RNA-Sequencing analysis with R

Differential expression and functional analysis of biological data

Maria Belen Rabaglino

Table of contents

INTRODUCTION **1**

THE EXPERIMENT **1**

INPUT DATA **2**

QUALITY ASSESSMENT **4**

- Pre-filtering 5
- Normalization through variance stabilization transformation 5
- Hierarchical clustering and Heatmap of the sample-to-sample distances 5
- Principal component analysis 6

DIFFERENTIALLY EXPRESSION ANALYSIS **8**

- The DESeqDataSet 8
- Identification of differentially expressed genes 8
- Exporting the results 12

FUNCTIONAL ENRICHMENT ANALYSIS **12**

- Overrepresentation analysis 13
- Enrichment analysis of Gene Ontology terms (biological processes) 14
- Enrichment analysis of KEGG pathways 16
- Visualization of the ORA results 17

CONCLUSION **21**

REFERENCES **21**

Introduction

This tutorial will show the main steps followed to analyze data generated through the application of RNA-seq technology. The data was obtained from a public database, and the analysis will be performed using packages from Bioconductor (https://www.bioconductor.org/). This is an open software for the analysis of high-throughput genomic data through the R programming language. RNA-seq data can be analyzed using different approaches, and the election depends mainly on the design and goal(s) of the experiment. Here, we will show the usual pipeline applied in most of them. First, we will perform an unsupervised analysis of the data, to evaluate the quality of the samples and the similarities between them. Afterwards, the differentially expressed genes between the groups of study will be determined. Finally, a functional analysis of those genes will be performed, in order to obtain biological insight about them. This last analysis helps not only to confirm or reject the experimental hypothesis but also to generate new ones.
The quality assessment and differential expression analysis will be performed with the DESeq2 package [1], one of the most commonly employed (together with edgeR) to analyze RNA-seq data in R. This package has an excellent tutorial with detailed information about the methods provided by DESeq2 to test for differential expression in RNAseq data:
https://bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html
The functional analysis will be performed with the clusterProfiler pakage [2]. This package has a complete vignette that can be found in: https://yulab-smu.github.io/clusterProfiler-book/.
In the present tutorial, we will show the standard workflow of data analysis. However, more information and details about each step can be found in the corresponding tutorials of each package.

The experiment

For this tutorial, we will use the data generated from a study performed in healthy women from Scandinavia [3], on where samples were obtained from the endometrium, which is the inner uterine lining. The goal was to elucidate endometrial maturation mechanisms at transcriptomic level and identify novel robust biomarker candidates for endometrial receptivity.
To understand this goal, let's review briefly the basic biology of the menstrual cycle. This cycle comprehends natural changes occurring in the ovary and uterus that lead to ovulation and pregnancy, if the ovule is fertilized. The cycle has an average duration of 28 days and takes place in the woman and primates while other mammals present an "estrous cycle".
In the menstrual cycle, the uterus, or more precisely, the endometrium, undergoes through two phases: the proliferative phase, in the first 14 days, and the secretory phase in the other 14 days. In the proliferative phase, a follicle, with an egg or ovule, growths in the ovary, and ovulation occurs at around day 14 after menstruation, because of a peak in the concentration of the "luteinizing hormone", or LH. This hormone induces ovulation and then transforms the follicle (without the ovule now) in a structure called "corpus luteum", which secretes progesterone, the hormone responsible for maintaining the pregnancy. Thus, after ovulation, it comes the secretory phase, where the endometrium, under the influence of progesterone, thicken to nest an embryo. If there is no pregnancy, the thickened part of the endometrium is released around 14 days after ovulation (menstruation), and the cycle reinitiates.
In this study, two paired samples were obtained from the endometrium of each patient, one during the early secretory phase (2 days after the LH peak) and the other in the mid-secretory phase (8 days after the LH peak), which is the day when the embryo would nest. There were 10 women involved in the experiment, and thus 20 samples in total (n=10 for LH +2 and n=10 for LH +8). Therefore, we will determine how the endometrial transcriptome changed during those 6 days, i.e., how many genes increased or decreased in expression and the biologic functions related with them.

Input data

The analysis requires two type of data (two files): the matrix of raw counts files and the experimental design file, available in the course's website in DTU inside.
That data was downloaded from a public genomic data repository: the Gene Expression Omnibus database (GEO, https://www.ncbi.nlm.nih.gov/geo/), which stores records related with array- and sequence-based data. The accession number for this particular study is GSE98386 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE98386).

Download both files (DataRaw.txt and SampleInfo.txt) and save them in a new folder in your computer. Then, open RStudio and set the working directory in that new folder (go to Session > Set Working Directory > Choose Directory).

The matrix of raw counts is a table where the features, or transcripts IDs, are listed in the rows and the samples in the experiment in the columns. Thus, each cell contain the number of transcript counts for a given transcript and sample.
Let's explore the matrix of raw counts (DataRaw.txt file):

```
CountTable = read.table( "DataRaw.txt", header=TRUE, row.names=1 )
dim(CountTable)
## [1] 63677    20
```

The table has 63677 rows and 20 columns. Let's take a look at the table…

```
View(CountTable)
```

	Healthy_NOT1210LH2E	Healthy_NOT1213LH2E	Healthy_NOTNV01LH2E	Healthy_NOTNV02LH2E	Healthy_NOTNV03LH2E	Healthy_NOTNV04LH2E	Healthy_NOTNV11LH2E
ENSG00000244656	0	0	0	0	0	0	0
ENSG00000127720	303	404	447	431	539	466	450
ENSG00000224440	0	0	0	0	0	0	0

Above are shown the first 3 rows and 7 samples. Genes can be identified using different IDs (For example: official symbol, NCBI ID or Entrez ID, RefSeq ID, etc).
We can see that the IDs for these transcripts begin with the letters "ENSG" and then numbers. These types of IDs are given by "ENSEMBL" (http://www.ensembl.org/index.html), a genome browser for vertebrate genomes. Each ENSEMBL ID is unique, and they can be related to protein-coding genes but also to other types of genes or even genes without a known name (for example, the first one, ENSG00000244656, belongs to a novel miRNA, with no name yet, while the second one, ENSG00000127720, is related to the gene METTL25, or methyltransferase like 25).
We can see also, that each sample has been identified as "Healthy_NOT…..". Thus, all the samples shown above have 0 counts, or no expression, of the first and third ENSEMBL ID, but they have around 400 counts for the second ENSEMBL ID, or the METTL25 gene.

The experimental design is a file that specifies the characteristics of each sample (minimally, to what group each one belongs to). This information should be stated in a table where the samples IDs are listed in the rows and each variable of interest (like Group) in the columns. For this tutorial, this file is the "SampleInfo.txt"

Let's load the file into the R environment and take a look at it:

```
coldesign=read.table( "SampleInfo.txt", header=TRUE, row.names=1 )
colnames(coldesign)
## [1] "Group"     "Patient"   "DayLHplus"
```

```
View(coldesign)
```

	Group	Patient	DayLHplus
Healthy_NOT1210LH2E	EarlySecretory	NOT1210	2
Healthy_NOT1213LH2E	EarlySecretory	NOT1213	2
Healthy_NOTNV01LH2E	EarlySecretory	NOTNV01	2
Healthy_NOTNV02LH2E	EarlySecretory	NOTNV02	2
Healthy_NOTNV03LH2E	EarlySecretory	NOTNV03	2
Healthy_NOTNV04LH2E	EarlySecretory	NOTNV04	2
Healthy_NOTNV11LH2E	EarlySecretory	NOTNV11	2
Healthy_NOTNV12LH2E	EarlySecretory	NOTNV12	2
Healthy_NOTNV13LH2E	EarlySecretory	NOTNV13	2
Healthy_NOTNV15LH2E	EarlySecretory	NOTNV15	2
Healthy_NOT1210LH8E	MidSecretory	NOT1210	8
Healthy_NOT1213LH8E	MidSecretory	NOT1213	8
Healthy_NOTNV01LH8E	MidSecretory	NOTNV01	8
Healthy_NOTNV02LH8E	MidSecretory	NOTNV02	8
Healthy_NOTNV03LH8E	MidSecretory	NOTNV03	8
Healthy_NOTNV04LH8E	MidSecretory	NOTNV04	8
Healthy_NOTNV11LH8E	MidSecretory	NOTNV11	8
Healthy_NOTNV12LH8E	MidSecretory	NOTNV12	8
Healthy_NOTNV13LH8E	MidSecretory	NOTNV13	8
Healthy_NOTNV15LH8E	MidSecretory	NOTNV15	8

The file, now named “coldesign”, has the information for three variables: Group (early or mid-secretory period), patient ID (notice that each ID in the early group is repeated in the mid-secretory group) and finally the day after the LH peak (2 or 8).

