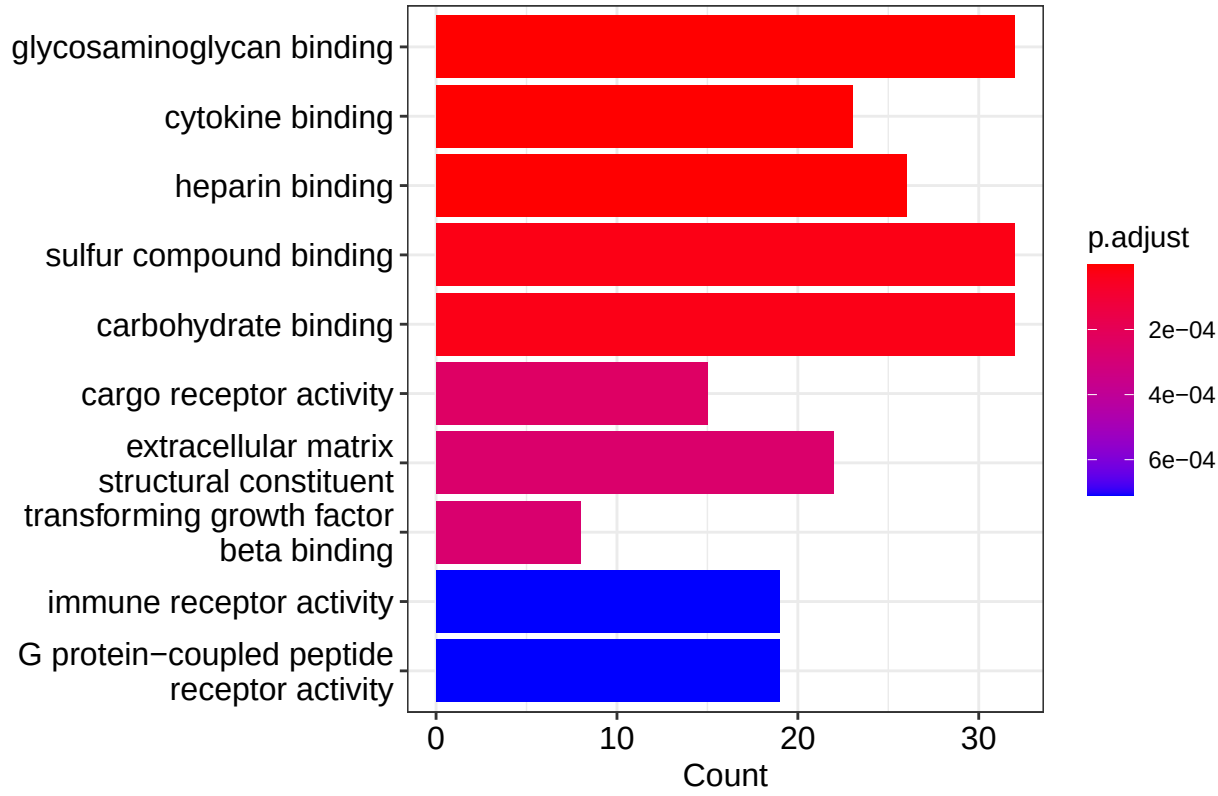


Figure 8: Molecular function up regulated genes

```
barplot(ego_MF_down, showCategory = 10, main = "Down-regulated gene list: top 10 enriched MP terms")
```

```
eWP_up <- enrichWP(gene = upDEGs$entrezgene_id, organism = "Homo sapiens", pvalueCutoff = 0.05,
    qvalueCutoff = 0.1)

head(eWP_up, n = 10)
```

```
##            ID                                                  Description
## WP2806 WP2806                                            Complement system
## WP545   WP545                                         Complement activation
## WP558   WP558                          Complement and coagulation cascades
## WP2431 WP2431                                            Spinal cord injury
## WP5094 WP5094                                        Orexin receptor pathway
## WP5090 WP5090 Complement system in neuronal development and plasticity
```

