

# 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

The packages needed for the analysis are the following:

```
library(sessioninfo)
library(tidyverse)
library(biomaRt)
library(MotifDb)  # large collection of motifs across different species
library(seqLogo)
library(PWMErich)
library(PWMErich.Hsapiens.background)
library(GEOquery)
library(oligo)
library(pd.hg.u133.plus.2)
library(hgu133plus2.db)
library(genefilter)
library(limma)
library(pheatmap)
library(stringr)
library(GenomicFeatures)  # to build object with specific information such as genomic coordinates
library(ggplot2)
library(edgeR)  # designed to perform Differential Expression Analysis from RNA-Seq data
library(fgsea)  # computation of enrichment scores and other statistics
library(org.Hs.eg.db)  # database annotation human specific transcript from Ensembl
library(clusterProfiler)  # uses Fisher test, explore gene list of interest against a reference
library(enrichplot)  # build enrich map
library(ggnewscale)
library(DOSE)
library(pathview)  # contains cartoons of pathways and we project on them our genes of interest
library(igraph)
```

## 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, # go through all count matrices by rows
  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 above the count
```

Then, we filter the previously updated data frames.



```
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()
```

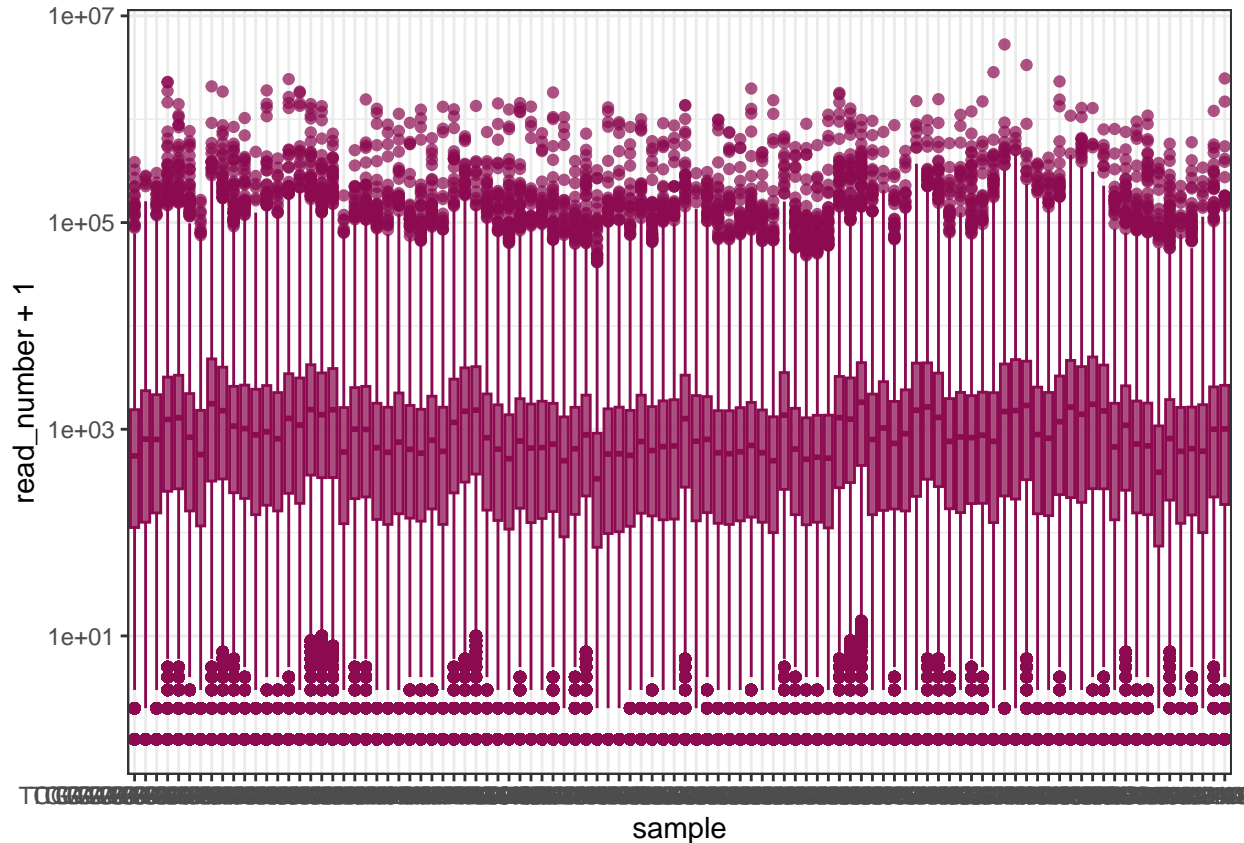


Figure 2: Boxplot of gene counts before normalization.

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 `DGRLIST` 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()
```