Before moving forward, take a look at the codes employed to input both files. The terms “header=TRUE” and “row.names=1” mean that the first row of the table will be used as column names and the first column of the table as row names.
Thus, the sample IDs are the column names of the matrix of raw counts (CountTable) and the row names of the experimental design (coldesign). The IDs should be the same and in the same order, (the name of column 1 in ‘CountTable’ should match the name of row 1 in ‘coldesign’ and so on).
To verify this, we can use the following command that should return TRUE

```
all(colnames(CountTable)==rownames(coldesign))
## [1] TRUE
```

Hint: if this command above returns `## [1] FALSE` , it is because the columns of the matrix of raw counts ant the rows of the experimental design are NOT in the same order. If this is the case, we can re-create the ‘CountTable’ object (matrix of raw counts) but placing the columns in the same order than the rows in the coldesign object (experimental design) by running:

```
CountTable <- CountTable[, rownames(coldesign)]
```

Now, we verify again is the order is the same.

```
all(colnames(CountTable)==rownames(coldesign))
## [1] TRUE
```

Quality assessment

Data quality assessment and quality control (i.e. the removal of outliers' samples) are essential steps that should be performed before testing for differential expression of genes between the groups of study. These steps do not preclude the quality control methods previously applied to the raw files (FASTQ) or to the alignment results. Rather, a data quality assessment is performed with the following general purposes:

- To evaluate similarities between samples according to their transcriptomic profile.
- To determine which condition (if any) is driving the transcriptomic variations among the samples.
- To identify that of those sample(s) that have abnormal transcriptomic profile compared to the other samples, that can affect the differentially expression test.

Data quality evaluation is performed usually through methods that determine similarities between samples and output the results in a visual way. These types of methods are known also as unsupervised learning, since they can find patterns in a data set without pre-existing labels, i. e., without the experimental information of each sample.
In this tutorial, we will use the DESeq2 package to run these methods. However, before that, we need to prepare the data by filtering the non-expressed transcripts and normalizing the counts. Afterwards, we will use two main methods of unsupervised learning: hierarchical clustering of the samples and principal component analysis.

Before starting with the analyses, we download and install the Bioconductor package with the following command:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("DESeq2")
## Bioconductor version 3.9 (BiocManager 1.30.4), R 3.6.0 (2019-04-26)
## Installing package(s) 'DESeq2
.....
library(DESeq2)
```

In addition, we need two packages from the CRAN repository, in order to create the graphs of the quality assessment methods. These packages can be installed with the following commands:

```
install.packages("RColorBrewer")
install.packages("pheatmap")
```

In RStudio, these packages also can be downloaded and installed by going to Tools > Install Packages, and then typing the name of the package.

Then we load the packages:

```
library("RColorBrewer")
library("pheatmap")
```

Pre-filtering
It is advisable to filter the data to remove those transcripts that have low or no counts for assessment of the data quality. Pre-filtering the data is not necessary for the statistical analysis because DESeq2 automatically applies an independent filtering by default using the mean of normalized counts, in order to increase the detection power at the same experiment-wide type I error, (which is measured in terms of false discovery rate, as explained below).

Here, we will keep all the transcripts, or rows, that have at least 1 count across all samples.

```
keep = apply(CountTable, 1, function(row) sum(row==0 ) < nrow(coldesign))
table(keep)
## keep
## FALSE  TRUE
## 27403 36274

data.filt <- CountTable[keep,]
```

With the command `table(keep)` we can see how many transcripts have less than 1 count (under FALSE) and how many have more than 1 read (under TRUE). Thus, 43% of the transcripts in this data (27403 out of 63677) have no counts, or *they were not expressed* in the endometrium. Now, the data.filt object will retain the 36274 expressed transcripts.

Normalization through variance stabilization transformation
The DESeq package requires the raw counts to test for differential expression, since it uses discrete distribution. However, quality assessment methods should be applied on transformed versions of the count data.
One obvious transformation is the logarithmic. Here, we will apply a 'variance stabilizing transformation' method to our data through the *varianceStabilizingTransformation* function. This function produces transformed data, on the log2 scale, which has been normalized with respect to library size or other normalization factors. The transformation removes the dependence of the variance on the mean, particularly the high variance of the logarithm of count data when the mean is low. This function is applied to a matrix object, which in our case, has been filtered for the non-expressed transcripts.

```
vsd <- varianceStabilizingTransformation(as.matrix(data.filt),
blind=TRUE, fitType="local")
```

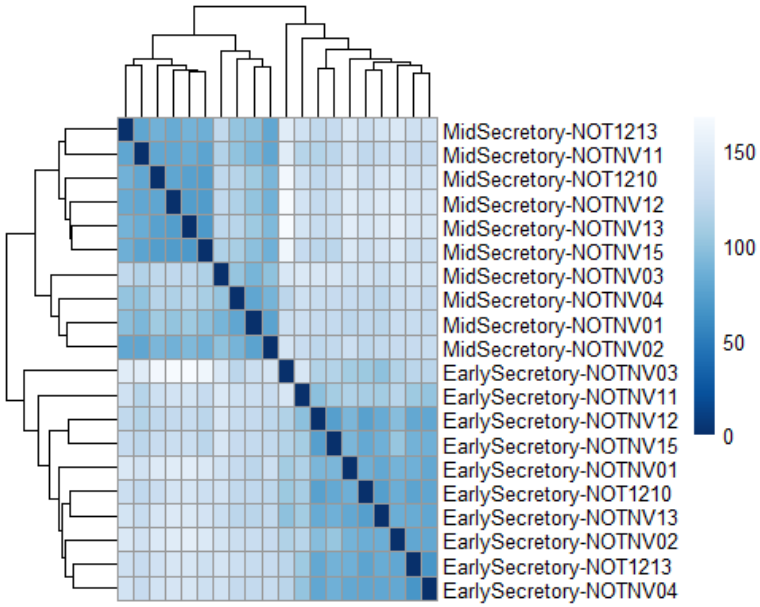
Note that the function has two arguments `blind=TRUE` and `fitType="local"`. When `blind=TRUE`, the transformation is blind to the sample information specified by the experimental design formula. Thus, samples are compared in an unbiased manner, as should be done for assessing their quality. The argument `fitType="local"` refers to the type of fitting of dispersions to the mean intensity.

Hierarchical clustering and Heatmap of the sample-to-sample distances
Hierarchical clustering is an algorithm that groups similar objects into clusters, where samples within each cluster are broadly similar to each other. First, we need to compute the sample-to-sample distances according to the transformed matrix of raw counts, through the function *dist*. This function by default applies the Euclidean method for distance measuring.

```
sampleDists <- dist(t(vsd))
```

The function is applied to the transposed matrix, which is output with `t(vsd)` .
Then, we create a heatmap of the distance matrix to have a graphical representation of similarities and dissimilarities between samples.

