Differential Gene Expression Analysis

RNA-seq Data Analysis course, EBI, April 2020

Alexey Larionov

2020-04-08

Contents

Chapter 1

Introduction

This is a practical tutorial illustrating analysis of Differentially Expressed Genes (DEGs) in RNA-seq data using **DESeq2** and **edgeR** packages. This tutorial avoids discussing statistical concepts that underlie the analysis, such as normalization, generalised regression models, distribution of counts etc. These aspects will be discussed separately in accompanying lecture slides. The only conceptual point that we touch here is the **"borrowing"** of data between genes during the dispersion (variance) assessment.

1.1 Dispersion assessment

Most of the current tools for Differential Gene Expression analysis in RNA-seq data (including **DESeq2** and **edgeR**) were developed at the time when RNA-seq was quite expensive, so the researchers could only analyse a small number of samples (e.g. less than 10 per group). The small number of samples led to a very large dispersion (variance) estimates in each separate gene. Thus, the statisticians had to **"borrow"** data between samples and genes to facilitate a meaningful analysis. Initially the statisticians suggested to estimate "average" dispersion using all the genes and samples together. This provided much narrower estimates. However, it was open to criticism because the dispersion in RNA-seq counts depends on the gene expression. Thus, the next step was to model a gene dispersion using only the genes with similar level of expression. This approach was adopted in **edgeR** and in **DESeq2** packages.

In essense, these packages estimate each individual gene dispersion using two components:

1) the actual dispersion observed for the gene

2) the dispersion of other genes with a similar level of expression

When the number of samples is sufficiently large, **DESeq2** and **edgeR** put most of the weight on the first component. When the number of samples is small, the actual dispersion estimates are too broad for a meaningful analysis, so the final estimate "shrinks" toward the second component. In other words, the degree of data **"borrowing"** is reduced if a large number of samples is available.

1.2 Bioconductor's tutorials

Currently (in April 2020) Bioconductor provides several good practical tutorials in RNA-seq Differential gene expresion, including:

- **DESeq2**: https://www.bioconductor.org/packages/devel/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html
- **edgeR** : http://bioconductor.org/packages/release/bioc/vignettes/edgeR/inst/doc/edgeRUsersGuide.pdf
- limma-voom and edgeR: https://master.bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html

It is highly advised to study these tutorials along with the present one. They will provide different perspectives to the analysis and lots of additional practical information (sometimes, even too much infromation :) Noteworthy, the **DESeq2** Bioconductor tutorial discusses in detail the modern data import packages, like **tximport** and **tximeta** for data input from the modern pseudo-alignment tools like **Salmon**.

1.3 Default threshiolds

Importantly, the Bioconductor tutorials illustrate the data analysis using small datasets (typically, less than 10 samples). To provide an alternative perspective, here we analyse a dataset consisting of hundreds of samples. This analysis illustrates that some of the packages' default settings (i.e. DESeq'2 default **no-change null hypothesis** and **0.1 FDR** threshold) may suggest an unrealistically large proportion of Differentially Expressed Genes (DEGs), when a large number of samples is available for significantly differet biological conditions. For instance, more than a half of the genes are suggested to be DEGs using the default **DESeq2** thresholds in our analysis. Adjusting the thresholds, for instance, to **2-fold change** at **FDR $<$ 0.01** brings the number of suggested DEGs to around 10%, which looks more realistic (and consistent with a common assumption that DEGs should constitute only a minority of genes).

1.4 TCGA

Data for this tutorial was obtained from TCGA project:

- https://portal.gdc.cancer.gov/repository

TCGA provides multiple types of data (including clinical annotations, germline and somatic DNA-sequencing, RNA-sequencing and epigenomic data) for multiple types of human cancer. The dataset has data for hundreds of clinical biopsies for each type of studied cancer, sometimes with paired normal tissue.

1.4.1 TCGA data access

TCGA data, which may identify individual patients (such as raw sequencing data) are protected. However, RNA-Seq counts at gene-level are provided with open access. For this tutorial we use a set of selected TCGA Breast Cancer cases (TCGA-BRCA).

TCGA provides RNA-seq counts in several formats, including raw counts generated by HTSeq and normalised counts (FPKM and FPKM-UQ normalised). Importantly, DESeq2 and edgeR require **raw** counts for analysis.