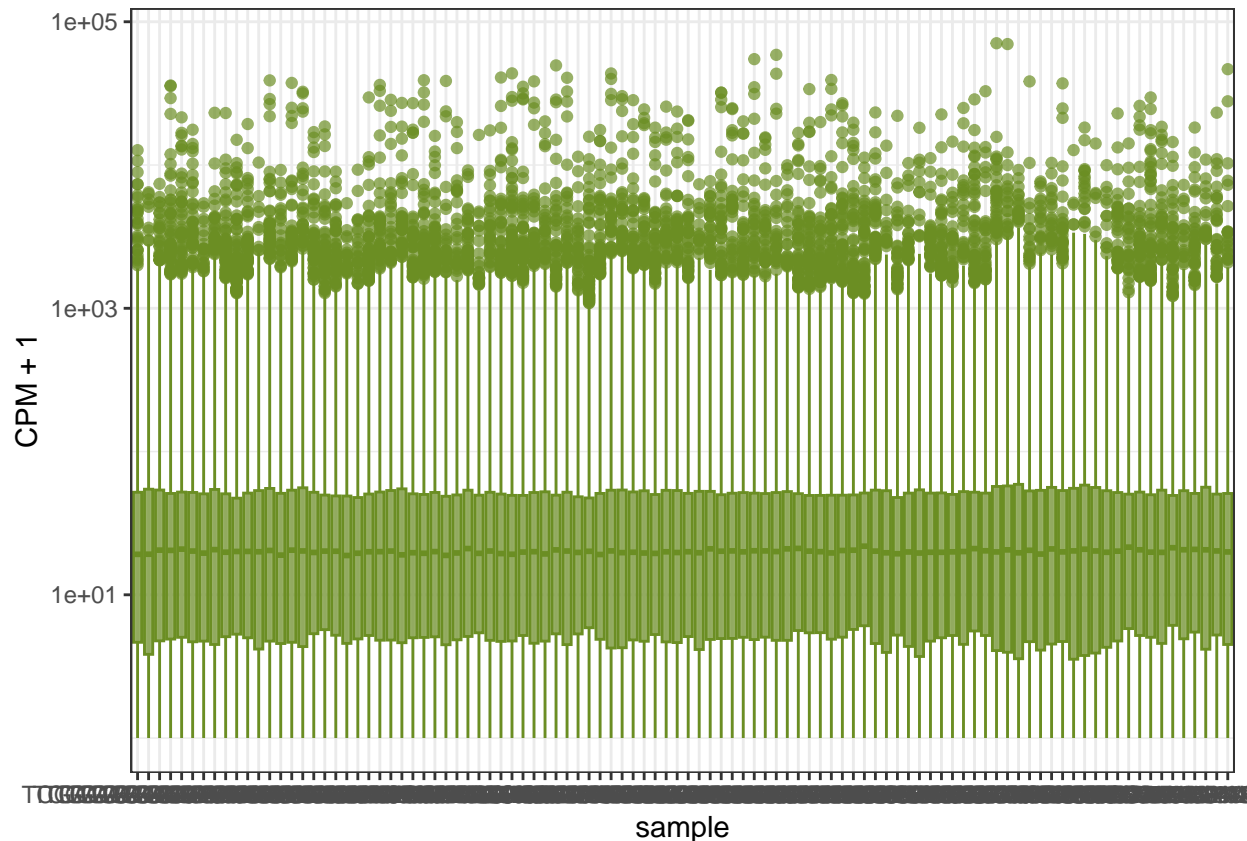


Figure 3: Boxplot distribution of gene expression signals after normalization.

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)
```

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  $\log FC > 1.5$
- down-regulated genes if  $\log FC < -1.5$
- 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")
```

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)
```

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.

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",
```