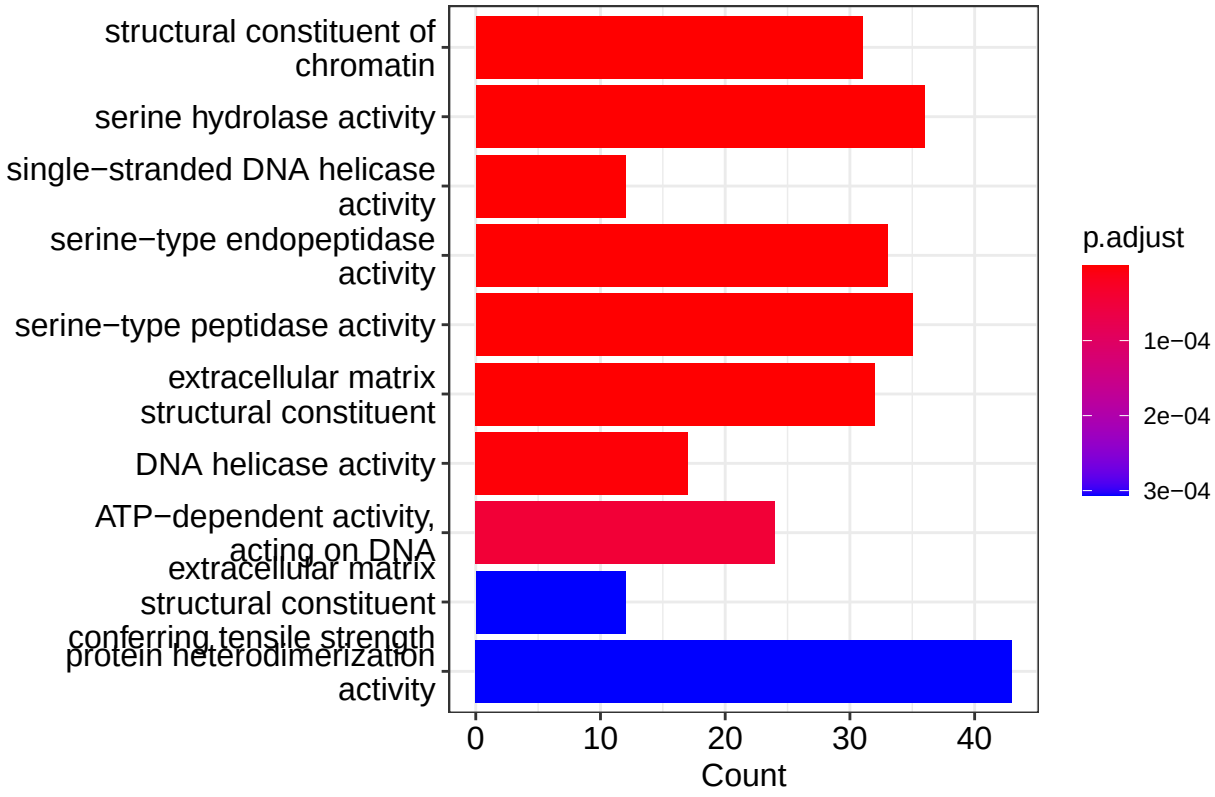


Figure 9: Molecular function down regulated genes

```
##         GeneRatio  BgRatio       pvalue     p.adjust       qvalue
## WP2806    19/437  97/8444 4.087733e-07 0.0001962112 0.0001789997
## WP545      8/437  23/8444 1.188920e-05 0.0028534071 0.0026031083
## WP558     12/437  58/8444 3.217488e-05 0.0051479809 0.0046964036
## WP2431    18/437 121/8444 4.548782e-05 0.0054585389 0.0049797197
## WP5094    24/437 201/8444 1.048950e-04 0.0089843459 0.0081962454
## WP5090    16/437 107/8444 1.123043e-04 0.0089843459 0.0081962454
##
## WP2806                        6401/729/2219/2160/7448/6404/730/653509/22918/5199/5648/11326/5806/235
## WP545                                                                                          732/729/
## WP558                                                                                  729/1361/7450/730/5648/713/
## WP2431                                        2920/4842/57556/6347/1958/3164/7538/3569/7099/9353/2353/361/
## WP5094 6401/862/2920/4616/1839/8013/1958/1959/3164/2354/3949/5468/3569/9314/2890/1346/2353/5581/3400/
## WP5090                                                  732/729/2219/7448/730/5199/5648/8547/81035/1524/
##         Count
## WP2806     19
## WP545       8
## WP558      12
## WP2431     18
## WP5094     24
## WP5090     16
```

```
eWP_down <- enrichWP(gene = downDEGs$entrezgene_id, organism = "Homo sapiens", pvalueCutoff = 0.05,
    qvalueCutoff = 0.1)

head(eWP_down, n = 10)
```

```
##            ID
## WP2446 WP2446
## WP2361 WP2361
## WP179   WP179
## WP1604 WP1604
## WP45     WP45
## WP466   WP466
## WP4240 WP4240
## WP698   WP698
## WP4016 WP4016
## WP5283 WP5283
##                                                                      Description
## WP2446                                                 Retinoblastoma gene in cancer
## WP2361                                                      Gastric cancer network 1
## WP179                                                                  Cell cycle
## WP1604                                               Codeine and morphine metabolism
## WP45                                                     G1 to S cell cycle control
## WP466                                                              DNA replication
## WP4240 Regulation of sister chromatid separation at the metaphase-anaphase transition
## WP698                                                              Glucuronidation
## WP4016                                    DNA IR-damage and cellular response via ATR
## WP5283                              Chronic hyperglycemia impairment of neuron function
##         GeneRatio  BgRatio       pvalue     p.adjust       qvalue
## WP2446    26/618  90/8444 5.946692e-10 3.098226e-07 2.835633e-07
## WP2361    13/618  28/8444 2.049156e-08 4.347812e-06 3.979308e-06
## WP179     28/618 120/8444 2.503538e-08 4.347812e-06 3.979308e-06
## WP1604     9/618  16/8444 4.082807e-07 4.384779e-05 4.013143e-05
```

```