```
sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste(coldesign$Group, coldesign$Patient,
sep="-")
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors)
```



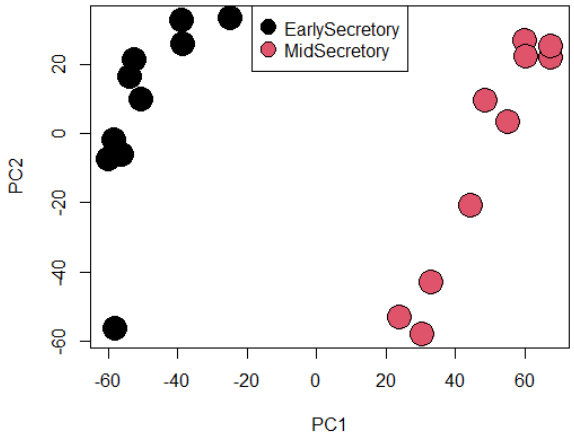
To construct the heat map, we asked to use the information of both the group and patient from each sample (**`paste(coldesign$Group, coldesign$Patient, sep="-")`**). Before running this analysis, we could have thought that samples from the same patient would have a similar transcriptomic profile. However, this graph is showing that samples are clustering according to the group (if endometrial samples were obtained in the early or mid-secretory phase) and thus the effect of paired samples is not the one determining the transcriptomic profile.

Principal component analysis

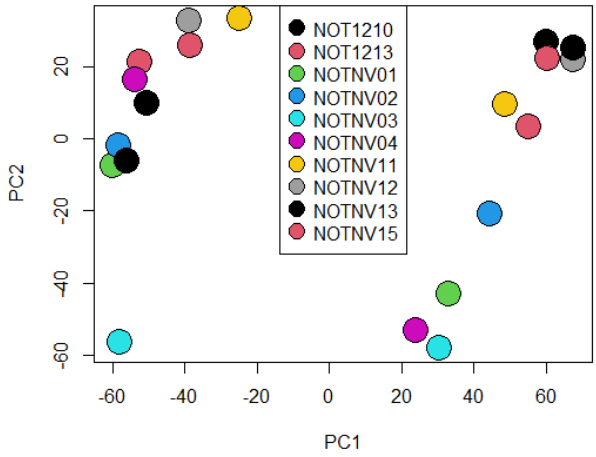
Briefly, this analysis is a statistical procedure where the transformed matrix of raw counts is converted to principal components variables, being the first component the one that accounts for the largest variability in the data.
This analysis and the plot of the first and second component in a 2D plane is performed with the *prcomp* function. Samples can be colored according to any condition in the experimental design. Thus, we will color the samples according to the group (early or mid-secretory endometrium) and according to the patient. This type of plot is useful for visualizing the overall effect of experimental covariates and batch effects.

```
pca <- prcomp(t(vsd)); pca2 <- as.data.frame(pca$x)
var <- as.factor(coldesign$Group)
ColGr <- unclass(var)
```

```
plot(pca2$PC1, pca2$PC2,  cex=3, pch=21, bg=ColGr, col="black",
xlab="PC1", ylab="PC2")
legend("top", unique(var), pch=21, pt.cex=2, pt.bg=unique(ColGr),
horiz=F, cex=1, box.lty=1)
```




```
pca <- prcomp(t(vsd)); pca2 <- as.data.frame(pca$x)
var <- as.factor(coldesign$Patient)
ColGr <- unclass(var)
plot(pca2$PC1, pca2$PC2,  cex=3, pch=21, bg=ColGr, col="black",
xlab="PC1", ylab="PC2")
legend("top", unique(var), pch=21, pt.cex=2, pt.bg=unique(ColGr),
horiz=F, cex=1, box.lty=1)
```



We can see that samples are segregated in the first component according to the group to which they belong, and not according to the patient. Therefore, the strongest source of transcriptomic variation among the samples is the period when they were obtained.
From these results, we can observe that all samples have good quality, and the transcriptomic profile depends on the endometrial period. Consequently, we can infer a large number of differentially expressed genes between both groups.
This type of sample behavior is the ideal, but not it is not currently to observe in all the experiments. Remember that the fact of samples from different groups clustering together does not mean that the whole experiment is useless. That is why we proceed with a supervised method, such as the detection of differentially expressed genes.

Differentially expression analysis

The goal of the analysis to identify differentially expressed genes will depend on the experimental design. For a case-control study, we want to identify those genes significantly changing in expression in the group of interest compared to a control group. When the experiment consists of more than two conditions, we are probably interested in performing contrasts between the conditions and, in addition, to determine if there are interactions between them. For time-course experiments, the analysis will depend if there is one condition, where the genes of interest are those that change over time respect the first-time point (baseline), or two conditions, where the differentially expressed genes are those that change over time respect the other condition.
Here, we will use a simple case-control experimental design with the aim to detect those genes whose expression levels are significantly higher or lower in one group (case) compared to the other (control). Genes with significant higher or lower expression levels than the control group are called up-regulated or down-regulated genes, respectively.

The DESeqDataSet

The DESeq2 package uses an object class, called "DESeqDataSet" to store the read counts and the intermediate estimated quantities during the statistical analysis. This object can be constructed in different ways according to the pipeline employed to generate the raw counts for each transcript. For the example presented here, we already have the raw counts in a matrix, where each row is a transcript and each column is a sample ID.
First, we need to transform the variable(s) of interest in the experimental design table to a factor, so we can do the statistical contrast

```
coldesign$Group <- as.factor(coldesign$Group)
coldesign$Patient <- as.factor(coldesign$Patient)
```

Next, the DESeqDataSet object will be constructed with the following command:

```
data <- DESeqDataSetFromMatrix(countData = CountTable, colData =
coldesign, design= ~ Patient + Group)
```

In this command, we need to specify the matrix of raw counts after 'countData' (CountTable), the experimental design after 'coldData' (coldesign) and finally, the model to use after 'design'. For example, if the name of the matrix of raw counts is other than CountTable (let's say, Table), then the command should be `countData = Table`.
The model after 'design', for our example, is ~Patient + Group. In the model, the effect of interest should go at the end of the formula while the batch effect to control for, should go before the main effect.
Here, we want to account for the differences between groups, or between mid and early secretory period, but controlling for the patient effect, since these were paired samples.

Identification of differentially expressed genes

The steps followed by DESeq2 to perform the statistical analysis are wrapped in one function: *DESeq*. Details about each step can be found in Love et al, [1]. Briefly, the differential expression analysis uses a generalized linear model (GLM) where the counts for each gene and each sample are modeled using a negative binomial distribution with a fitted mean and a gene specific dispersion parameter. Then, the Wald test statistic is applied to test for significance of the GLM coefficients.

P-values correction: The resulting p-values are adjusted with the Benjamini-Hochberg method. Controlling for p-value is necessary when performing this type of analysis because of the multiple comparisons effect, which increases the likelihood of type I errors, or detecting false positives (a

gene will be classified as differentially expressed when it is not). For example, the probability that a gene is false positive is 0.05 for that single gene. However, if 100 genes are tested, the probability to identify falsely a differentially expressed gene is 0.994 and the results will have at least one gene that is false positive. Therefore, any p-values that could be derived from tests on the individual transcript should be interpreted in the context of thousands of transcripts that are being tested simultaneously. Thus, p-values need to be modified through methods such as Benjamini-Hochberg, which controls for the false discovery rate (FDR), in order to have sensible error rate control for all of the tests in parallel.

To apply the *DESeq* function, we will use the DeSeqDataset object. Then, we run the following command:

```
dds <- DESeq(dds)
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
```

To look at the names of the estimated effects (coefficients):

```
resultsNames(dds)
##  [1] "Intercept"
##  [2] "Patient_NOT1213_vs_NOT1210"
##  [3] "Patient_NOTNV01_vs_NOT1210"
##  [4] "Patient_NOTNV02_vs_NOT1210"
##  [5] "Patient_NOTNV03_vs_NOT1210"
##  [6] "Patient_NOTNV04_vs_NOT1210"
##  [7] "Patient_NOTNV11_vs_NOT1210"
##  [8] "Patient_NOTNV12_vs_NOT1210"
##  [9] "Patient_NOTNV13_vs_NOT1210"
## [10] "Patient_NOTNV15_vs_NOT1210"
## [11] "Group_MidSecretory_vs_EarlySecretory"
```

As the variable “Patient” is in the model, the effect of each patient compared to another is tested. However, we are interested only in the effect of the moment of sample collection (mid secretory or early secretory phase).
In order to obtain the raw and adjusted p-values for the coefficient of interest, we run the following command: (if we are interested in another contrast rather than the last one, it should be specified after `name=`)

```
res <- results(dds, name="Group_MidSecretory_vs_EarlySecretory")

head(res)
## log2 fold change (MLE): Group MidSecretory vs EarlySecretory
## Wald test p-value: Group MidSecretory vs EarlySecretory
## DataFrame with 6 rows and 6 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
padj
```