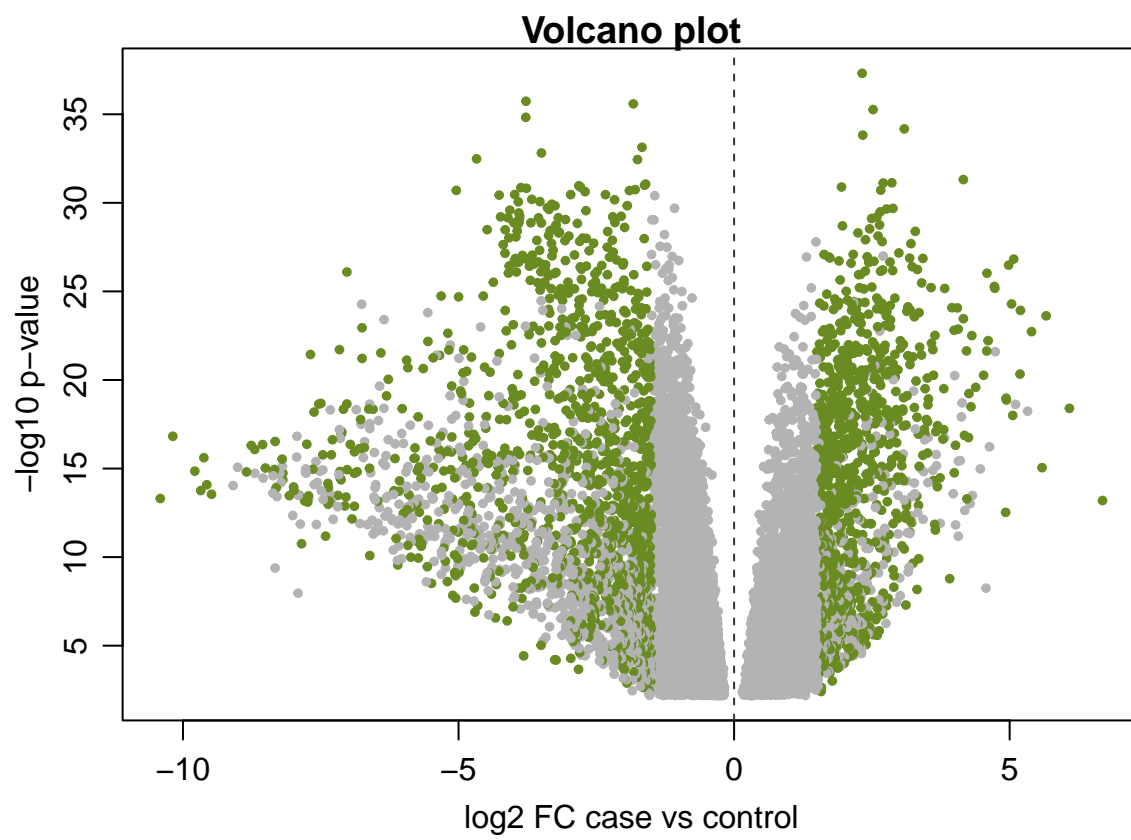


Figure 4: Volcano plot





```

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 info
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")
```