Selected clinical annotations were extracted from TCGA **xml** annotation files using a set of in-house scripts. In addition to **xml** files, TCGA provides clinical annotations in plain tabulated format (**txt** format). However, these **txt** files have less annotations than the source **xml** files. Also, at the time of preparing these notes (beginnig of 2020), I experienced some bugs in retrieval of **txt** files from TCGA.

1.4.2 TCGA RNA-seq pipeline

Information about TCGA pipeline upstream of the HTSeq counts is available here:

- https://docs.gdc.cancer.gov/Data/Bioinformatics_Pipelines/Expression_mRNA_Pipeline

The TCGA pipeline represents a good example of cutting-edge RNA-seq analysis, how it was performed several years ago. Importantly, it says that RNA-seq counts were generated using **GENCODE v.22** gene annotations, which we will use later in this analysis.

1.5 Tutorial structure

This tutorial is written in R-markdown, where chunks of code are combined with the output.

1.5.1 Samples and Genes

First we will **read and explore data for samples and genes**. The source files include more information than we need, so only the information required for this analysis will be extracted.

1.5.2 DESeq2 analysis

Then **the data is imported into the DSEq2 data set** and the **dataset is explored**.

As a part of exploration, the data will be variance-normalised, which is required for **PCA** and **Hierarchical Clustering**. Importantly, this normalization is used *for data exploration only.* The actual differential expression analysis will use non-normaliased raw data.

Then the **differential expression analysis** will be done (DSEq2 does it with just a single line of code :)

Finally the results of differential expression analysis are extracted and explored: including technical data (**variance adjustments plot**), a table with with top differentially extracted genes, and several ways of visualising the results (**MA plot**, **Volcano plot** and **Hierarchical Clustering**).

Importantly, first we will extract results using the default settings (H0 of no change at 0.1 FDR), which will suggest unsralisticaly hight number of DEGs. Then we will adjust the settings (H0 of less than 2-fold change at 0.01 FDR) to obtain a meaninful result.

1.5.3 edgeR analysis

A similar analysis will be performed using **edgeR**, except for Hierarchical clustering, which will not be repeated using **edgeR** data.

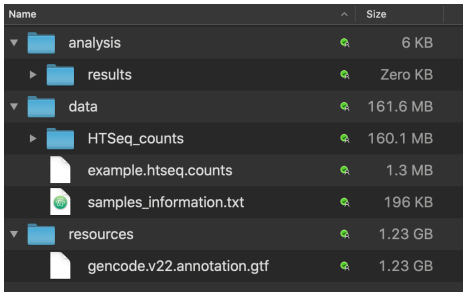
1.5.4 Comparison of DEGs

The lists of DEGs be compared between **DESeq2** and **edgeR** to show a reasonable concordance between the methods.

Importantly, this tutorial is not concerned about the biological interpretation of the results (although the it confirms the previously known ER-related genes in breast cancer). The aim of this tutorial is to give an example of a practical use of **DESeq2** and **edgeR** for differential gene analysis in RNA-seq data.

1.6 Content of the base folder

The analysis expects a base-folder, which contains sub-folders and files as ilustrated below:



For convinience, the counts data, the samples information file and gencode.v22 **gtf** file were downloaded to local storage, as shown.

Chapter 2

Setting the Enviroment

2.1 Required libraries

This chunk shows how to install the required libraries. The libraries should be installed just once. So, you dont need to run this chunk again each time when you run a new analysis. The required packages will install many dependancies. Importantly, *BiocManager* may require R version 3.6 or above.

```
install.packages("dplyr") # for data wrangling
install.packages("RColorBrewer") # for plotting heatmaps
install.packages("pheatmap") # for plotting heatmaps
install.packages("VennDiagram") # for plotting Venn diagram

install.packages("BiocManager") # for installing Bioconductor packages
BiocManager::install("rtracklayer") # for reading GTF file
BiocManager::install("DESeq2") # for differential expression analysis
```

2.2 Base folder etc

Record the analysis start time, clean-up environment, customise options and set the **base folder**

```
# Start time
Sys.time()
```

```
## [1] "2020-04-08 16:26:36 BST"
```

```
# Intial clean-up
rm(list=ls())
```

```
graphics.off()

# Customised options
options(stringsAsFactors = F)

# Base folder
base_folder="/Users/alexey/OneDrive/Documents/Teaching/Lecturing/rna-seq-ebi-2020-tuto
```

Chapter 3

Preparing sample information

This chunk reads and explores the samples' information. Then it selects **ER-positive** (Estrogen Receptor Positive) and **Triple-negative** (Estrogen receptor, Progesterone resceptor and HER2 negative) tumours. You may try different comparisons, e.g. the dataset includes some normal tissue samples paired with tumours.