```
                  <numeric>      <numeric> <numeric> <numeric> <numeric>
<numeric>
ENSG00000244656   0.000000             NA        NA        NA        NA
NA
ENSG00000127720 389.736308      -0.137507  0.095321 -1.442572  0.149141
0.252296
ENSG00000224440   0.000000             NA        NA        NA        NA
```

With `head(res)` the first rows of the results are shown. These rows are not ordered by p-value. Also, notice that some adjusted p-values are set to NA. This could happen because of one of the following reasons:
-The gene or transcript has zero counts for all the samples (this is not the case, since we pre-filtered the non-expressed transcripts),
-One sample was an extreme outlier, i.e., the sample has a number of counts for a given transcript too different from the other samples. These outlier counts are detected by Cook's distance.
-The transcript was filtered out by the automatic independent filtering applied by DESeq in the *results* function, because the mean normalized count for that transcript was too low.

In order to explore the results from the differential expression analysis, we can use the following commands:

```
summary(res)
## out of 36274 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 5809, 16%
## LFC < 0 (down)     : 5240, 14%
## outliers [1]       : 0, 0%
## low counts [2]     : 12831, 35%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

The output shows the number of genes that were significant with an adjusted p-value less than 0.1 and a log fold change (LFC) higher or lower than zero. The LFC is the logarithm (in base 2) of the fold change (FC), which is a measure describing how much a quantity changes between two values. Thus, a FC of 2 (the same than a LFC of 1) means that the quantity of one value is the double of the other value.
The proportion of up-regulated or down-regulated genes (according to adjusted p-value, or FDR) is very high, but this is expected looking at the results from the quality assessment analyses.
In addition, we can see that there were no outliers' genes, but a high proportion of them (37%) has low counts.

Definition of a 'differentially expressed gene' (DEG): The criteria to set up the threshold to define a significant DEG is variable. It is generally accepted that DEGs should be those with adjusted p-value, or FDR <0.05. However, some researchers use only FDR <0.1, while others combine FDR with fold change (FC), such as FDR<0.05 and FC higher or lower than 2 or -2, respectively. The decision will depend not only on the total number of genes with low p-value but also on objective of the experiment.
For example, if the detection of DEGs will be followed by functional analysis to characterize their biological functions, a higher number of DEGs is preferable. However, if the researcher wants to identify only those genes with strong differences between groups, then he/she will be interested

not only in the significant ones according to FDR but also in those with very high positive or negative FC.

At this point, we know already that the two groups defined in the experiment used for this tutorial (early and mid-secretory endometrium) have a strongly different transcriptomic profile, even when samples were obtained from the same person. Thus, we are going to use more stringent criteria to define the DEGs. These DEGs will next be employed for the functional analysis.

We will define DEGs as those with FDR <0.01 and FC higher than 2 or lower than -2. Genes with FC > 2 are the up-regulated DEGs (with higher expression in the mid-secretory phase compared to the early phase); while genes with FC < -2 are down-regulated DEGs (the expression in the mid-secretory phase is lower than in the early phase).
Notice that in the output, the fold changes are in logarithmic scale with base 2 (log2FC or LFC). Thus, a FC > 2 is the same than LFC > 1.

Now, we are going to generate the list for the up- or down- regulated genes, in order to use these lists for the next analysis.

```
results <- as.data.frame(res)

upreg <- subset(results, padj<0.01 & log2FoldChange> 1); dim(upreg)
## [1] 2042    6
downreg<- subset(res, padj<0.01 & log2FoldChange< -1); dim(downreg)
## [1] 1797    6
```

The number of up-regulated or down-regulated DEGs is 2046 and 1801, respectively.

We can also identify those genes with the highest or lowest FC on each list:

```
upreg[which.max(upreg$log2FoldChange),]
##                  baseMean log2FoldChange     lfcSE     stat      pvalue
## ENSG00000267380 336.5881       10.03858 0.7251048 13.84432 1.37684e-43
##                          padj
## ENSG00000267380 1.900371e-41
downreg[which.min(downreg$log2FoldChange),]
##                  baseMean log2FoldChange    lfcSE      stat
pvalue
## ENSG00000231589 64.51067      -7.570647 1.032406 -7.333013 2.250345e-
13
##                         padj
## ENSG00000231589 3.62777e-12
```

Thus, the transcripts ENSG00000267380 and ENSG00000231589 are the most and least expressed ones, respectively, in the mid-secretory endometrium compared to the early secretory period.

Exporting the results
If you want to export the results, to save them as excel files for example, the following command can be used:

```
write.table(upreg, "UpregulatedDEGs.txt", sep=",", col.names=NA)
```

This command will create a text file in the working directory, named "UpregulatedDEGs", which is the table generated above with `upreg<-subset(results, padj<0.01 & log2FoldChange> 1)`.

With the command:

```
write.table(results, "Results.txt", sep="\t", col.names=NA)
```

We can export the full table of results.

These files can be opened with excel, but remember that each field is separated by tabs.

Functional enrichment analysis

The goal of the functional analysis is to gain greater biological insight on the DEGs, which could validate the hypotheses or generate new ones. Of course, results from these analyses should be interpreted in the context of the experiment.
The biological roles of the genes have been organized in 'gene sets', which are collections of genes that are functionally related. These main sets are Gene Ontology and KEGG pathways. A gene that can be classified in these sets is an Annotated gene.

- *Gene Ontology* (http://geneontology.org/) is a knowledgebase containing the largest information that describes gene function, organized in three major categories:
  - molecular functions (MF): biochemical activities of gene products,
  - cellular components (CC): location in the cell of the gene product
  - biological processes (BP): pathways and larger processes made up of the activities of multiple gene products.
- *KEGG pathway* https://www.genome.jp/kegg/pathway.html is a collection of manually drawn pathway maps representing molecular interactions and reaction networks between gene products. These pathways cover a wide range of biochemical processes, divided seven major categories.

Thus, the functional analysis consists in the identification of the relevant and significant 'gene ontology' terms or KEGG pathways associated with the genes of interest.
There are three main types of analyses for this purpose, briefly overviewed below:

- Over-representation analysis (ORA): This is a widely used approach, and the one that we will employ in this tutorial. Genes are grouped in categories that have been consistently named (controlled vocabulary) based on how the gene has been functionally annotated. Overrepresented categories are determined according to the probability of having the observed proportion of genes associated with a specific category in our gene list (up or down-regulated DEGs) based on the proportion of genes associated with the same category in the background set (gene categorizations for the appropriate organism). The statistical test employed is the hypergeometric test.
  In other words, the enrichment analysis is indeed to compare the Annotation composition in your gene list to that of a population of background genes.
- Gene Set Enrichment Analysis (GSEA): This approach does not need a list of DEGs, but all the genes in our data and the information about the groups of study. GSEA aggregates the per gene statistics across genes within a gene set, therefore, making it possible to detect situations where all genes, in a predefined set, change in a small but coordinated way (many relevant phenotypic differences could be manifested by small but consistent changes in a set of genes).

- Gene-network analysis: The principle behind a gene network is to determine gene-gene associations. Genes can be connected because they are part of similar biological pathway or controlled by the same transcription factor, or their products interact, for example. Functionally related genes tend to have similar expression levels (often referred as the guilt-by-association principle). This idea provides the support to use the gene expression data to infer networks, which are called gene co-expression networks (GCN). More details about this method will be given in the next tutorial about WGCNA (weighted GCN analysis).

In the present tutorial, we are going to use a Bioconductor package for ORA. However, take into account that there are several excellent and free tools that can be used for this purpose. Most of these tools are online and/or they provide a software to download.
Other tools, such as Ingenuity Pathway Analysis, or IPA, provides more options that only functional analysis. For example, it helps to identify upstream or downstream regulators of the genes of interest. IPA requires a license, but it is possible to get a free trial.

Those tools that perform ORA need the specification of the organism and the type of ID for each gene (for our example, human and ENSEMBL ID) and the input of a list of genes (for example, up-regulated DEGs).
The background list could be customized (i. e., the user input the list) or it can be selected from a list of default background genes pre-defined on each tool. These default population backgrounds are usually the corresponding genome-wide genes with at least one Annotation in the analyzing categories.