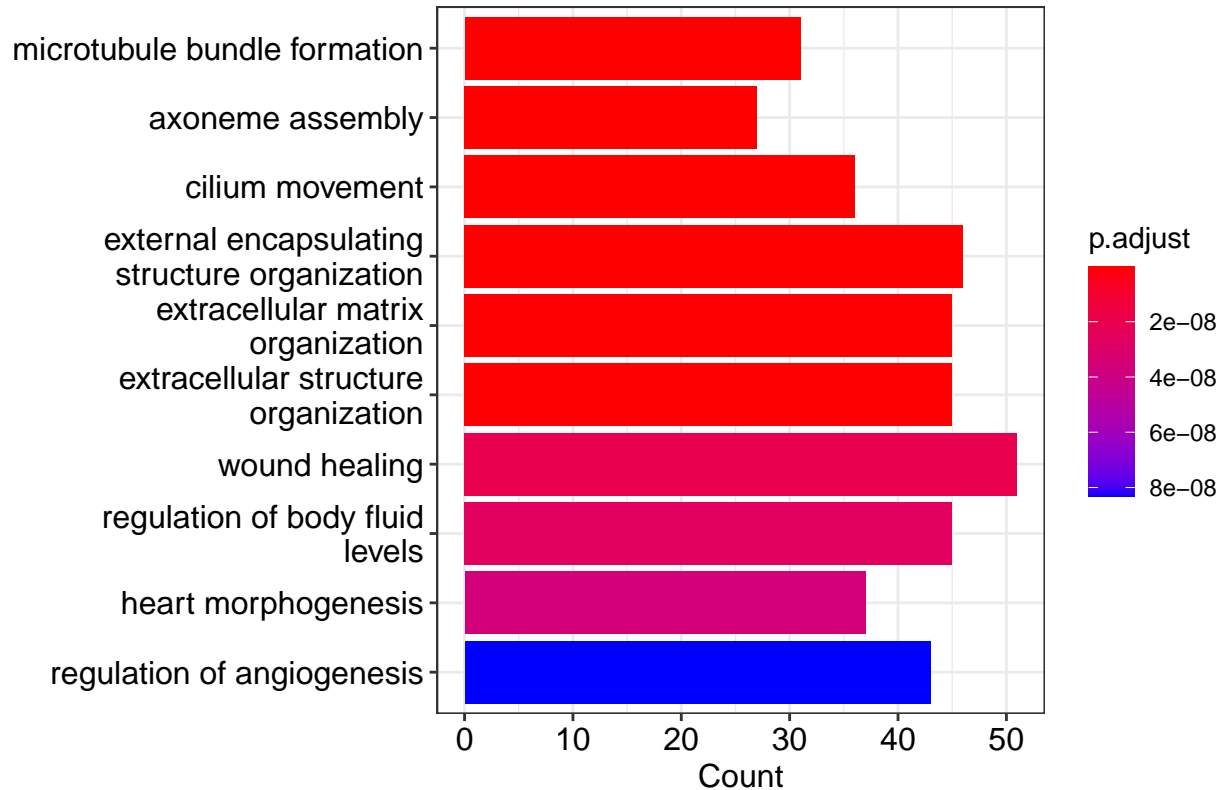


Figure 6: Biological process up regulated genes

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")
```

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 explanation 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.

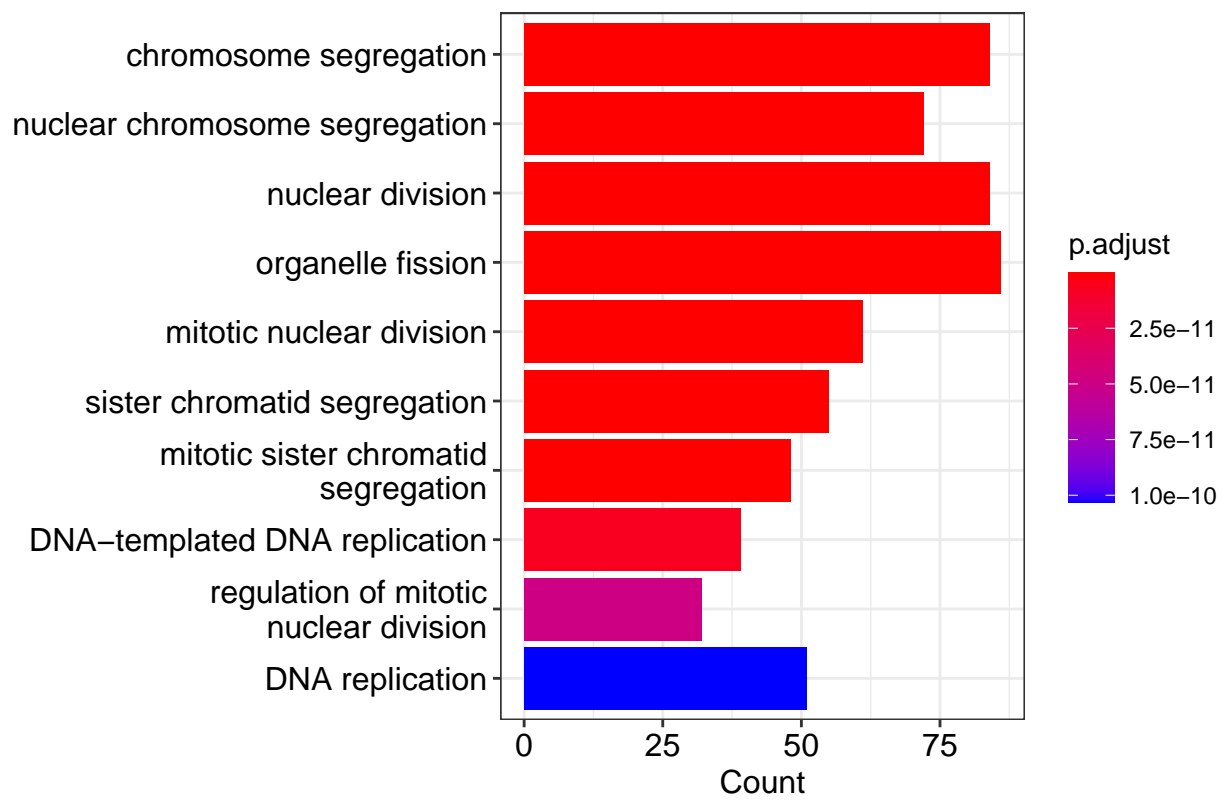


Figure 7: Biological process down regulated genes

## 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)
```

