Week 6 – R analysis: From DEGs to Top 30 genes

The created **GP.csv** (for counts) and **metadata.csv** were loaded onto a colab environment for further analysis. Google colab was chosen for this task due to its easy-to-use interface and the ability to create an R environment inside a python file, which is much easier to work with compared to conventional R files. This platform makes it easier to switch between python and R when needed.

The google drive was first mounted:

```
from google.colab import drive
drive.mount('/content/drive')
```

Then , the R environment was invoked and necessary Bioconductor libraries were downloaded

```
!apt-get install -y r-base

#Switching to R
%load_ext rpy2.ipython

#Defining path to store R libraries
%%R
r_lib <- "/content/drive/MyDrive/Rlibs" #Define path for R library
dir.create(r_lib, showWarnings = FALSE, recursive = TRUE) #Creating
directory
.libPaths(r_lib) #Setting library path
print(.libPaths()) #Confirmation

#Installing Bioconductor and needed packages
%%R
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("DESeq2")
BiocManager::install("AnnotationDbi")
BiocManager::install("edgeR")
BiocManager::install("GenomicFeatures")
BiocManager::install("pheatmap")
if (!requireNamespace("org.Hs.eg.db", quietly = TRUE)) {
    BiocManager::install("org.Hs.eg.db")
}
install.packages("dplyr")
install.packages("ggplot2")
install.packages("tidyverse")
```

Following this, both GP.csv and metadata.csv were read and QC was performed

Reading both GP.csv and metadata.csv and performing QC

```
#Reading .csv files of counts and metadata
%%R

count_path <- "/content/drive/My Drive/6th sem assignments_Pragathi_PES1UG22BT038/GP_assignments_Pragathi_PES1UG22BT038/GP.csv"
meta_path <- "/content/drive/My Drive/6th sem assignments_Pragathi_PES1UG22BT038/GP_assignments_Pragathi_PES1UG22BT038/metadata.csv"

# Read into R
counts <- read.csv(count_path, row.names=1)
meta <- read.csv(meta_path, row.names=1)

#Check
str(counts)
str(meta)
```

```
'data.frame':   78932 obs. of  7 variables:
 $ ERR3322433_24d       : int  636 2 677 252 68 2 713 1300 526 1303 ...
 $ ERR3322431_18d       : int  637 0 1181 227 47 0 430 2315 679 1015 ...
 $ ERR3322430_12d       : int  457 0 985 248 62 0 342 2049 680 1582 ...
 $ ERR3322435_6d        : int  722 0 973 246 121 1 312 2911 471 1704 ...
 $ ERR3322434_3d        : int  632 0 695 172 78 0 239 2731 280 1009 ...
 $ ERR3322432_1d        : int  318 0 1729 294 87 2 166 1342 479 1453 ...
 $ ERR3322436_Fibroblast: int  1204 0 795 141 64 1 66 2194 264 266 ...
'data.frame':   7 obs. of  2 variables:
 $ day      : int  24 18 12 6 3 1 0
 $ condition: chr  "neuron" "neuron" "neuron" "neuron" ...
```

```
#Quality control to see if column names of count are matching row names of meta
%%R
all(colnames(counts) %in% rownames(meta))
```

```
[1] TRUE
```

```
#Chekcing order of row and column names
%%R
all(colnames(counts)==rownames(meta))
```

```
[1] TRUE
```

Following this, differential expression analysis was initiated. First the dds object (which stands for DESeqDataSet) was created, by running the code as follows:

```
%%R
library(DESeq2)
meta$day_factor <- factor(meta$day)
dds <- DESeqDataSetFromMatrix(countData = counts,
                              colData = meta,
                              design = ~ day_factor)
dds
```

This code loads the DESeq2 library and creates a DESeqDataSet object (dds) for differential expression analysis using count data (counts) and sample metadata (meta), where the experimental design is based on the day variable converted to a factor (day_factor). The resulting dds object contains the counts, sample info, and design formula needed to run DESeq2 analysis. The resulting output is given by:

```
class: DESeqDataSet
dim: 78932 7
metadata(1): version
assays(1): counts
rownames(78932): ENSG00000000003 ENSG00000000005 ... ENSG00000310556
  ENSG00000310557
rowData names(0):
colnames(7): ERR3322433_24d ERR3322431_18d ... ERR3322432_1d
  ERR3322436_Fibroblast
colData names(3): day condition day_factor
```

This output shows that the dds object is a DESeqDataSet containing gene expression counts for 78,932 genes across 7 samples. It includes one assay ("counts") and sample metadata columns: day, condition, and the factor-converted day_factor used for the experimental design.