Some of the free tools employed for functional analysis are:

- DAVID: https://david.ncifcrf.gov/summary.jsp
- Webgestalt: http://www.webgestalt.org/ (it can also be used for GSEA)
- Panther: http://pantherdb.org/ (it can also be used for GSEA)
- g:Profiler: http://biit.cs.ut.ee/gprofiler/gost
- There is another tool, REVIGO (http://revigo.irb.hr/) that is useful to summarize and visualize Gene Ontology terms, since it removes the redundant ones.
- Gene set enrichment analysis can be run using the software provided by the Broad Institute (UCSan Diego (http://software.broadinstitute.org/gsea/index.jsp). This tool requires the expression dataset (the matrix of gene counts) and the phenotype label (to which group each sample belongs to). The software has a very well explained tutorial to run this analysis. (http://software.broadinstitute.org/gsea/doc/desktop_tutorial.jsp).

As mentioned above, in this tutorial we will apply ORA to the list of up and down-regulated DEGs. We will do so using the package “clusterProfiler” from Bioconductor. This package can be used to perform GSEA as well.

Overrepresentation analysis

First, we need to download and install the clusterProfiler package from Bioconductor.
In addition, we need to download and install the genome annotation for the specie of interest, which is the human, in our case. However, if working with another specie, the corresponding package should be installed. Genome wide annotations packages in Bioconductor are for 19 species, and can be found in:
https://bioconductor.org/packages/release/BiocViews.html#___OrgDb.
The one for human is ‘org.Hs.eg.db’.

To install both packages from Bioconductor we do:

```
BiocManager::install("org.Hs.eg.db")
## Bioconductor version 3.9 (BiocManager 1.30.4), R 3.6.0 (2019-04-26)
## Installing package(s) 'org.Hs.eg.db'
## installing the source package 'org.Hs.eg.db'

BiocManager::install("clusterProfiler")
## Bioconductor version 3.9 (BiocManager 1.30.4), R 3.6.0 (2019-04-26)
## Installing package(s) 'clusterProfiler'
## package 'clusterProfiler' successfully unpacked and MD5 sums checked
```

Afterward, we call both packages

```
library(org.Hs.eg.db)
## Loading required package: AnnotationDbi
## Warning: package 'AnnotationDbi' was built under R version 3.6.1
##
library(clusterProfiler)
## Registered S3 method overwritten by 'enrichplot':
##   method               from
##   fortify.enrichResult DOSE
## clusterProfiler v3.12.0  For help:
https://guangchuangyu.github.io/software/clusterProfiler

The following object is masked from ‘package:DelayedArray’:
    simplify
```

The genes of interest are the up-regulated or down-regulated DEGs. Thus, we want to elucidate the biological processes and pathways that are stimulated or inhibited, respectively, in the mid-secretory endometrium compared to the early secretory.
Let’s apply the ORA in the up-regulated DEGs.

```
sigUP<- as.character(row.names(upreg))
```

Enrichment analysis of Gene Ontology terms (biological processes)
To perform this enrichment analysis, we use the function *enrichGO*.

```
ego <- enrichGO(gene = sigUP,
                keyType = "ENSEMBL",
                OrgDb = org.Hs.eg.db,
                ont = "BP",
                pAdjustMethod = "BH",
                qvalueCutoff = 0.05,
                readable = TRUE)
```

Before moving forward, let’s understand the options from this function.
After `gene =` we need to specify the lists of genes of interest.
The parameter `keyType =` is the type of gene ID. Any gene ID type that is supported in “OrgDb” can be directly used in GO analyses. To look at the key types supported for each organism, we can use:

```
keytypes(org.Hs.eg.db)
##  [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"
##  [5] "ENSEMBLTRANS" "ENTREZID"     "ENZYME"       "EVIDENCE"
##  [9] "EVIDENCEALL"  "GENENAME"     "GO"           "GOALL"
## [13] "IPI"          "MAP"          "OMIM"         "ONTOLOGY"
## [17] "ONTOLOGYALL"  "PATH"         "PFAM"         "PMID"
## [21] "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"
## [25] "UNIGENE"      "UNIPROT"
```

The `OrgDb =` requires the genome wide annotation for the organism of interest.
The parameter `ont =` is used to specify the subontology we want to explore. As explained in above, these gene ontologies can be cellular component (CC), molecular function (MF) or biological processes (BP), which is the one used in this example. If we want to explore the three terms, we can use `ont = "ALL"`
The `pAdjustMethod =` is to use Benjamini-Hochberg for p-value correction
Once the analysis is complete, we can create and view the table of results using:

```
GO_ORA <- data.frame(ego)
View(GO_ORA)
```

	ID	Description	GeneRatio	BgRatio	pvalue	p.adjust	qvalue	geneID	Count
GO:0043062	GO:0043062	extracellular structure organization	79/1302	352/14996	1.650901e-15	7.351520e-12	5.778191e-12	MMP1/TNFRSF11B/COL4A2/FGG/PDGFRA/NFKB2/MMP19/...	79
GO:0030198	GO:0030198	extracellular matrix organization	72/1302	307/14996	2.744641e-15	7.351520e-12	5.778191e-12	MMP1/TNFRSF11B/COL4A2/FGG/PDGFRA/NFKB2/MMP19/...	72
GO:0009617	GO:0009617	response to bacterium	95/1302	497/14996	9.652307e-14	1.723580e-10	1.354710e-10	PLCG2/B2M/CCL2/INAVA/C15orf48/ADM/FOS/PALM3/ARG2...	95
GO:0001525	GO:0001525	angiogenesis	89/1302	478/14996	2.782520e-12	3.726490e-09	2.928968e-09	COL15A1/CCL2/PRL/AQP1/RAPGEF3/COL4A2/ADM/STAT3/P...	89
GO:0045785	GO:0045785	positive regulation of cell adhesion	74/1302	367/14996	3.979370e-12	4.263497e-09	3.351048e-09	CCL2/PNP/RUNX3/XCL1/PLPP3/NFKBIZ/FGG/CDH13/ERBB2...	74
GO:0022407	GO:0022407	regulation of cell-cell adhesion	73/1302	365/14996	8.396310e-12	7.496506e-09	5.892148e-09	CCL2/C1QTNF1/PNP/RUNX3/XCL1/PLPP3/NFKBIZ/ARG2/FG...	73
GO:0050727	GO:0050727	regulation of inflammatory response	71/1302	357/14996	2.142968e-11	1.639983e-08	1.289003e-08	XCL1/CLU/NFKBIZ/C3/EDNRB/ITGAM/TLR3/FABP4/FCER1G/...	71
GO:0050865	GO:0050865	regulation of cell activation	88/1302	497/14996	6.039420e-11	4.044147e-08	3.178642e-08	IRS2/CCL2/C1QTNF1/PNP/RUNX3/XCL1/NFKBIZ/ARG2/CAP...	88
GO:0050900	GO:0050900	leukocyte migration	71/1302	371/14996	1.337948e-10	7.963767e-08	6.259408e-08	SLC8B1/MMP1/CCL2/XCL1/SLC7A7/ARHGEF5/WDR1/EDNR...	71
GO:0007159	GO:0007159	leukocyte cell-cell adhesion	61/1302	303/14996	3.395918e-10	1.819193e-07	1.429860e-07	CCL2/PNP/RUNX3/XCL1/NFKBIZ/ARG2/ERBB2/DOCK8/IGF2...	61
GO:0002526	GO:0002526	acute inflammatory response	36/1302	137/14996	9.677525e-10	4.712954e-07	3.704316e-07	CLU/STAT3/C3/EDNRB/FCER1G/CRP/CNR1/PROS1/RHBDD3...	36
GO:0002237	GO:0002237	response to molecule of bacterial origin	56/1302	282/14996	3.051746e-09	1.362350e-06	1.070788e-06	PLCG2/B2M/CCL2/INAVA/ADM/FOS/PALM3/NFKB2/NOS3/L...	56
GO:0043312	GO:0043312	neutrophil degranulation	79/1302	464/14996	3.843460e-09	1.420644e-06	1.116606e-06	CRISP3/B2M/SLC44A2/GPR84/PNP/PIGR/FABP5/SLC15A4/L...	79
GO:0032102	GO:0032102	negative regulation of response to external stimulus	58/1302	299/14996	3.981650e-09	1.420644e-06	1.116606e-06	CCL2/C1QTNF1/ARG2/FGG/PDGFRA/NOS3/FGB/HTRA1/RIO...	58

The table has in total 593 rows (only the first 14 are shown above).
The columns are:

- ID: Corresponding ID of the Gene Ontology terms
- Description: Corresponding names for the Gene Ontology term
- GeneRatio: Number of genes associated to the term relative to the total number of genes of interest
- BigRatio: Number of genes in the term relative to the total number of genes in the background
- geneID: Names of the genes associated to the term. Notice that these ID are different from the ones used for input (ENSEMBL). This is because we asked to map the gene ID to a gene Name when using `readable = TRUE`

Only the ENSEMBL ID that are mapped to a gene Name will be used in the analysis (even if `readable = FALSE`). Thus, the total number of genes of interest and genes in the background in GeneRatio and BigRatio can be different to the number of genes in the input lists.