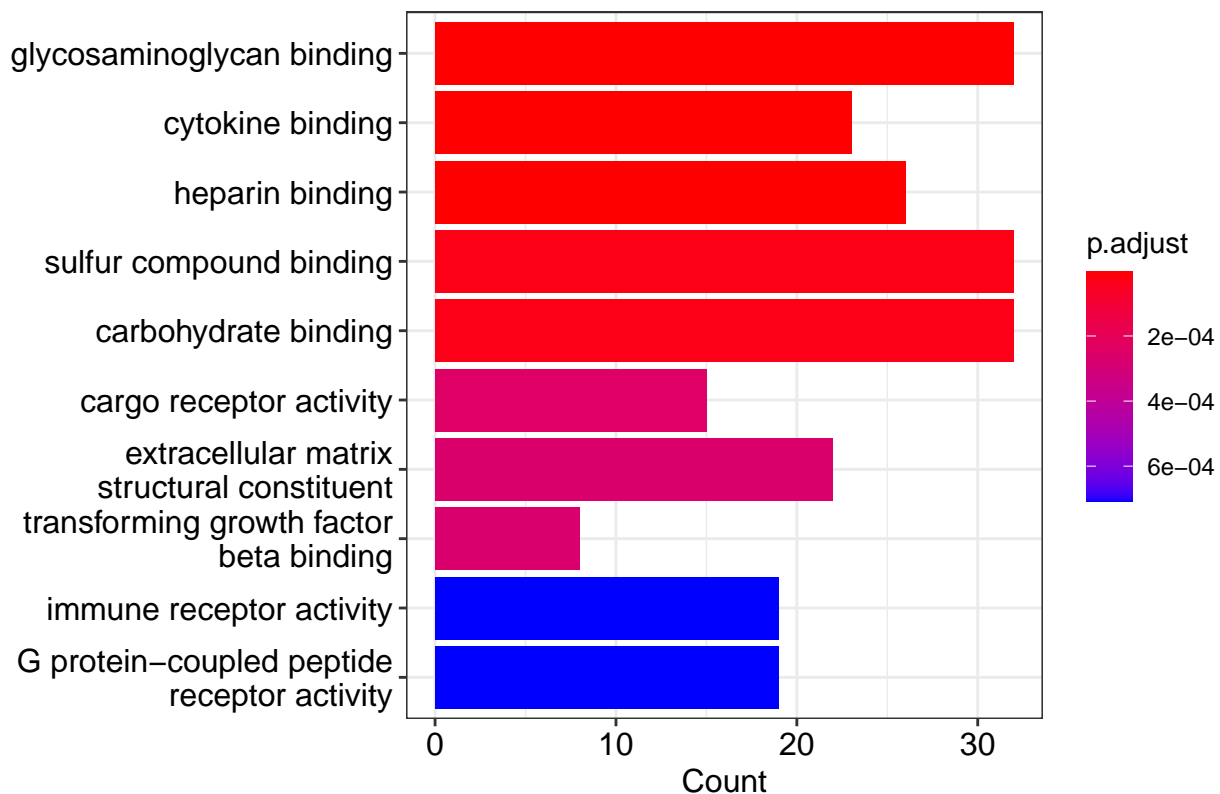


Figure 8: Molecular function up regulated genes

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

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

head(eWP_up, n = 10)
```

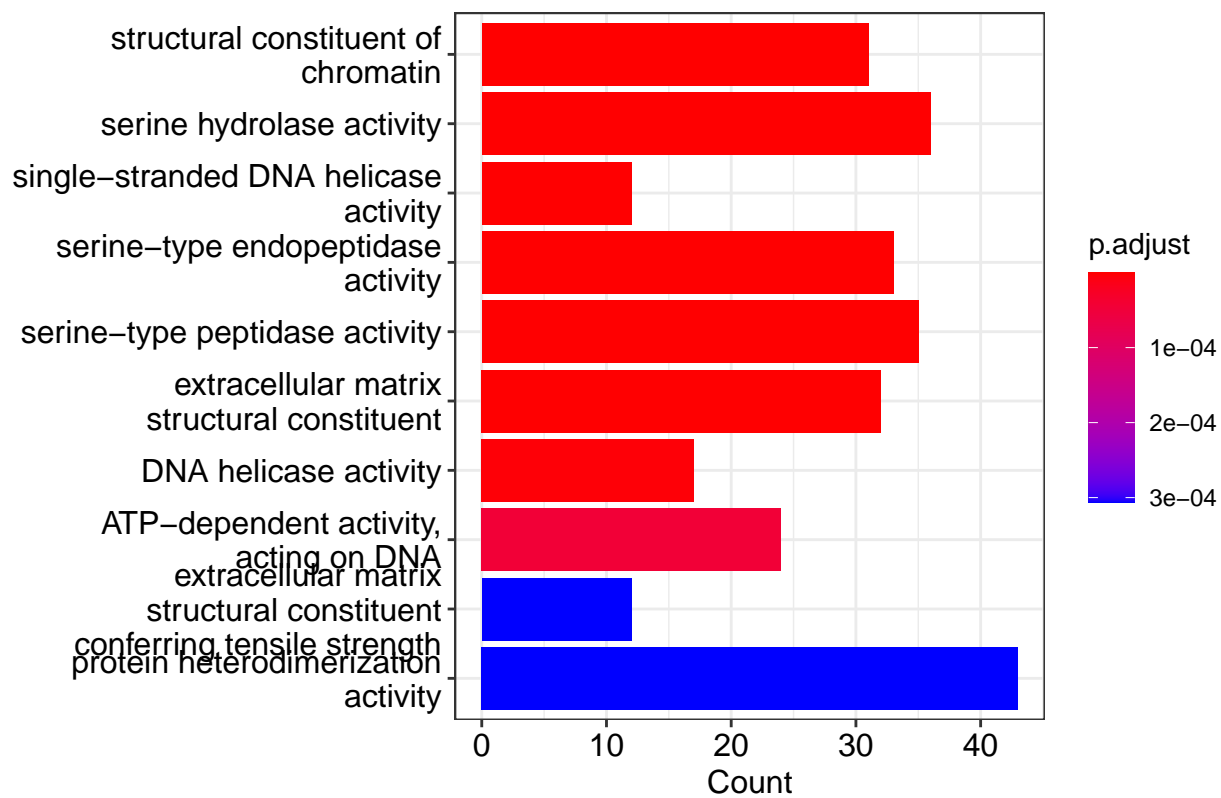


Figure 9: Molecular function down regulated genes

##	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
##	GeneRatio BgRatio pvalue p.adjust qvalue	
##	WP2806 19/436 97/8442 3.960197e-07 0.0001892974 0.0001725812	
##	WP545 8/436 23/8442 1.171095e-05 0.0027989174 0.0025517547	
##	WP558 12/436 58/8442 3.152788e-05 0.0050234421 0.0045798393	
##	WP2431 18/436 121/8442 4.427616e-05 0.0052910008 0.0048237708	
##	WP5094 24/436 201/8442 1.015995e-04 0.0087342052 0.0079629178	
##	WP5090 16/436 107/8442 1.096344e-04 0.0087342052 0.0079629178	
##		
##	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	Description
##	WP2446 WP2446	Retinoblastoma gene in cancer
##	WP2361 WP2361	Gastric cancer network 1
##	WP179 WP179	Cell cycle
##	WP1604 WP1604	Codeine and morphine metabolism
##	WP45 WP45	G1 to S cell cycle control
##	WP466 WP466	DNA replication
##	WP4240 WP4240	Regulation of sister chromatid separation at the metaphase-anaphase transition
##	WP698 WP698	Glucuronidation

```