In addition to the ER status, it keeps **OCT-imbedding** status. **OCT-imbedding** is a small variation in experimental protocol during RNA extraction. Some of the TCGA BRCA samples were imbedded into OCT prior RNA extraction, while the others were not imbedded. Thus **OCT-imbedding** could be considered a potentially confounding variable to illustrate dealing with batch effects.

```
# Read samples information
samples_information_file <- file.path(base_folder,"data","samples_information.txt")
samples_information.df <- read.table(samples_information_file, header=T, sep="\t")

# Explore samples information data
dim(samples_information.df)
```

```
## [1] 629  18
```

```
colnames(samples_information.df)
```

```
##  [1] "file_id"                "file_name"              "sample_type"            "sample_oct
```

```
# Explore samples types
table(samples_information.df$sample_type)
```

```
##
```

```
##       Primary Tumor Solid Tissue Normal
##                 567                  62
```

```
# Select tumours
tumours <- samples_information.df$sample_type == "Primary Tumor"

# Select ER-positive samples
er_pos <- samples_information.df$er == "Positive" &
          samples_information.df$pr == "Positive" &
          samples_information.df$her2 == "Negative"

# Select Triple-negative samples
triple_neg <- samples_information.df$er == "Negative" &
              samples_information.df$pr == "Negative" &
              samples_information.df$her2 == "Negative"

# Combine the conditions and check the count
selected_samples <- tumours & ( er_pos | triple_neg )
sum(selected_samples)
```

```
## [1] 238
```

```
# Select fields needed for analysis
selected_fields <- c("bcr_patient_barcode","file_name", "sample_oct_embedded", "er")

# Keep only selected samples and columns
samples.df <- samples_information.df[selected_samples, selected_fields]

# Rename the columns: to make the names concise
colnames(samples.df) <- c("patient","file","oct","er")

# Check the samples data frame
dim(samples.df)
```

```
## [1] 238   4
```

```
head(samples.df)
```

```
##        patient                                                file   oct       er
## 1 TCGA-A7-A0DA a33029dd-b5fa-4be0-9cbf-971d289146dd.htseq.counts.gz false Negative
## 3 TCGA-D8-A1XU 8d54214a-1d9b-4fea-9c42-5bbb3cd11da9.htseq.counts.gz false Positive
## 5 TCGA-D8-A143 4b19c0e2-2a61-4f0a-9257-4a528e6b320e.htseq.counts.gz false Negative
## 6 TCGA-A7-A4SB cc233ff9-d5fb-4e9b-9007-f58d008df995.htseq.counts.gz false Positive
## 7 TCGA-D8-A1XR bdd8c340-250b-474a-8802-7653b7884ced.htseq.counts.gz false Positive
## 8 TCGA-BH-A18L 5bc7e90d-0fa4-4bde-bee8-4f9b92de03a2.htseq.counts.gz  true Positive
```

```
# ER-status vs OCT-embedded
table(samples.df[,c("oct","er")])
```

```
##        er
## oct     Negative Positive
##   false       23       73
##   true        24      118
```

```
# Clean-up
rm(samples_information_file, tumours, er_pos, triple_neg, selected_samples, selected_fields, samp
```

Chapter 4

Preparing gene information

TCGA used the **gencode.v22.annotation.gtf** annotation file to generate RNA-seq counts:

- https://docs.gdc.cancer.gov/Data/Bioinformatics_Pipelines/Expression_mRNA_Pipeline

For convinience, this GTF file was downloaded to the resources folder for this tutorial. However, it can also be downloaded directly from gencode:

- https://www.gencodegenes.org/human/release_22.html

GTF files can be red into R data-frame using **readGFF** function from **rtracklayer** package.