We then proceed to deal with the leaky expression of genes. It refers to the unintended, low-level expression of a gene that is supposed to be tightly regulated or turned off under certain conditions. It often occurs in inducible gene systems, where the gene is slightly active even without the inducer, which can interfere with experiments requiring strict control over gene expression. We find an eliminate such genes. For this purpose, following code was executed:

```
#Leaky expression of gene : Low expression of mRNA in a place it's not
supposed to express. So remove low read count genes
%%R
# Filter genes: keep those with >=10 counts in at least 2 samples
keep <- rowSums(counts >= 10) >= 2
dds <- dds[keep, ]
cat("Number of genes kept:", sum(keep), "\n")
cat("Total genes before filtering:", nrow(counts), "\n")
```

This code removes genes have less then 10 counts found in at least 2 samples. On running this code, the following output was obtained:

```
Number of genes kept: 18121
Total genes before filtering: 78932
```

Then the design of the experiment was slightly modified keeping fibroblast as reference and a new dds object was created

```
%%R
# Ensure condition is a factor
meta$condition <- factor(meta$condition)

# Set fibroblast as the reference level
meta$condition <- relevel(meta$condition, ref = "fibroblast")
meta$day_factor <- factor(meta$day)
dds <- DESeqDataSetFromMatrix(countData = counts,
```

```
                                colData = meta,
                                design = ~ condition)
```

Then, the differentially expressed genes were extracted and stored in a variable res.

```
#DIff expression
%%R
deg <- DESeq(dds)
```

```
estimating size factors
estimating dispersions
gene-wise dispersion estimates
mean-dispersion relationship
final dispersion estimates
fitting model and testing
```

```
[ ] #STore results in deg variable
    %%R
    res <- results(deg)
```

```
[ ] #Summary stats
    %%R
    summary(res)   #p value <0.1
```

```
out of 40855 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)       : 3150, 7.7%
LFC < 0 (down)     : 1737, 4.3%
outliers [1]       : 120, 0.29%
low counts [2]     : 22595, 55%
(mean count < 6)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

Since the results were for p <0.1, a new variable was created to only take into consideration the DEGs with p < 0.05

```
[ ] #Regenerate results fo p <0.05
    %%R
    res0.05 <- results(deg, alpha = 0.05)
```

```
[ ] %%R
    summary(res0.05)
```

```
out of 40855 with nonzero total read count
adjusted p-value < 0.05
LFC > 0 (up)       : 2298, 5.6%
LFC < 0 (down)     : 1130, 2.8%
outliers [1]       : 120, 0.29%
low counts [2]     : 24092, 59%
(mean count < 9)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

```
[ ] %%R
    vsd <- vst(dds, blind = FALSE)
```

From this we understand, 2298 genes were overall upregulated, 1130 genes were overall downregulated, 120 genes were outliers and roughly 24902 genes appeared in very low counts.

Following this, a column was added to the degs dataframe which consisted of the gene names corresponding to the ENSEMBL ids. The R annotation package org.Hs.eg.db was used for this purpose, which is a detailed reference database on human genes.

```
#Install reference database for naming genes
%%R
if (!requireNamespace("org.Hs.eg.db", quietly = TRUE)) {
    BiocManager::install("org.Hs.eg.db")
}
    library("org.Hs.eg.db")
```

```
Loading required package: AnnotationDbi
```

```
#Adding column with gene symbols

%%R
res0.05.df$ENSEMBL <- rownames(res0.05.df)

# Map ENSEMBL IDs to gene symbols
res0.05.df$Symbol <- mapIds(
  org.Hs.eg.db,
  keys = res0.05.df$ENSEMBL,
  column = "SYMBOL",
  keytype = "ENSEMBL",
  multiVals = "first"
)

# Reorder columns
res0.05.df <- res0.05.df[, c("ENSEMBL", "Symbol", setdiff(colnames(res0.05.df), c("ENSEMBL", "Symbol")))]
```