As we can notice, there are several biological processes enriched in this analysis that seem very similar. This is because several terms are “child” of other terms, i. e., they are related terms (for example, ‘positive regulation of cell adhesion’ and ‘regulation of cell-cell adhesion’).

The function *simplify* helps to remove redundancy of enriched GO terms from the output of *enrichGO* function.

A word of caution: if you look above in page 14, after running `library(clusterProfiler)` , there is a text saying: `The following object is masked from 'package:DelayedArray': simplify`
This is meaning that the function *simplify*, is 'masked', i. e., it is recognized as part of the other package (DelayedArray) and thus, it will perform the function from that package. To 'unmask' the function for simplify, we run first:

```
clusterProfiler::simplify
```

Afterward, we can run the function simplify with the defaults options

```
egoSimp <- simplify(ego)
```

Remember that this function works with the output of enrichGO, which we named 'ego'.
Now, we can create and view the table of results.

```
GO_ORASimp <- data.frame(egoSimp)
View(GO_ORASimp)
```

	ID	Description	GeneRatio	BgRatio	pvalue	p.adjust	qvalue	geneID	Count
GO:0043062	GO:0043062	extracellular structure organization	79/1302	352/14996	1.650901e-15	7.351520e-12	5.778191e-12	MMP1/TNFRSF11B/COL4A2/FGG/PDGFRA/NFKB2/MMP19/...	79
GO:0030198	GO:0030198	extracellular matrix organization	72/1302	307/14996	2.744641e-15	7.351520e-12	5.778191e-12	MMP1/TNFRSF11B/COL4A2/FGG/PDGFRA/NFKB2/MMP19/...	72
GO:0009617	GO:0009617	response to bacterium	95/1302	497/14996	9.652307e-14	1.723580e-10	1.354710e-10	PLCG2/B2M/CCL2/INAVA/C15orf48/ADM/FOS/PALM3/ARG2...	95
GO:0001525	GO:0001525	angiogenesis	89/1302	478/14996	2.782520e-12	3.726490e-09	2.928968e-09	COL15A1/CCL2/PRL/AQP1/RAPGEF3/COL4A2/ADM/STAT3/P...	89
GO:0045785	GO:0045785	positive regulation of cell adhesion	74/1302	367/14996	3.979370e-12	4.263497e-09	3.351048e-09	CCL2/PNP/RUNX3/XCL1/PLPP3/NFKBIZ/FGG/CDH13/ERBB2...	74
GO:0050727	GO:0050727	regulation of inflammatory response	71/1302	357/14996	2.142968e-11	1.639983e-08	1.289003e-08	XCL1/CLU/NFKBIZ/C3/EDNRB/ITGAM/TLR3/FABP4/FCER1G/...	71
GO:0050865	GO:0050865	regulation of cell activation	88/1302	497/14996	6.039420e-11	4.044147e-08	3.178642e-08	IRS2/CCL2/C1QTNF1/PNP/RUNX3/XCL1/NFKBIZ/ARG2/CAP...	88
GO:0050900	GO:0050900	leukocyte migration	71/1302	371/14996	1.337948e-10	7.963767e-08	6.259408e-08	SLC8B1/MMP1/CCL2/XCL1/SLC7A7/ARHGEF5/WDR1/EDNR...	71
GO:0007159	GO:0007159	leukocyte cell-cell adhesion	61/1302	303/14996	3.395918e-10	1.819193e-07	1.429860e-07	CCL2/PNP/RUNX3/XCL1/NFKBIZ/ARG2/ERBB2/DOCK8/IGF2...	61
GO:0002526	GO:0002526	acute inflammatory response	36/1302	137/14996	9.677525e-10	4.712954e-07	3.704316e-07	CLU/STAT3/C3/EDNRB/FCER1G/CRP/CNR1/PROS1/RHBDD3...	36
GO:0043312	GO:0043312	neutrophil degranulation	79/1302	464/14996	3.843460e-09	1.420644e-06	1.116606e-06	CRISP3/B2M/SLC44A2/GPR84/PNP/PIGR/FABP5/SLC15A4/L...	79

This table has now 213 rows (so the 593 significantly enriched BP were reduced to 213) and as we can see, from redundant terms such as 'positive regulation of cell adhesion' and 'regulation of cell-cell adhesion', only one is chosen.

Enrichment analysis of KEGG pathways
To perform this enrichment analysis, we use the function *enrichKEGG*.
This function requires the input ID type to be 'kegg', 'ncbi-geneid', 'ncbi-proteinid' or 'uniprot'. The ID of the genes in our example is ENSEMBL. Therefore, we will need to convert this ID first before using the enrichKEGG function.
The clusterProfiler package have the function *bitr* that converts biological ID:

```
sigUPKegg<- bitr(sigUP, fromType='ENSEMBL', toType='UNIPROT',
OrgDb='org.Hs.eg.db')
## 'select()' returned 1:many mapping between keys and columns
## Warning in bitr(sigUP, fromType = "ENSEMBL", toType = "UNIPROT", OrgDb
="org.Hs.eg.db"): 26.07% of input gene IDs are fail to map...
```

Of course, the organism `OrgDb=` should be specified. Here, we converted the list of ENSEMBLE IDs to Uniprot IDs. The resulting output (sigUPKegg) has two columns, one for each identifier

```
View(sigUPKegg)
```

	ENSEMBL	UNIPROT
2	ENSG00000182333	P07098
3	ENSG00000155792	Q8TB45
4	ENSG00000105053	Q8IV63

(only the first 3 rows are shown)

Not all the ENSEMBL IDs are mapped to a Uniprot IDs, since several are not related to protein-coding genes, as mentioned in page 2 of this tutorial. For this reason, some gene IDs are fail to map (26.07% in our example).
Now we can perform the KEGG enrichment analysis.

```
kk <- enrichKEGG(gene= sigUPKegg$UNIPROT,
                 keyType = 'uniprot'
                 organism = 'hsa',
                 pvalueCutoff = 0.05)
```

After `gene=` we input the list of genes with an accepted ID type. In our example, this is the second column of the sigUPKegg object, named "UNIPROT", that we can specify doing `sigUPKegg$UNIPROT`. Another way to obtain the same output is doing: `sigUPKegg[,2]`
For `organism =` we input the specie of interest (human). However, notice that the nomenclature for the organism (`'hsa'`) is different from the one we used for the GO enrichment analysis (`org.Hs.eg.db`). This is because the function adopts the same nomenclature as in the KEGG pathway database. The full list of KEGG supported organisms can be accessed via: http://www.genome.jp/kegg/catalog/org_list.html.
Once the analysis is complete, we can create and view the table of results using:

```
KK_ORA <- data.frame(kk)
View(KK_ORA)
```