4.0.1 Load required libraries

Warnings and **messages** printed during the library loading would clutter the tutorial. Thus they **are supressed** in this code. **This should not be done during actual analysis** because these Warnings and Messages are important. Thus, many functions in *genomic packages* may have the same names as *dplyr* functions. So, ignoring Warnings and Messages may lead to unexpected interferences.

```
#install.packages(dplyr)
suppressWarnings(suppressMessages(library(dplyr)))

#install.packages("BiocManager")
#BiocManager::install("rtracklayer")
suppressWarnings(suppressMessages(library(rtracklayer))) # for readGFF()
```

4.0.2 Read genes data

The genes' Names and IDs **must** be read from the file that was used during RNA-seq counts preparation. In case of the TCGA RNA-seq data, this is **gencode.v22.annotation.gtf**.

```
# Read appropriate annotations file
gencode_22_file <- file.path(base_folder,"resources","gencode.v22.annotation.gtf")
gencode_22.df <- readGFF(gencode_22_file)

# Explore the annotations
dim(gencode_22.df)
```

```
## [1] 2563671      26
```

```
colnames(gencode_22.df)
```

```
##  [1] "seqid"                      "source"                     "type"
```

```
# Extract gene_id-s and gene_name-s
genes.df <- gencode_22.df %>%
  select(gene_id, gene_name) %>%
  arrange(gene_name) %>%
  distinct()

# Copy gene IDs to rownames
rownames(genes.df) <- genes.df$gene_id

# Check result
head(genes.df) # Interestingly, the gene names are not unique
```

```
##                              gene_id gene_name
## ENSG00000275877.1 ENSG00000275877.1 5_8S_rRNA
## ENSG00000276871.1 ENSG00000276871.1 5_8S_rRNA
## ENSG00000252830.2 ENSG00000252830.2   5S_rRNA
## ENSG00000276442.1 ENSG00000276442.1   5S_rRNA
## ENSG00000274408.1 ENSG00000274408.1   5S_rRNA
## ENSG00000274059.1 ENSG00000274059.1   5S_rRNA
```

```
tail(genes.df)
```

```
##                                gene_id gene_name
## ENSG00000203995.8   ENSG00000203995.8    ZYG11A
## ENSG00000232242.2   ENSG00000232242.2  ZYG11AP1
## ENSG00000162378.11 ENSG00000162378.11    ZYG11B
## ENSG00000159840.14 ENSG00000159840.14       ZYX
## ENSG00000074755.13 ENSG00000074755.13     ZZEF1
## ENSG00000036549.11 ENSG00000036549.11      ZZZ3
```

```
# Clean-up
rm(gencode_22_file, gencode_22.df)
```

Chapter 5

DESeq2 analysis

First, load DESeq2 library. Beware: DESeq2 dependencies may interfere with *dplyr*. To avoid interference you may use the library name explicitly, when calling functions, e.g. use **DESeq2::plotPCA()** instead of just **plotPCA()** (not used in this tutorial).

```
#install.packages("BiocManager")
#BiocManager::install("DESeq2")
suppressWarnings(suppressMessages(library(DESeq2)))
```

5.1 DESeqDataSet object

5.1.1 Make DESeqDataSet object

DESeq2 provides several functions for data import from different upstream tools that generate RNA-seq counts. We use the **DESeqDataSetFromHTSeqCount** function that matches the type of data provided by PCA.

```
# Set counts folder
counts_folder=file.path(base_folder,"data","HTSeq_counts")

# Make the design variables as factors
# (remember: we customised stringsAsFactors to FALSE in the Start section)
samples.df$oct <- as.factor(samples.df$oct)
samples.df$er <- as.factor(samples.df$er)

# Import sanmples data and counts into DESeqDataSet object
# "dds" stands for Deseq-Data-Set
dds <- DESeqDataSetFromHTSeqCount(sampleTable = samples.df,
```

```
                                    directory = counts_folder,
                                    design = ~ oct + er)

# Clean-up
rm(counts_folder)
```

5.1.2 Explore DESeqDataSet object

In short, the **DESeqDataSet** object generated by **DESeqDataSetFromHTSeqCount** includes the following slots:

- slot for the matrix with **counts** (a part of the **assays** collection, although no other assays is added when reading HTSeq counts)
- slot for the samples information (**colData**)
- slot for the genes information (**rowData**). It is empty when reading HTSeq counts. To illustrate the use of **rowData** we will manually add the gene names (from gencode.22 GTF file)
- slot for some **metadata** (is virtually empty when reading HTSeq counts)
- slot for **design**: an important slot that includes a string that defines the analysis, which will be performed by DESeq2 later

```
# A very short summary
summary(dds)
```

```
## [1] "DESeqDataSet object of length 60483 with 0 metadata columns"
```

```
# A bit more expanded summary
dds
```

```
## class: DESeqDataSet
## dim: 60483 238
## metadata(1): version
## assays(1): counts
## rownames(60483): ENSG00000000003.13 ENSG00000000005.5 ... ENSGR0000280767.1 ENSGR000
## rowData names(0):
## colnames(238): TCGA-A7-A0DA TCGA-D8-A1XU ... TCGA-D8-A1XB TCGA-AO-A03M
## colData names(2): oct er
```

```
# Metadata slot
metadata(dds)
```

```
## $version
## [1] '1.26.0'
```

```
# "counts" slot
counts(dds)[1:5,1:5]
```

```
##                    TCGA-A7-A0DA TCGA-D8-A1XU TCGA-D8-A143 TCGA-A7-A4SB TCGA-D8-A1XR
## ENSG00000000003.13         2724         5645         6180         2558         3884
## ENSG00000000005.5             7            6            1           32          109
## ENSG00000000419.11         1962         4926         2624         1068         3337
## ENSG00000000457.12         1973         2271         1860         1362         4953
## ENSG00000000460.15          867          675         2359          494         2284
```

```
# colData slot: samples information
head(as.data.frame(colData(dds)))
```

```
##                oct       er
## TCGA-A7-A0DA false Negative
## TCGA-D8-A1XU false Positive
## TCGA-D8-A143 false Negative
## TCGA-A7-A4SB false Positive
## TCGA-D8-A1XR false Positive
## TCGA-BH-A18L  true Positive
```

```
# rowData slot
rowData(dds) # Genes information is empty
```

```
## DataFrame with 60483 rows and 0 columns
```

```
# Design slot: a string that follows R conventions for a "formula" in glm models.
design(dds)
```

```
## ~oct + er
```

In general, the string in **design** slot folows the conventions for a **formula** in R glm analyses. Importantly, by default, **the last element** in the design formula ("er" in this example) will be used for differential expression analysis. The previous elements ("oct" in this example) will be treated as confounding factors (e.g. for batch effect correction etc).

5.1.3 rowData: adding genes names

If data was imported from **Salmon** counts by **tximeta** then the genes names and coordinates would be added in **Genomic Ranges** format. Unfortunately, **HTSeq** counts are not supported by **tximeta**. So, we add the gene names manually: just to illustrate such opportunity (we are not going to use **rowData** slot later).

```
# Syncronise genes order in genes.df with genes order in dds
genes.df <- genes.df[rownames(dds),]
```

```
# Add gene names to rowData slot of dds
rowData(dds) <- genes.df
```

5.2 Genes filtering

Removing consistently low-expressed genes reduces the number of tests and improves conditions for multiple testing correction later in the analysis. Arbitrarily, in this tutorial we remove any genes with less than 10 cases having counts more than 10.

An alternative/additional approach might be to keep only the most variable genes. However, such approach could be complicated by the fact that variance in RNA-seq counts may depend on the level of expression.

```
# Check the initial number of genes in DESeqDataSet object
nrow(dds)
```

```
## [1] 60483
```

```
# Make index for genes with less than 10 cases having count >= 10
keep <- rowSums(counts(dds) >= 10) >= 10
sum(keep)
```

```
## [1] 28362
```

```
# Update the DESeqDataSet object
dds <- dds[keep,]
nrow(dds)
```

```
## [1] 28362
```

```
# Clean-up
rm(keep)
```

5.3 Exploring source data

Before looking for the specific genes differentialy expressed between ER-positive and Triple-negative breast cancers, it might be interesting to see if these groups are separated in the space of genes expression. One of the methods that allows such visual check is **PCA plot**. An alternative widely used method is **Hierarchical Clustering** (HC).

5.3.1 Normalizing variance

Both, **PCA** and **HC**, prioritise the most variable genes when calculating distances between samples. Because the variance in RNA-Seq counts is often higher in highly-expressed genes, **PCA** and **HC** may be dominated by the data from the most expressed genes. Normalising variance between low- and highly- expressed genes improves informativeness of **PCA** and **HC**.

DESeq2 provides two methods for such variance normalization: **VST** (Variance Stabilizing Transformation) and **Rlog** (regularised log-transformation). In essense, both methods are similar to a simple log-transfrormation with some additional correction for the low-expressed genes. Also, DeSeq2 provides a function for generating **PCA plot** from the *transformed* data.