## WP45      18/618  64/8444 4.208041e-07 4.384779e-05 4.013143e-05
## WP466     13/618  42/8444 5.352265e-06 4.647550e-04 4.253642e-04
## WP4240     7/618  15/8444 4.161758e-05 3.097537e-03 2.835002e-03
## WP698      9/618  26/8444 5.724683e-05 3.537787e-03 3.237938e-03
## WP4016    17/618  81/8444 6.111340e-05 3.537787e-03 3.237938e-03
## WP5283    12/618  46/8444 8.178878e-05 4.261195e-03 3.900033e-03
##
## WP2446    1870/54443/4175/4998/24137/8318/5427/1869/4173/898/1871/7272/7153/891/10733/890/1111/9133/5
## WP2361                                                                    6790/22974/4605/4173/1894/1063/715
## WP179  1870/4171/4175/4998/23594/8318/990/1869/4173/898/1871/7272/991/9088/891/9700/890/1029/1111/913
## WP1604                                                                                      1244/1565/87
## WP45                                                  1870/4171/4175/4998/23594/8318/5427/1869/4173/898/1
## WP466                                                                  55388/4171/4175/4998/23594/831
## WP4240
## WP698                                                                                       7358/7363/5457
## WP4016                                                672/5888/4171/63967/8318/1869/2305/83990/55215/
## WP5283                                                                                  8277/4320/4318/6513/43
##         Count
## WP2446     26
## WP2361     13
## WP179      28
## WP1604      9
## WP45       18
## WP466      13
## WP4240      7
## WP698       9
## WP4016     17
## WP5283     12
```

Task 5: Visualization of the enriched pathway

KEGG analysis was performed using the function `enrichKEGG`.

```
eWP_KEGG <- enrichKEGG(gene = upDEGs$entrezgene_id, organism = "human", pvalueCutoff = 0.05,
    qvalueCutoff = 0.1)
```

Here are reported the top 10 enriched KEGG pathways resulting from the up-regulated list of genes.

```
knitr::kable(head(eWP_KEGG[, 1:6], n = 10))
```

ID	Description	GeneRatio	BgRatio	pvalue	p.adjust
hsa05144	Malaria	13/382	50/8464	0.0000002	0.0000306
hsa04610	Complement and coagulation cascades	17/382	86/8464	0.0000002	0.0000306
hsa04270	Vascular smooth muscle contraction	17/382	134/8464	0.0001035	0.0098968
hsa04061	Viral protein interaction with cytokine and cytokine receptor	14/382	100/8464	0.0001464	0.0105063
hsa04614	Renin-angiotensin system	6/382	23/8464	0.0004262	0.0237535
hsa03320	PPAR signaling pathway	11/382	75/8464	0.0004966	0.0237535
hsa04380	Osteoclast differentiation	15/382	128/8464	0.0006213	0.0249197
hsa04514	Cell adhesion molecules	17/382	157/8464	0.0006946	0.0249197
hsa04060	Cytokine-cytokine receptor interaction	26/382	297/8464	0.0008879	0.0280393

ID	Description	GeneRatio	BgRatio	pvalue	p.adjust
hsa04924	hsa04924 Renin secretion	10/382	69/8464	0.0009770	0.0280393

The more enriched pathway (using the list of up-regulated genes) is related to malaria (and has ID `hsa05144`). We then visualized it using the package `pathview`:

```
# set working directory to save output
setwd("~/Documents/Repo_Git/Project_BR/output")

logFC <- upDEGs$logFC
names(logFC) <- upDEGs$entrezgene_id
pathview(gene.data = logFC, pathway.id = "hsa05144", species = "human")
```

Here is reported the network representing the pathway, with the up-regulated genes colored in red:

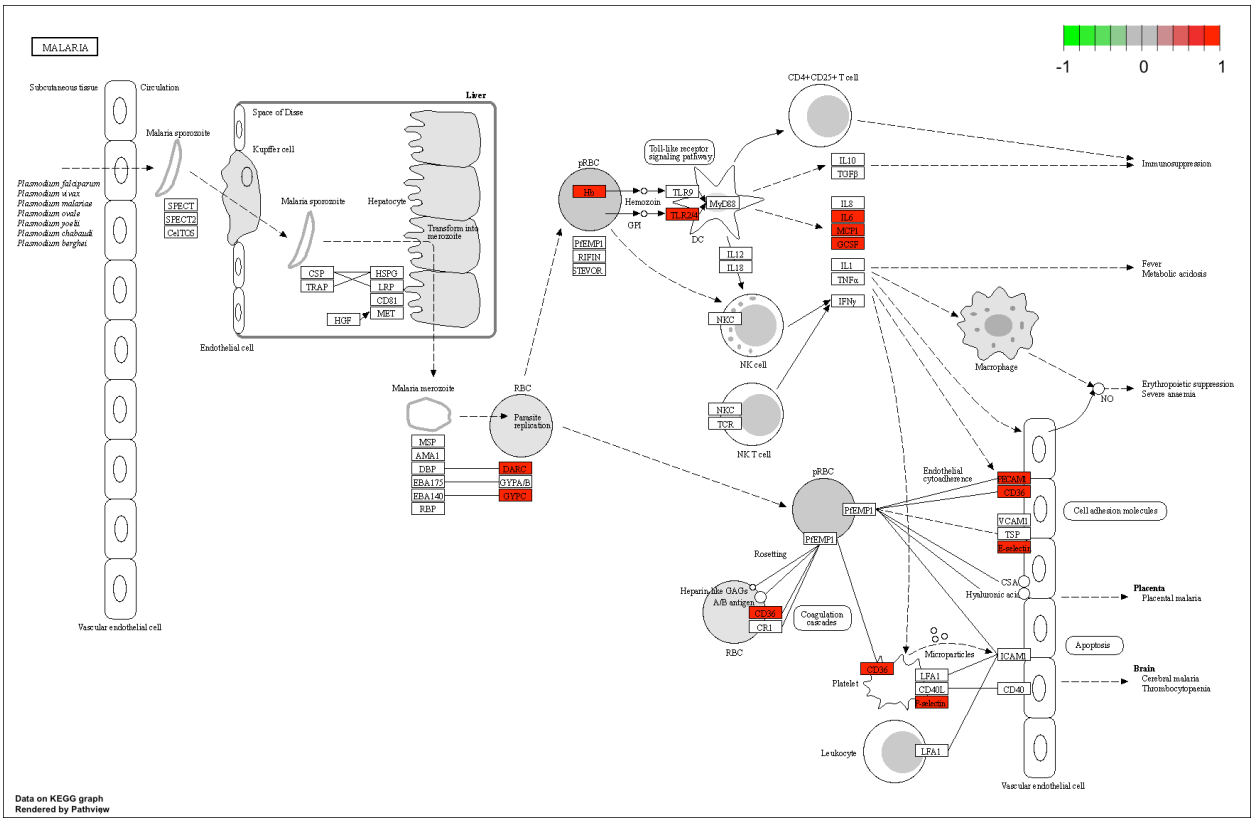


Figure 10: Most enriched pathway

Task 6: Enrichment score transcription factors

First of all, we retrieve the promoter sequence for all the genes in the list of those down-regulated using ensembl gene id as identifier.

```
promoter_regions <- getSequence(id = downDEGs$ensembl_gene_id, type = "ensembl_gene_id",
    seqType = "gene_flank", upstream = 500, mart = ensembl)
```

Then we load the background for MotifDb human PWMs and convert the obtained sequence into a DNAString object.

```
data(PWMLogn.hg19.MotifDb.Hsap)
seq <- lapply(promoter_regions$gene_flank, function(x) DNAString(x))
```

`motifEnrichment` function performs enrichment analysis on the promoter sequences we obtained a few chunks above.