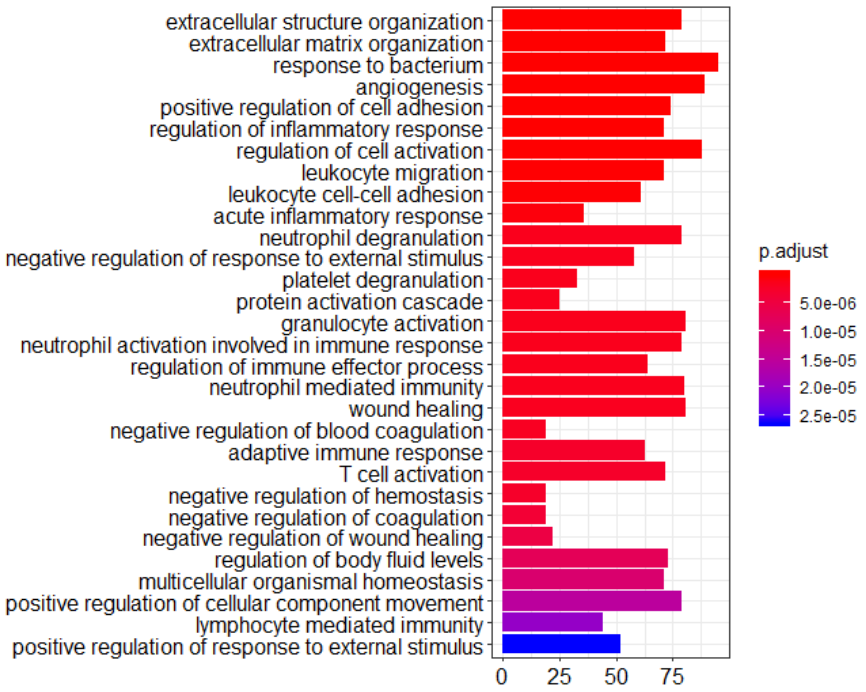
	ID	Description	GeneRatio	BgRatio	pvalue	p.adjust	qvalue	geneID	Count
hsa04610	hsa04610	Complement and coagulation cascades	36/1028	109/10907	6.611108e-12	2.069277e-09	1.670175e-09	P10909/P02679/B4DR57/P01024/V9HWA9/P11215/P02675/...	36
hsa04066	hsa04066	HIF-1 signaling pathway	40/1028	183/10907	3.016781e-07	4.721262e-05	3.810670e-05	P16885/P40763/Q01813/P29474/P04626/X5DNK3/P17948/...	40
hsa04010	hsa04010	MAPK signaling pathway	74/1028	443/10907	6.148854e-07	5.679067e-05	4.583744e-05	P01100/Q6FG41/Q16620/Q5VWE5/Q548C2/A0A087WVR4/...	74
hsa05202	hsa05202	Transcriptional misregulation in cancer	49/1028	253/10907	7.257594e-07	5.679067e-05	4.583744e-05	Q9BYH8/A0A087WVR4/P01106/A0A024R759/P08581/P112...	49
hsa05144	hsa05144	Malaria	22/1028	77/10907	1.411681e-06	8.837120e-05	7.132702e-05	P13500/A0A024R759/P08581/P18827/P16581/P35442/P618...	22

The table has 34 rows in total.

Visualization of the ORA results
The application of visualization methods is always recommended since these methods help to interpret the enrichment results. In addition, graphs are nicer than tables or text to show in presentations, reports, articles, etc.
All the online tools listed in page 13 provide some visualization method, such as Venn diagrams, bar plots, dot plots, networks and so on.
The package clusterProfiler also offers several functions to create graphs. We will see some of them in this tutorial, but they are all detailed here: https://yulab-smu.github.io/clusterProfiler-book/chapter12.html for more information.
We will use the results from the Gene Ontology ORA (or significantly enriched biological terms) for visualization. Of course, all the methods can be applied to the KEGG pathway ORA results as well.

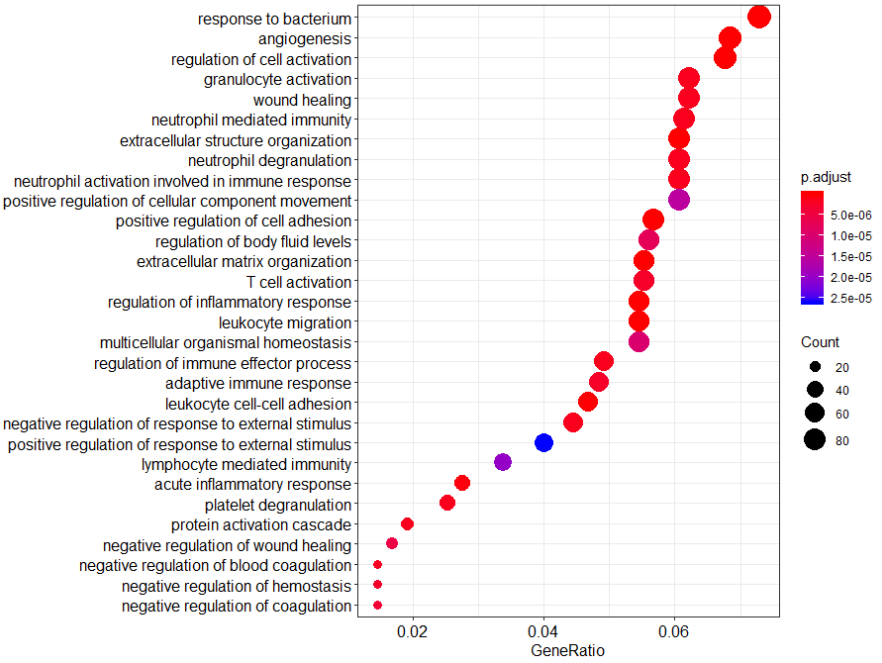
***Bar plots**:* The heights of the bar are related to the number of genes enriching the term. The color of the bar depends on the p-value.
To look at the top 30 non-redundant terms we run:

```
barplot(egoSimp, showCategory=30)
```



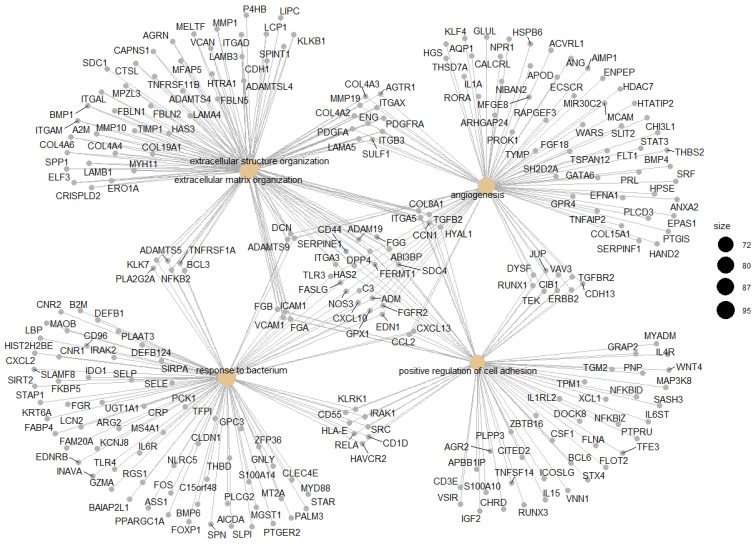
Dot plots: They are similar to the bar plots but the dot size is related to number of genes. The list of the top 30 significant terms are ordered according to the gene ratio (number of genes of interest enriching the term relative to the total number of genes in the term).

```
dotplot(egoSimp, showCategory=30, orderBy = 'GeneRatio')
```