## WP4016 DNA IR-damage and cellular response via ATR
## WP2525 Trans-sulfuration, one-carbon metabolism and related pathways
## GeneRatio BgRatio pvalue p.adjust qvalue
## WP2446 26/618 90/8442 5.976882e-10 3.107978e-07 2.850029e-07
## WP2361 13/618 28/8442 2.054960e-08 4.361461e-06 3.999477e-06
## WP179 28/618 120/8442 2.516227e-08 4.361461e-06 3.999477e-06
## WP1604 9/618 16/8442 4.091055e-07 4.391632e-05 4.027145e-05
## WP45 18/618 64/8442 4.222724e-07 4.391632e-05 4.027145e-05
## WP466 13/618 42/8442 5.366172e-06 4.650682e-04 4.264694e-04
## WP4240 7/618 15/8442 4.168136e-05 3.096330e-03 2.839347e-03
## WP698 9/618 26/8442 5.735318e-05 3.541417e-03 3.247494e-03
## WP4016 17/618 81/8442 6.129375e-05 3.541417e-03 3.247494e-03
## WP2525 15/618 68/8442 9.059661e-05 4.711024e-03 4.320028e-03
##
## WP2446 1870/54443/4175/4998/24137/8318/5427/1869/4173/898/1871/7272/7153/891/10733/890/1111/9133/
## WP2361 6790/22974/4605/4173/1894/1063/71
## WP179 1870/4171/4175/4998/23594/8318/990/1869/4173/898/1871/7272/991/9088/891/9700/890/1029/1111/91
## WP1604 1244/1565/8
## WP45 1870/4171/4175/4998/23594/8318/5427/1869/4173/898/
## WP466 55388/4171/4175/4998/23594/83
## WP4240
## WP698 7358/7363/545
## WP4016 672/5888/4171/63967/8318/1869/2305/83990/55215
## WP2525 2729/2730/10797/1789/2571/23743/29968/570/5
## Count
## WP2446 26
## WP2361 13
## WP179 28
## WP1604 9
## WP45 18
## WP466 13
## WP4240 7
## WP698 9
## WP4016 17
## WP2525 15

```

## Task 5: Visualization of the enriched pathway