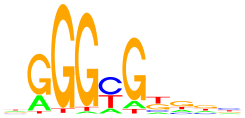
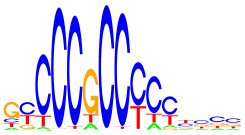
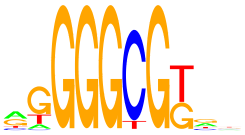
```
enriched_TFs = motifEnrichment(seq, PWMLogn.hg19.MotifDb.Hsap, score = "affinity")
```

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

```
report = groupReport(enriched_TFs)
```

Here we plot the top 5 enriched transcription factors resulted from our analysis:

```
plot(report[1:5])
```

Rank	Target	PWM	Motif ID	Raw score	P-value	In top motifs
1	PGAM2		PGAM2	7.47	2.41e−86	18 %
2	TFAP4		M2944_1.02	3.42	3.37e−84	18 %
3	SP2		SP2	12	0.6.25e−84	19 %
4	CEBPB		M4556_1.02	2.31	1.71e−83	17 %
5	JUND		M4506_1.02	6.86	1.5e−81	19 %

Task 7

We select one among the top enriched TFs (TFAP4), compute the empirical distributions of scores for all PWMs that you find in MotifDB for the selected TF and determine for all of them the distribution (log2) threshold cutoff at 99.75%.

```
tfs <- report$target[2]  # contains TFAP4
tfs_motifs = subset(MotifDb, organism == "Hsapiens" & geneSymbol == tfs)

# transformation to a PWM matrix
PWM = toPWM(as.list(tfs_motifs))
```

For the selected TF we found 15 PWMs:

- Hsapiens-cisbp_1.02-M2943_1.02
- Hsapiens-cisbp_1.02-M2944_1.02
- Hsapiens-cisbp_1.02-M2947_1.02
- Hsapiens-cisbp_1.02-M5926_1.02
- Hsapiens-cisbp_1.02-M5927_1.02
- Hsapiens-cisbp_1.02-M6513_1.02
- Hsapiens-HOCOMOCOv10-TFAP4_HUMAN.H10MO.C
- Hsapiens-HOCOMOCOv11-core-A-TFAP4_HUMAN.H11MO.0.A
- Hsapiens-jaspar2016-TFAP4-MA0691.1
- Hsapiens-jaspar2018-TFAP4-MA0691.1
- Hsapiens-jaspar2022-TFAP4-MA0691.1
- Hsapiens-jaspar2022-TFAP4-MA1570.1
- Hsapiens-jolma2013-TFAP4
- Hsapiens-jolma2013-TFAP4-2
- Hsapiens-SwissRegulon-TFAP4.SwissRegulon

To get the empirical distribution we used the function `motifEcdf`, which associates to each PWM a distribution. Then, for each score we computed the scores with a cutoff threshold of 99.75%.

```
ecdf = motifEcdf(PWM, organism = "hg19", quick = TRUE)

thresholds = lapply(ecdf, function(x) log2(quantile(x, 0.9975)))
# for each of the distribution, take the quantile as reference

scores = motifScores(seq, PWM, raw.scores = FALSE, cutoff = unlist(thresholds))
```

In the following table are reported the calculated scores for each PWM of the first 10 motifs.

```
head(scores, n = 10)
```

```
##       Hsapiens-cisbp_1.02-M2943_1.02 Hsapiens-cisbp_1.02-M2944_1.02
##  [1,]                              2                              2
##  [2,]                              3                              2
##  [3,]                              3                              4
##  [4,]                              6                              6
```

```
##  [5,]                                2                                3
##  [6,]                                0                                0
##  [7,]                                4                                4
##  [8,]                                4                                3
##  [9,]                                1                                3
## [10,]                                3                                4
##       Hsapiens-cisbp_1.02-M2947_1.02 Hsapiens-cisbp_1.02-M5926_1.02
##  [1,]                                3                                2
##  [2,]                                2                                0
##  [3,]                                1                                2
##  [4,]                                6                                5
##  [5,]                                5                                2
##  [6,]                                1                                4
##  [7,]                                4                                6
##  [8,]                                4                                0
##  [9,]                                3                                3
## [10,]                                1                                0
##       Hsapiens-cisbp_1.02-M5927_1.02 Hsapiens-cisbp_1.02-M6513_1.02
##  [1,]                                6                                2
##  [2,]                                0                                4
##  [3,]                                1                                3
##  [4,]                                4                                4
##  [5,]                                2                                4
##  [6,]                                4                                0
##  [7,]                                4                                3
##  [8,]                                1                               10
##  [9,]                                4                                6
## [10,]                                0                                2
##       Hsapiens-HOCOMOCOv10-TFAP4_HUMAN.H10MO.C
##  [1,]                                        2
##  [2,]                                        4
##  [3,]                                        3
##  [4,]                                        4
##  [5,]                                        4
##  [6,]                                        0
##  [7,]                                        3
##  [8,]                                        9
##  [9,]                                        7
## [10,]                                        2
##       Hsapiens-HOCOMOCOv11-core-A-TFAP4_HUMAN.H11MO.0.A
##  [1,]                                                 1
##  [2,]                                                 3
##  [3,]                                                 1
##  [4,]                                                 3
##  [5,]                                                 2
##  [6,]                                                 2
##  [7,]                                                 6
##  [8,]                                                 4
##  [9,]                                                 5
## [10,]                                                 2
##       Hsapiens-jaspar2016-TFAP4-MA0691.1 Hsapiens-jaspar2018-TFAP4-MA0691.1
##  [1,]                                  6                                  6
##  [2,]                                  0                                  0
##  [3,]                                  1                                  1
```

```
##  [4,]                                4                                4
##  [5,]                                2                                2
##  [6,]                                4                                4
##  [7,]                                4                                4
##  [8,]                                1                                1
##  [9,]                                4                                4
## [10,]                                0                                0
##       Hsapiens-jaspar2022-TFAP4-MA0691.1 Hsapiens-jaspar2022-TFAP4-MA1570.1
##  [1,]                                6                                5
##  [2,]                                0                                0
##  [3,]                                1                                2
##  [4,]                                4                                8
##  [5,]                                2                                0
##  [6,]                                4                                6
##  [7,]                                4                               11
##  [8,]                                1                                6
##  [9,]                                4                                4
## [10,]                                0                                3
##       Hsapiens-jolma2013-TFAP4 Hsapiens-jolma2013-TFAP4-2
##  [1,]                        2                          6
##  [2,]                        0                          0
##  [3,]                        2                          1
##  [4,]                        5                          4
##  [5,]                        2                          2
##  [6,]                        4                          4
##  [7,]                        6                          4
##  [8,]                        0                          1
##  [9,]                        3                          4
## [10,]                        0                          0
##       Hsapiens-SwissRegulon-TFAP4.SwissRegulon
##  [1,]                                        2
##  [2,]                                        0
##  [3,]                                        2
##  [4,]                                        5
##  [5,]                                        2
##  [6,]                                        4
##  [7,]                                        6
##  [8,]                                        0
##  [9,]                                        3
## [10,]                                        0
```