Note that the transformed data (**vst_dds**) is *NOT* used for the actual differential expression analysis later. The differential analysis will be performed usng the raw counts (**dds**).

```
# Make transformed DESeq dataset
vst_dds <- vst(dds)

# Summary information for the transformed DESeq dataset
vst_dds
```

```
## class: DESeqTransform
## dim: 28362 238
## metadata(1): version
## assays(1): ''
## rownames(28362): ENSG00000000003.13 ENSG00000000005.5 ... ENSG00000281912.1 ENSG00000281920.1
## rowData names(6): gene_id gene_name ... allZero dispFit
## colnames(238): TCGA-A7-A0DA TCGA-D8-A1XU ... TCGA-D8-A1XB TCGA-A0-A03M
## colData names(3): oct er sizeFactor
```

5.3.2 PCA plot

The **PCA plot** below shows a clear separation of ER-positive and ER-negative cases in the space of gene expression. At the same time, OCT-imbedding (true / false) does not influence position of samples in the PCA plot:

```
plotPCA(vst_dds, intgroup = c("er", "oct"))
```

5.3.3 Hierarchical Clustering

The **HC** plot shows that most of ER-negative cases are aggregated together into a separate cluster. However, the separation is not perfect and the **heatmap** is not very informative because of the large number of irrelevant genes.

In this example we use **pheatmap** (pretty heatmap) package to make the heatmap and to perform hierarchical clustering. Other often used packages for plotting heatmaps include **heatmap** and **heatmap.2**. **heatmap** is a reliable base-R tool; however it lacks many convinience features. **heatmap.2** (from gplots package) has a bug, which makes it very slow for plotting large matrices (~28k genes x ~230 samples is considered to be a large matrix for this purpose). Also, the heatmap could be plotted with **ggplot2**, which could provide many advantages (e.g. animation with **plotly**). I selected **pheatmap** here because of its simplicity and convinience; and because this tutorial does not need the advasnced **ggplot2** features.

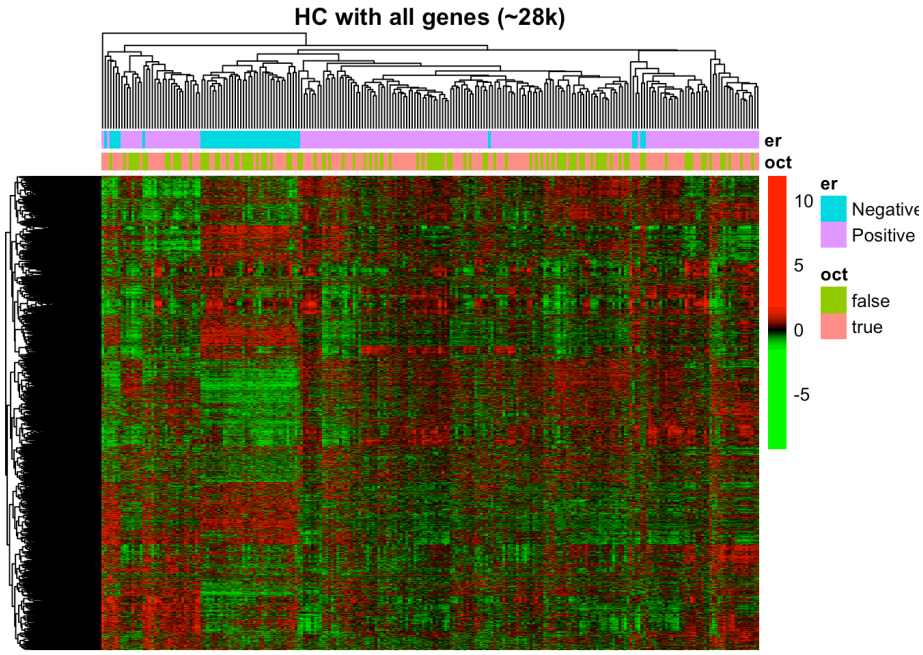
Beware: calculating dendrogram for ~28k hgenes may take very long time and require large memory. The code includes **cluster_rows = FALSE** option in **pheatmap** to avoid the long calculations.

```
# Libraries for drawing heatmap
#install.packages("RColorBrewer")
#install.packages("pheatmap")
```

```r
suppressWarnings(suppressMessages(library(RColorBrewer)))
suppressWarnings(suppressMessages(library(pheatmap)))

# Matrix for clustering and heatmap
vst_counts <- assay(vst_dds)
dim(vst_counts)

# Scale and center by rows.
# Heatmap-making functions often include in-built functions for scaling.
# However, this script does it manually to allow explicit centering, facilitating
# green-black-red palette, when black corresponds to zero (i.e. black = mean gene expression).
vst_counts <- t(scale(t(vst_counts)))

# Make a dataframe with ER- and OCT- status
# (for colour-coded strips along the hea)
cases.df <- as.data.frame(colData(vst_dds))
cases.df <- cases.df %>% select(oct, er)

# Make palette of 21 colours for heatmap drawing.
# The default heatmap palette would be white-yellow-red,
# which is not good for visualizing differences in genes expression.
gbr_21 <- colorRampPalette(c("green", "black", "red"))(n = 21)

# Make breaks between colours
# By default, most hearmap-plotting tools place breaks at equal distance
# Customising breaks by placing them at quantiles makes the hearmap more informative
# (although this is a matter of personal preference :)
# The number of breaksshould be one more than the number of colours in palette
breaks_22 <- quantile(vst_counts, probs=seq(0,1,length.out=22))

# Make heatmap with dendrograms
# Calculating dendrogram for ~28k genes requires large memory and may crash a laptop
# On a laptop with 4 cores and 8GB RAM it may take ~30-40 min
# To suppress genes dendrogram add "cluster_rows = FALSE" option
pheatmap(
  mat = vst_counts, # matrix with data
  color = gbr_21, # user-defined palette
  breaks = breaks_22, # user-defined breaks between colours
  scale = "none", # has been scaled manually
  annotation_col = cases.df, # data frame with ER- and OCT- status
  border_color = NA, # dont draw borders inside the heatmap
  cluster_rows = FALSE, # dont cluster genes (to avoid long calculations)
  show_colnames = FALSE, # too many samples names to show
  show_rownames = FALSE, # too many gene names to show
  main = "HC with all genes (~28k)")
```

```
# Clean-up
rm(vst_counts, gbr_21, breaks_22, cases.df)
```