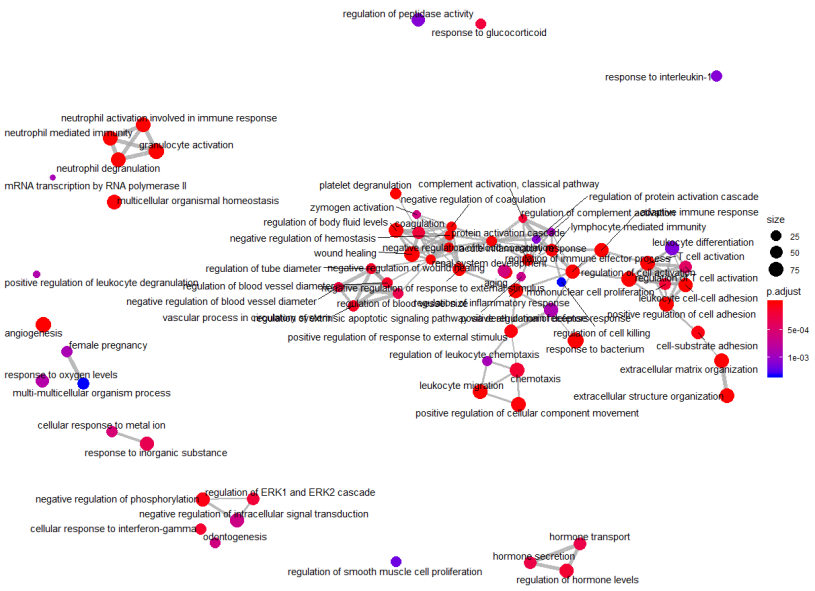
Gene-Concept Network: This function is useful to visualize the genes that are involved in the significant terms. In addition, it helps to identify those genes that are involved in more than one category.
To construct the graph, the gene IDs are converted to symbols first with the function *setReadable*. This function is applied to an enrichResult object, the corresponding organism (OrgDB) should be specified and finally, the key type, or type of ID, of the genes.
Then, the network is graphed with the function *cnetplot*. Notice that we are going to display the top five non-redundant terms for simplicity of the plot.

```
edox <- setReadable(egoSimp, OrgDb = 'org.Hs.eg.db', keyType = 'ENSEMBL')
cnetplot(edox, showCategory=5)
```



Enriched Map: This function organizes the enriched terms into a network with edges connecting overlapping gene sets, which helps to identify the main functional clusters. In this example, we will run this function in the top 70 non-redundant terms.

```
emapplot(egoSimp, showCategory = 70)
```

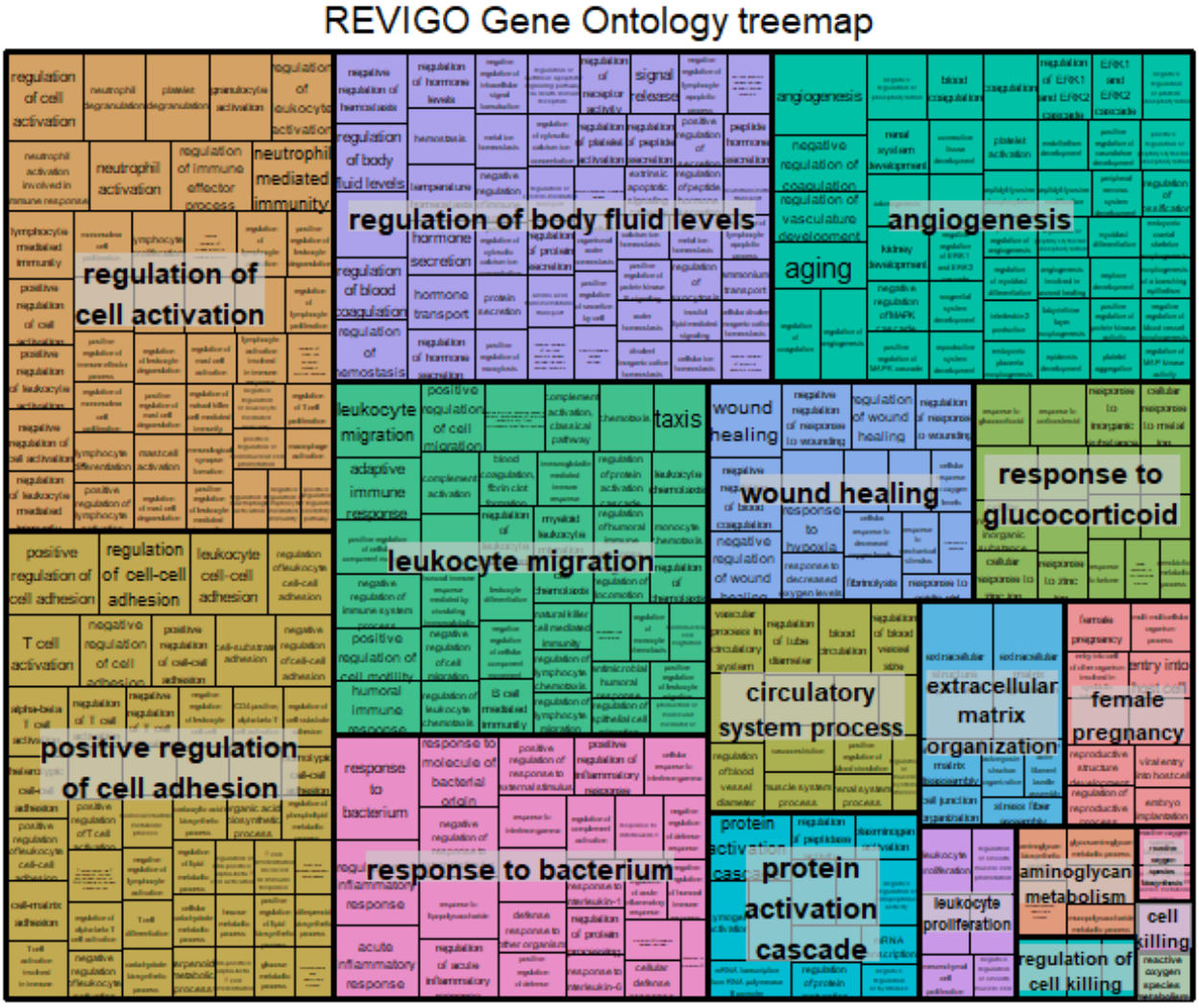


Looking at this plot, we can see that the terms have been organized in clusters related to extracellular matrix organization and cell adhesion, immune response, neutrophil activation, regulation of blood vessels and coagulation female pregnancy, intracellular signanling and hormone secretion; among others.

Tree Map: One nice online free tool that provides a general view of the main enriched processes is REVIGO [4], mentioned in page 13. This tool summarize the results by clustering the terms according to the semantic similarities. To use REVIGO, you should export the table GO_ORA table (with the redundant terms) using:

```
write.table(GO_ORA, "ResultsGO-ORA.txt", sep=",", col.names=NA)
```

And then copy and paste the list of GO IDs into REVIGO. The TreeMap results looks as follow:



Where the all the list of BP terms are summarized in 18 terms. The size of each box depends on the number of terms related to the main terms and the p-value for each term.

Conclusion

In this tutorial, we have shown the standard main steps followed to analyze RNA sequencing data, after the generation of the matrix of raw counts. These steps are an unsupervised analysis of the data, determination of DEGs and a functional analysis of those genes.
Here, we analyzed data generated from human endometrial samples to comprehend the transcriptomic changes occurring in the uterus between the early and mid-secretory phase, or in other words, in the 6-days period -after ovulation- when the endometrium develops the potential to nest an embryo.
The unsupervised analysis showed a well-defined transcriptomic profile for each phase, even when samples were obtained from the same woman in one phase and the other. The different transcriptomic profile between phases resulted in a very high number of genes with low adjusted p-values after the differential expression analysis. Finally, the functional analysis showed that genes that are stimulated in the mid-secretory phase, and thus responsible for preparing the uterus to receive the embryo, are related to cell adhesion, development of blood vessels, inflammation and interestingly, female pregnancy, among others. These processes are already recognized as part of the modifications that undergoes the endometrium to nest the embryo, including the high infiltration of immune cells. The same functional analysis can be applied to the list of down-regulated DEGs, in order to gain insights about the processes that should be inhibited during this period.
In summary, the application of the pipeline presented in this tutorial allowed us to elucidate the biological meaning of the analyzed experiment. However, remember that there are other ways to analyze RNA-seq data, which do not always include the determination of DEGs.

References

[1] Love MI, Huber W, Anders S, Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol 2014;15(12):550.
[2] Yu G, Wang LG, Han Y, He QY, clusterProfiler: an R package for comparing biological themes among gene clusters. OMICS 2012;16(5):284–7.
[3] Altmäe S, Koel M, Võsa U, Adler P, Suhorutšenko M, Laisk-Podar T et al., Meta-signature of human endometrial receptivity: a meta-analysis and validation study of transcriptomic biomarkers. Sci Rep 2017;7(1):10077.
[4] Supek F, Bošnjak M, Škunca N, Šmuc T, REVIGO summarizes and visualizes long lists of gene ontology terms. PLoS One 2011;6(7):e21800.