Task 8: Pattern matching

We identify which down-regulated genes have a region in their promoter (defined as previously) with binding scores above the computed thresholds for any of the previously selected PWMs.

To do that, we using the table of scores displayed before. We selected only genes having a region in their promoter with binding scores above the computed thresholds for any of the previously selected PWMs.

```
fractions_99.75 <- c()
fractions_99.75 <- c(fractions_99.75, length(which(apply(scores, 1, sum) > 0))/length(scores))
fractions_99.75
```

```
## [1] 0.06638514
```

```
highscore_seq <- which(apply(scores, 1, sum) > 0)
genes_id <- downDEGs[highscore_seq, ]
```

The resulting list contains 1179 genes.

```
dim(genes_id)
```

```
## [1] 1179   10
```

Task 9: PPI interactions

We use STRING database to find PPI interactions among differentially expressed genes.

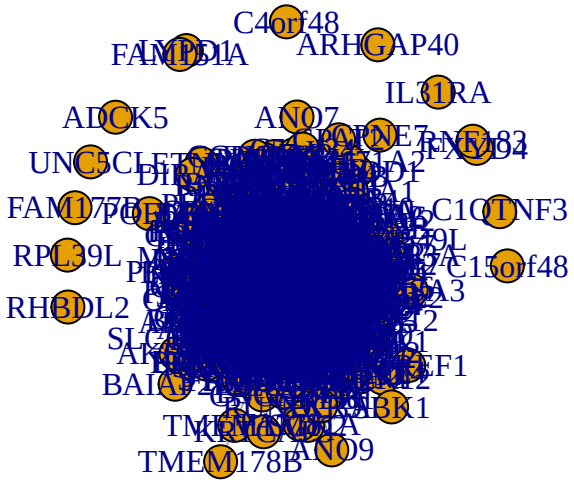
```
dat <- read.table("output/down_DEGs.txt", sep = "\t", header = TRUE)

write.table(dat$ensembl_gene_id, file = "output/ens_gene_id_down.txt", row.names = FALSE,
    col.names = FALSE)
```

We generate a `.txt` file with the gene of interest to retrieve from STRING the network. Then, the network is exported in `.tsv` format.