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

```
#Observing changed dataframe
%%R
str(res0.05.df)
```

```
'data.frame':   78932 obs. of  8 variables:
 $ ENSEMBL       : chr  "ENSG00000000003" "ENSG00000000005" "ENSG00000000419" "ENSG00000000457" ...
 $ Symbol        : chr  "TSPAN6" "TNMD" "DPM1" "SCYL3" ...
 $ baseMean      : num  711.662 0.264 976.572 217.027 74.551 ...
 $ log2FoldChange: num  -1.7326 0.2808 -0.3123 0.0722 -0.4074 ...
 $ lfcSE         : num  0.612 5.398 0.536 0.324 0.609 ...
 $ stat          : num  -2.83 0.052 -0.583 0.223 -0.669 ...
 $ pvalue        : num  0.00465 0.95851 0.56015 0.82391 0.50377 ...
 $ padj          : num  0.0294 NA 0.7177 0.8968 0.6735 ...
```

```
#Saving deg csv file with Symbols column
%%R
write.csv(res0.05.df, file = "/content/drive/My Drive/6th sem assignments_Pragathi_PES1UG22BT038/GP_assignments_Pragathi_PES1UG22BT038/final_test.csv")
```

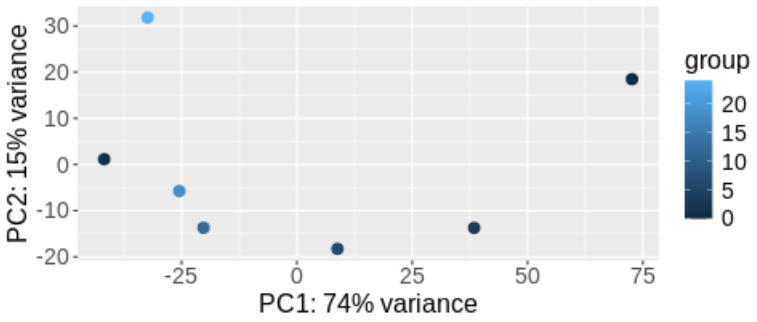
```
#Removing extra ENSEMBL column
%%R
res0.05.df$ENSEMBL <- NULL
write.csv(res0.05.df, file = "/content/drive/My Drive/6th sem assignments_Pragathi_PES1UG22BT038/GP_assignments_Pragathi_PES1UG22BT038/final_test.csv", row.names = TRUE)
```

The file, final_test.csv is the deg file consisting of an additional column with gene symbols, which looks as follows:

	Symbol	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
ENSG00000000003	TSPAN6	711.661553	-1.732637248	0.612146629	-2.830428472	0.004648570408	0.02943917705
ENSG00000000005	TNMD	0.2639225901	0.2808255256	5.397797712	0.05202594475	0.9585080205	NA
ENSG00000000419	DPM1	976.5721885	-0.3123439217	0.5360973785	-0.582625348	0.5601455377	0.7177228566
ENSG00000000457	SCYL3	217.0274691	0.07219596252	0.3244478218	0.2225194859	0.8239095059	0.8968166061
ENSG00000000460	FIRRM	74.55081994	-0.4073569151	0.6092916785	-0.6685745588	0.503766906	0.6734832209
ENSG00000000938	FGR	0.8380069972	-0.9887320421	3.004332932	-0.3291020219	0.7420785761	NA
ENSG00000000971	CFH	304.6500773	1.801257304	0.779348051	2.311236041	0.02081982062	0.07907407689
ENSG00000001036	FUCA2	2168.993312	-0.7001815756	0.6589327523	-1.062599443	0.2879636397	0.4698976523
ENSG00000001084	GCLC	457.9021405	0.270269658	0.4621119697	0.5848575144	0.5586435279	0.7169574518
ENSG00000001167	NFYA	1101.349077	1.636754498	0.3325721747	4.921501625	8.59E-07	3.74E-05
ENSG00000001460	STPG1	68.20542658	-1.506208164	0.61835682	-2.435823647	0.01485792628	0.06376494766

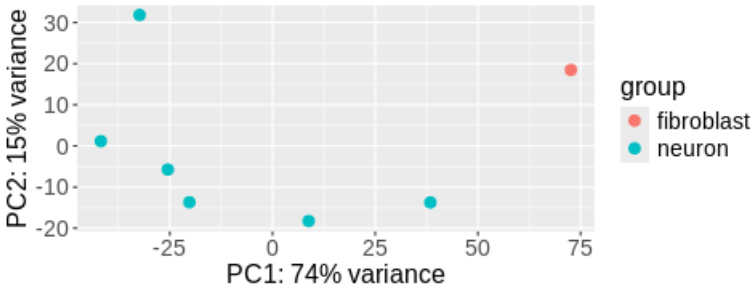
Next, PCA plots were plotted for taking both day as well as condition as interval plots. PCA (Principal Component Analysis) plots in RNA-seq analysis are used to visualize differences and similarities between samples based on their overall gene expression profiles. They reduce complex expression data into a 2D plot, helping you see if samples group by condition, batch, or other factors.