```

logFC <- upDEGs$logFC
names(logFC) <- upDEGs$entrezgene_id
pathview(gene.data = logFC, pathway.id = "WP2446", species = "human")

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

head(eWP_KEGG, n = 20)

```

```

## ID Description
## hsa05144 hsa05144 Malaria
## hsa04610 hsa04610 Complement and coagulation cascades
## hsa04270 hsa04270 Vascular smooth muscle contraction

```



##	hsa04061	hsa04061	Viral protein interaction with cytokine and cytokine receptor			
##	hsa04614	hsa04614	Renin-angiotensin system			
##	hsa03320	hsa03320	PPAR signaling pathway			
##	hsa04380	hsa04380	Osteoclast differentiation			
##	hsa04514	hsa04514	Cell adhesion molecules			
##	hsa04060	hsa04060	Cytokine-cytokine receptor interaction			
##	hsa04924	hsa04924	Renin secretion			
##	hsa04080	hsa04080	Neuroactive ligand-receptor interaction			
##	hsa04925	hsa04925	Aldosterone synthesis and secretion			
##	hsa04976	hsa04976	Bile secretion			
##		GeneRatio	BgRatio	pvalue	p.adjust	qvalue
##	hsa05144	13/382	50/8464	2.037683e-07	3.062438e-05	2.673245e-05
##	hsa04610	17/382	86/8464	2.134103e-07	3.062438e-05	2.673245e-05
##	hsa04270	17/382	134/8464	1.034508e-04	9.896797e-03	8.639052e-03
##	hsa04061	14/382	100/8464	1.464287e-04	1.050626e-02	9.171062e-03
##	hsa04614	6/382	23/8464	4.261842e-04	2.375352e-02	2.073478e-02
##	hsa03320	11/382	75/8464	4.965893e-04	2.375352e-02	2.073478e-02
##	hsa04380	15/382	128/8464	6.213148e-04	2.491966e-02	2.175272e-02
##	hsa04514	17/382	157/8464	6.946247e-04	2.491966e-02	2.175272e-02
##	hsa04060	26/382	297/8464	8.878939e-04	2.803929e-02	2.447589e-02
##	hsa04924	10/382	69/8464	9.769787e-04	2.803929e-02	2.447589e-02
##	hsa04080	30/382	367/8464	1.120368e-03	2.923141e-02	2.551651e-02
##	hsa04925	12/382	98/8464	1.455772e-03	3.481721e-02	3.039243e-02
##	hsa04976	11/382	89/8464	2.105813e-03	4.648988e-02	4.058167e-02
##						
##	hsa05144					6401/14
##	hsa04610					732/729/1361/2160/74
##	hsa04270					10203/1906/50487/10268/196883/10266
##	hsa04061					53832/2920/6369
##	hsa04614					
##	hsa03320					
##	hsa04380					7305/6688/10326/54/110
##	hsa04514					6401/51208/8516/64115/6404/90952/584
##	hsa04060					53832/2920/3568/6369/1440/6347/2690/3977/9173/8809/650/51554/3569/90865/9
##	hsa04924					
##	hsa04080	153/10203/1906/1511/5023/2690/7433/1268/5737/8698/2922/187/1910/6751/185/2900/2890/5745/112				
##	hsa04925					4
##	hsa04976					
##		Count				
##	hsa05144	13				
##	hsa04610	17				
##	hsa04270	17				
##	hsa04061	14				
##	hsa04614	6				
##	hsa03320	11				
##	hsa04380	15				
##	hsa04514	17				
##	hsa04060	26				
##	hsa04924	10				
##	hsa04080	30				
##	hsa04925	12				
##	hsa04976	11				

## Task 6: Enrichment score transcription factors

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

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

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

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

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