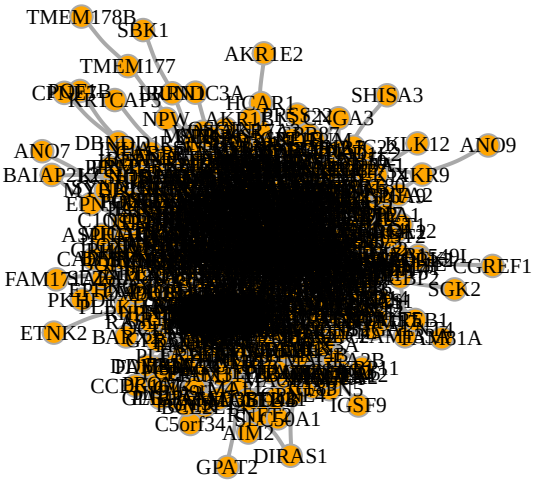
Task 10: visualize the network

We import the network in R and, using `igraph` package, we identify and plot the largest connected component. A connected component of a graph is a subgraph in which all nodes are connected with each other via a path. So, it exists a path (series of edges belonging to set of edges E) that connects any two nodes belonging to the subgraph. Identifying connected components is usually useful in biology to underline some patterns or groups of genes which are extremely connected and which tend to influence each other.

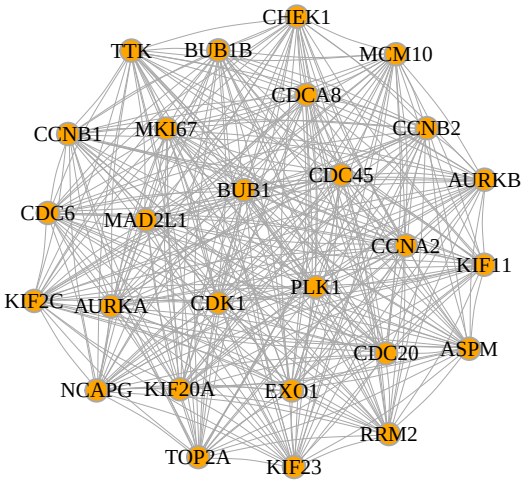
```
links_high_conf <- read.delim("data_STRING/string_interactions_short_high_conf.tsv")
links <- read.delim("data_STRING/string_interactions_short.tsv")
downDEGs <- read.table("output/down_DEGs.txt", sep = "\t", header = TRUE)

ensembl <- useMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl")
nodes <- getBM(attributes = c("external_gene_name", "ensembl_gene_id", "description",
    "gene_biotype", "start_position", "end_position", "chromosome_name", "strand"),
    filters = c("ensembl_gene_id"), values = downDEGs[, 1], mart = ensembl)  # search info about nodes
nodes <- unique(nodes[, c(1, 3:6)])

nodes <- nodes[nodes$external_gene_name %in% links$X.node1 & nodes$external_gene_name %in%
    links$X.node1, ]

links <- links[links$X.node1 %in% nodes$external_gene_name & links$node2 %in% nodes$external_gene_name,
    ]

# create network
net <- graph_from_data_frame(d = links, vertices = nodes, directed = FALSE)

plot(net)
```




```
# identifying components
c <- components(net, mode = c("weak", "strong"))
# taking only the biggest component
net.c <- induced_subgraph(net, V(net)[which(c$membership == 1)])

plot(net.c, edge.width = 2, vertex.color = "orange", vertex.size = 10, vertex.frame.color = "darkgray",
    vertex.label.color = "black", vertex.label.cex = 0.7, edge.curved = 0.1)
```

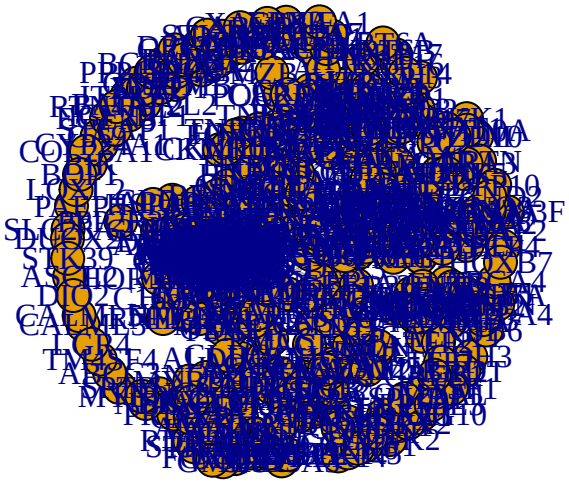



```
# this first plot looks very confusing, there are a lot of connections. Let's
# try to take only nodes with at least a certain degree value.

deg <- degree(net.c, mode = "all")
names_nodes <- names(deg[which(deg > 150)])

nodes.deg <- nodes[nodes$external_gene_name %in% names_nodes, ]

nodes.deg <- nodes.deg[nodes.deg$external_gene_name %in% links$X.node1 & nodes.deg$external_gene_name %
    links$node2, ]
links.deg <- links[links$X.node1 %in% nodes.deg$external_gene_name & links$node2 %in%
    nodes.deg$external_gene_name, ]

net.deg <- graph_from_data_frame(d = links.deg, vertices = nodes.deg, directed = FALSE)

plot(net.deg, edge.width = 0.5, vertex.color = "orange", vertex.size = 10, vertex.frame.color = "darkgra
    vertex.label.color = "black", vertex.label.cex = 0.7, edge.curved = 0.1)
```