Note that the figure shown below was generated without < cluster_rows = FALSE > option, to illustrate the genes clustering.



5.4 Calculating DEGs

Overall, the preliminary exploration has suggested that ER-positive and ER-negative tumours have **distinctive** gene expression profiles. So, what are the specific Differentially Expressed Genes (DEGs)?

In practice, all the complicated DESeq2 statistics (discussed elsewere) is hidden from the user within **a single function call**, which adds the DEG results to the original DESeq2 dataset object ...

```
# Calculate the differentially expressed genes
ddx <- DESeq(dds)
```

```
## estimating size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## -- replacing outliers and refitting for 2448 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)

## estimating dispersions

## fitting model and testing
```

```
# summary of the DESeq Dataset with added results
ddx
```

```
## class: DESeqDataSet
## dim: 28362 238
## metadata(1): version
## assays(6): counts mu ... replaceCounts replaceCooks
## rownames(28362): ENSG00000000003.13 ENSG00000000005.5 ... ENSG00000281912.1 ENSG00000281920.1
## rowData names(29): gene_id gene_name ... maxCooks replace
## colnames(238): TCGA-A7-A0DA TCGA-D8-A1XU ... TCGA-D8-A1XB TCGA-AO-A03M
## colData names(4): oct er sizeFactor replaceable
```

```
# Clean-up
rm(dds)
```

5.5 Extracting results

The results of differential gene expression analysis have been added to DESeq2 Data Set. The next chunk of code shows how to extract these results. This tutorial only considers the **key results** of DESeq2 analysis. Look for more details in DESeq2 package documentation.

5.5.1 Explore default thresholds

For each gene DESeq2 provides mean expression, fold change and measures of statistical significance (p and "adjusted p") that allow to conclude whether this gene has been significantly changed.

The default DESeq's thersholds for labelling genes as "significant" are

- the null hypothesis asuming that there is no difference between the compared groups
- FDR 0.1

These thersholds reflect the practices for RNA-seq experiments with a small number of sampes (the most common case at the early time of RNA-seq development).