Day:



This PCA plot visualizes RNA-seq samples across two principal components (PC1 and PC2), with PC1 explaining 74% and PC2 15% of the total variance in gene expression. Each point is a sample, colored by day, where lighter blue indicates later time points. The clear separation along PC1 suggests that day (time) is the main driver of gene expression changes, meaning the transcriptome evolves significantly over time. Clustering of samples with similar colors (i.e., similar days) further supports this temporal progression in gene expression patterns.

Condition:



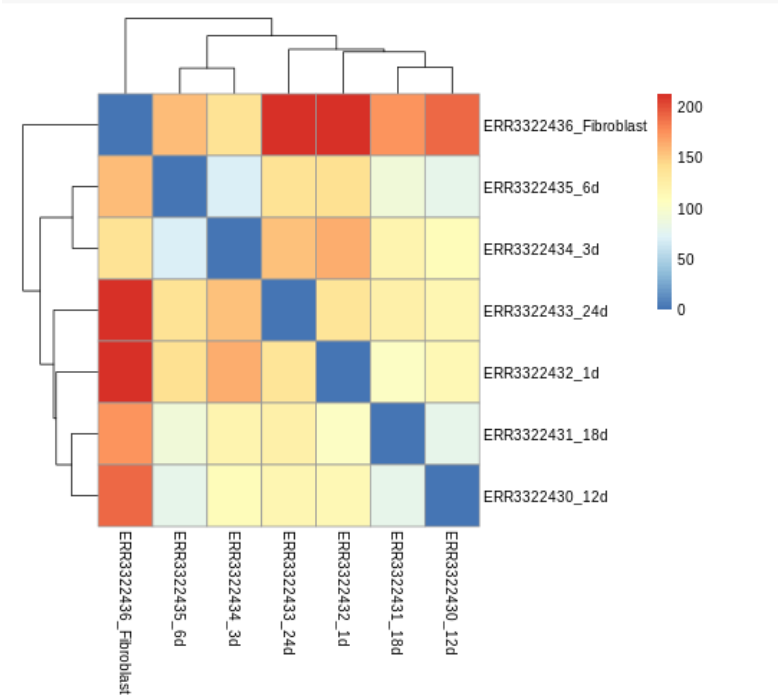
This PCA plot shows how gene expression varies between two cell types: fibroblast and neuron.

Each point represents a sample, colored by condition—red for fibroblast and cyan for neuron. The strong separation along PC1, which explains 74% of the variance, indicates that cell type (condition) is the primary source of variation in gene expression. The tight clustering of neurons and the distinct position of the fibroblast sample suggest consistent transcriptomic profiles within groups and major transcriptional differences between them.

Next, we compute the distance matrix to observe how related the samples are using the code as follows:

```
%%R
#Add sample distance
sampleDists <- dist(t(assay(vsd)))
sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- colnames(vsd)
pheatmap::pheatmap(sampleDistMatrix)
```

We get the following output:



The distance matrix is computed using Euclidean distance between the columns (samples) of the vsd matrix, and the heatmap shows how similar or different the expression profiles are across samples. Samples that cluster together (as shown by the dendrogram) are more similar, suggesting similar biological conditions or time points. For instance, samples like ERR3322434_3d and ERR3322435_6d are closely related, while ERR3322436_Fibroblast is quite distinct, indicating potential biological differences.

We then proceed to do size estimation, followed by dispersion plots. Size estimation and dispersion plotting are done in RNA-seq analysis to normalize for sequencing depth and assess variability across genes. Size factors adjust for differences in library size between samples, allowing accurate comparison of gene expression levels. Dispersion plots estimate the variability (biological + technical noise) of gene counts across replicates, which is crucial for identifying truly differentially expressed genes rather than those varying due to random noise. Together, they ensure reliable statistical modeling for downstream differential expression analysis.