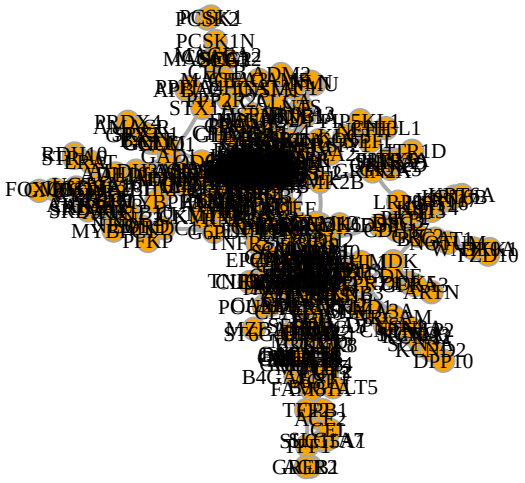


Let's try now to set a higher confidence on STRING since we have just seen that the first graph is quite dense of nodes and edges.

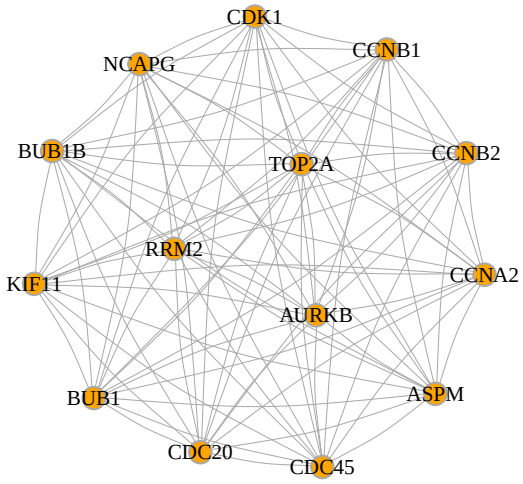
```
links_high_conf <- read.delim("data_STRING/string_interactions_short_high_conf.tsv")
downDEGs <- read.table("output/down_DEGs.txt", sep = "\t", header = TRUE)

ensembl <- useMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl")
nodes <- getBM(attributes = c("external_gene_name", "ensembl_gene_id", "description",
    "gene_biotype", "start_position", "end_position", "chromosome_name", "strand"),
    filters = c("ensembl_gene_id"), values = downDEGs[, 1], mart = ensembl)  # search info about nodes
nodes <- unique(nodes[, c(1, 3:6)])

nodes <- nodes[nodes$external_gene_name %in% links_high_conf$X.node1 & nodes$external_gene_name %in%
    links_high_conf$X.node1, ]

links_high_conf <- links_high_conf[links_high_conf$X.node1 %in% nodes$external_gene_name &
    links_high_conf$node2 %in% nodes$external_gene_name, ]

# create network
net <- graph_from_data_frame(d = links_high_conf, vertices = nodes, directed = FALSE)

plot(net)
```



```r
# identifying components
c <- components(net, mode = "strong")
# taking only the biggest component
net.c <- induced_subgraph(net, V(net)[which(c$membership == 1)])

plot(net.c, edge.width = 2, vertex.color = "orange", vertex.size = 10, vertex.frame.color = "darkgray",
    vertex.label.color = "black", vertex.label.cex = 0.7, edge.curved = 0.1)
```



```r
# Let's try again to keep only nodes with at least a certain degree value
deg <- degree(net.c, mode = "all")
names_nodes <- names(deg[which(deg > 110)])

nodes.deg <- nodes[nodes$external_gene_name %in% names_nodes, ]

nodes.deg <- nodes.deg[nodes.deg$external_gene_name %in% links_high_conf$X.node1 &
    nodes.deg$external_gene_name %in% links_high_conf$node2, ]
links.deg.high <- links_high_conf[links_high_conf$X.node1 %in% nodes.deg$external_gene_name &
    links_high_conf$node2 %in% nodes.deg$external_gene_name, ]

net.deg <- graph_from_data_frame(d = links.deg.high, vertices = nodes.deg, directed = FALSE)

plot(net.deg, edge.width = 0.5, vertex.color = "orange", vertex.size = 10, vertex.frame.color = "darkgra
    vertex.label.color = "black", vertex.label.cex = 0.7, edge.curved = 0.1)
```


CDK1
CCNB1
NCAPG
BUB1B
TOP2A
CCNB2
RRM2
KIF11
CCNA2
AURKB
BUB1
ASPM
CDC20
CDC45