```
## An object of class 'MotifEnrichmentReport':
```

```
##      rank target      id      raw.score      p.value
## 1      1  PGAM2      PGAM2 7.47058707343782 2.41159784654048e-86
## 2      2  TFAP4  M2944_1.02 3.42391273544219 3.36836197404242e-84
## 3      3   SP2      SP2 119.54842162525 6.25197615973972e-84
## 4      4  CEBPB  M4556_1.02 2.30767167521828 1.7054813305699e-83
## 5      5  JUND   M4506_1.02 6.8634137704217 1.50332672791467e-81
## 6      6  MAFK   M4573_1.02 2.41935358645508 4.83938302985492e-81
## 7      7  ZMAT2   ZMAT2 2.16420212862343 9.19698805317131e-81
## 8      8   JUN   M4591_1.02 2.76520576103839 2.36897498529839e-80
## 9      9  KLF5      KLF5 30.1023912345585 2.91297830264159e-79
## 10     10  NNT      NNT 1.87741003326598 6.17885793377267e-79
## ...    ...    ...    ...    ...    ...
## 2287 2001.5 Oct-1 NBT06/Oct-1.pwm 1.72351938773035 1
##      top.motif.prop
## 1      0.177364864864865
## 2      0.178209459459459
## 3      0.185810810810811
## 4      0.174831081081081
## 5      0.185810810810811
## 6      0.171452702702703
## 7      0.163851351351351
## 8      0.168074324324324
## 9      0.158783783783784
## 10     0.173986486486486
## ...    ...
## 2287 0.0278716216216216
```

## Task 7

Select one among the top enriched TFs, 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[1]

tfs_motifs = subset(MotifDb, organism == "Hsapiens" & geneSymbol == tfs)
# tfs_motifs

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

ecdf = motifEcdf(PWM, organism = "hg19", quick = TRUE)
# calculate for each motif of my gene the empirical distribution scores ecdf

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

## Task 8

Identify which up-regulated (or down-regulated depending on the choice you made at point 7) genes have a region in their promoter (defined as previously) with binding scores above the computed thresholds for any of the previously selected PWMs. Use pattern matching as done during the course

```
scores = motifScores(seq, PWM, raw.scores = FALSE, cutoff = unlist(thresholds)) # in my promoters, i w

highscore_seq <- which(apply(scores, 1, sum) > 0)
genes_id <- promoter_regions$ensembl_gene_id[highscore_seq]
# genes_id # down-regulated genes that have a region in their promoter with
# binding scores above the defined threshold for any PWMs
write(genes_id, "output/enriched_genes_id.txt")
```

## Task 9

Use STRING database to find PPI interactions among differentially expressed genes and export the network in TSV format.

```
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)
```

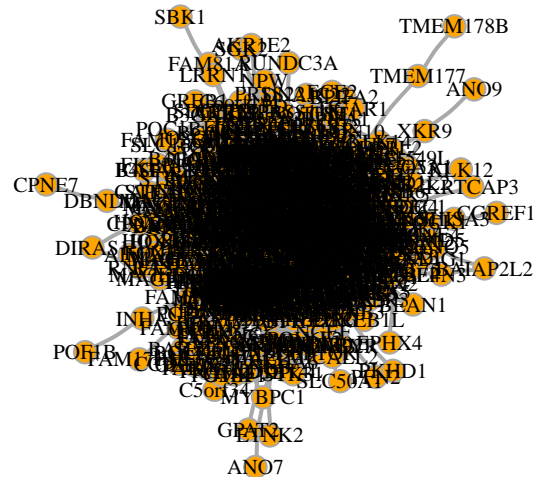
## Task 10

Import the network in R and using igraph package and identify and plot the largest connected component.

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

ensembl <- useMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl")
```





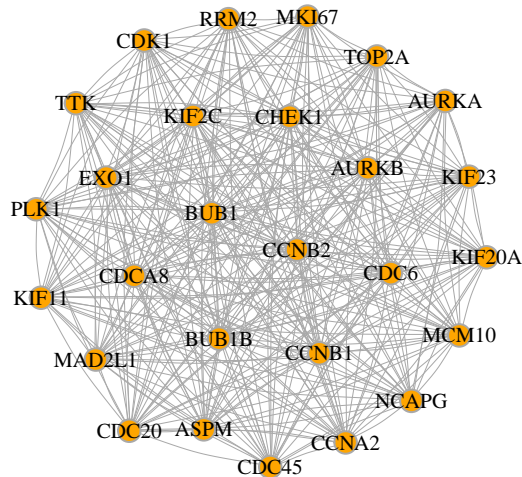
```
deg <- degree(net.c, mode = "all")
names_nodes <- names(deg[which(deg > 150)])

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$external_gene_name %in%
  links$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 = "darkgreen",
  vertex.label.color = "black", vertex.label.cex = 0.7, edge.curved = 0.1)
```



```

links_high_conf <- read.delim("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)

```





```
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 = "darkgray",
  vertex.label.color = "black", vertex.label.cex = 0.7, edge.curved = 0.1)
```