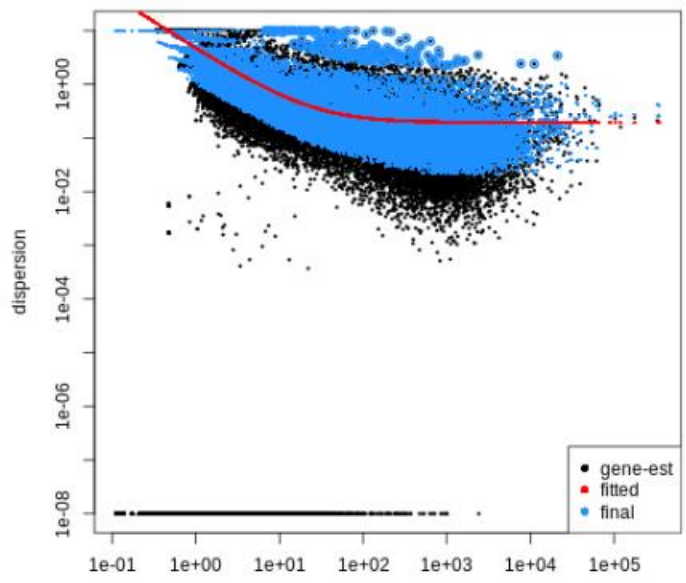
Size estimation: to remove biases

```
[ ] %%R
    sizeFactors(deg)
```

```
        ERR3322433_24d         ERR3322431_18d         ERR3322430_12d
             1.0825685              1.0369194              1.2821276
         ERR3322435_6d          ERR3322434_3d          ERR3322432_1d
             1.1718687              0.8281179              1.1828827
ERR3322436_Fibroblast
             0.6782107
```

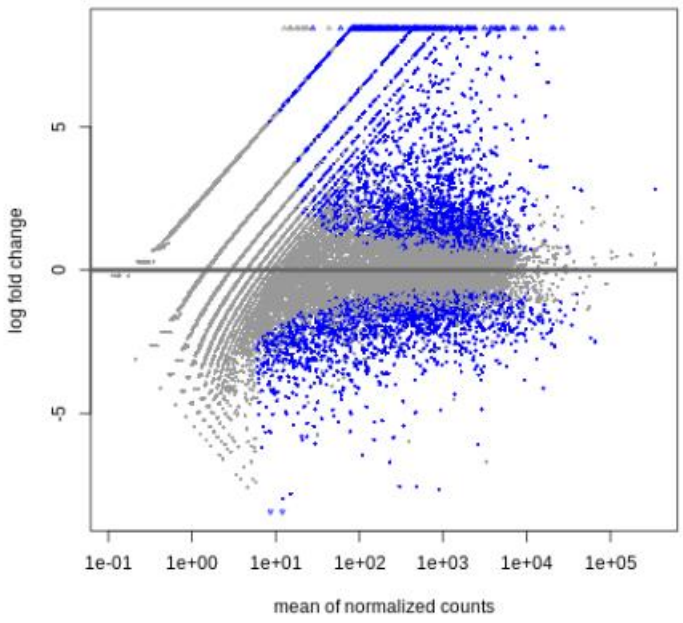
Disperssion plot

```
[ ] %%R
    plotDispEsts(deg)
```



Next we plot the Moving averages plot, using the following code:

```
#MA plot
%%R
library(dplyr)
library(ggplot2)
library(tidyverse)
plotMA(deg)
```



The MA plot visualizes gene expression changes between two conditions, where each point represents a gene. The x-axis shows the mean of normalized counts (expression level), and the y-axis shows the log2 fold change in expression. Blue points represent genes with statistically significant changes, while gray points are not significant. Genes above or below the horizontal line (log fold change = 0) are up- or downregulated, respectively, and triangle symbols indicate extreme values beyond plot limits.

We then proceed to look at the top 30 genes being expressed as a part of all our smaples, using the following code:

```
#getting idea about 30 best genes
%%R
best_genes <- res0.05.df %>%
 arrange(padj) %>%
 head(30)
best_genes
```

The top 30 genes are determined based on the adjusted p-value (padj), which measures the statistical significance of differential expression after correcting for multiple testing. By arranging the genes in ascending order of padj, the code selects those with the strongest evidence of differential expression—i.e., the 30 genes most likely to be truly differentially expressed between conditions.