In this tutorial, when the data includes more than two hundreed samples, **the default DESeq's thresholds suggest that more than a half of the genes are "significantly" changed** between ER-positive and Triple-negative breast cancers (68% = 32 Up-regilated and 36% Down-regulated genes). While it is well known that these cancers have very strong biological differences, such high proportion of changed genes is not realistic. It contradicts to a common assumption, that ony a small proportion of genes should be significantly changed. Also, the distribution of p-values shows noticeable inflation (increased proportion of low p-values) comparatively to the uniform distribution, typically expected under the null.

```
# Testing against H0 of no difference at FDR 0.1
DESeq2_any_change_fdr_0.1 <- results(ddx)
DESeq2_any_change_fdr_0.1
```

```
## log2 fold change (MLE): er Positive vs Negative
## Wald test p-value: er Positive vs Negative
## DataFrame with 28362 rows and 6 columns
##                            baseMean     log2FoldChange              lfcSE
##                           <numeric>          <numeric>          <numeric>
## ENSG00000000003.13 3254.88256154068 -0.690175380631575  0.142786200450243
## ENSG00000000005.5    56.40674633187   1.83304965161698  0.344713183710671
## ENSG00000000419.11 2311.61509753107 -0.373419784953396 0.0828390167004984
## ENSG00000000457.12 2069.88515433684  0.376458293887623 0.0860125820091379
## ENSG00000000460.15 864.522147486874 -0.812238896522597  0.112023122992459
## ...                             ...                ...                ...
## ENSG00000281883.1  3.63933518666046 0.0230481676343476  0.217488426209903
## ENSG00000281896.1  53.2908748093281  0.268939774867321  0.152935139447986
## ENSG00000281903.1  24.7506455382964  0.770290213483771  0.175801373989849
## ENSG00000281912.1   98.450301361596 -0.423446403741261  0.139085533004298
## ENSG00000281920.1    5.641455151427   1.29108849263691  0.253834348705963
```

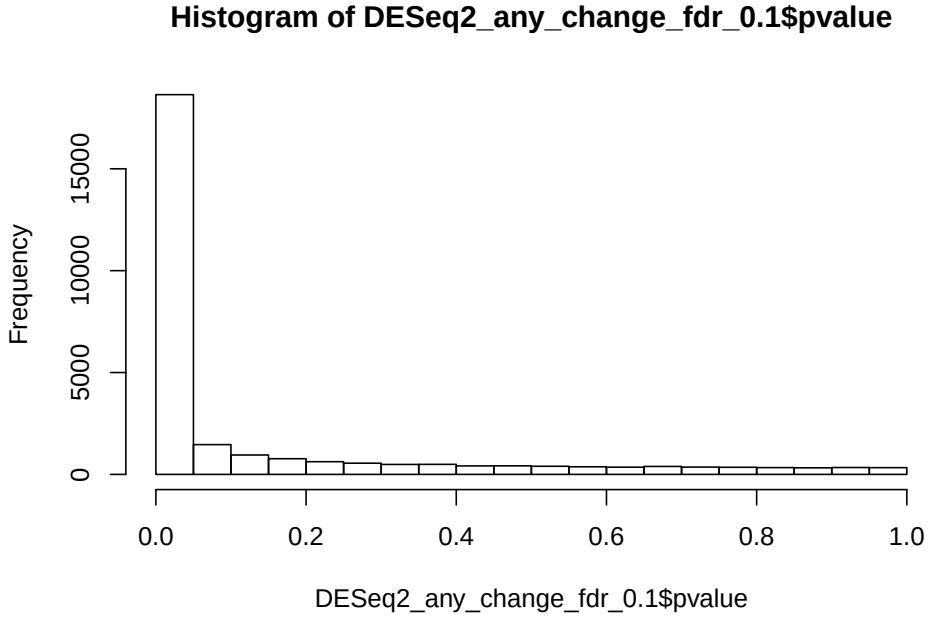
```
# Explaing the columns in result table
mcols(DESeq2_any_change_fdr_0.1, use.names = TRUE)
```

```
## DataFrame with 6 rows and 2 columns
##                        type                                     description
##                <character>                                     <character>
## baseMean       intermediate       mean of normalized counts for all samples
## log2FoldChange      results log2 fold change (MLE): er Positive vs Negative
## lfcSE               results         standard error: er Positive vs Negative
## stat                results         Wald statistic: er Positive vs Negative
## pvalue              results      Wald test p-value: er Positive vs Negative
## padj                results                           BH adjusted p-values
```

```r
# A very high proportion of suggested DEGs
summary(DESeq2_any_change_fdr_0.1)
```

```
##
## out of 28362 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 9053, 32%
## LFC < 0 (down)     : 10212, 36%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```r
# Inflated p-values comparatively to the expected uniform distribution
hist(DESeq2_any_change_fdr_0.1$pvalue)
```



5.5.2 User-defined thresholds

DESeq2 allows customised thresholds to select significant genes. The following chunk of code shows how to select genes that are significant for

- null hypothests that differences are less than 2-fold (log2 FC 1)
- FDR 0.001

The adjusted thresholds suggest less than 10% are diferentially expressed genes: 3.9% up-regulated and 4.8% Down-regulated. This looks much more realistic estimate, than the one suggested using the default settings.

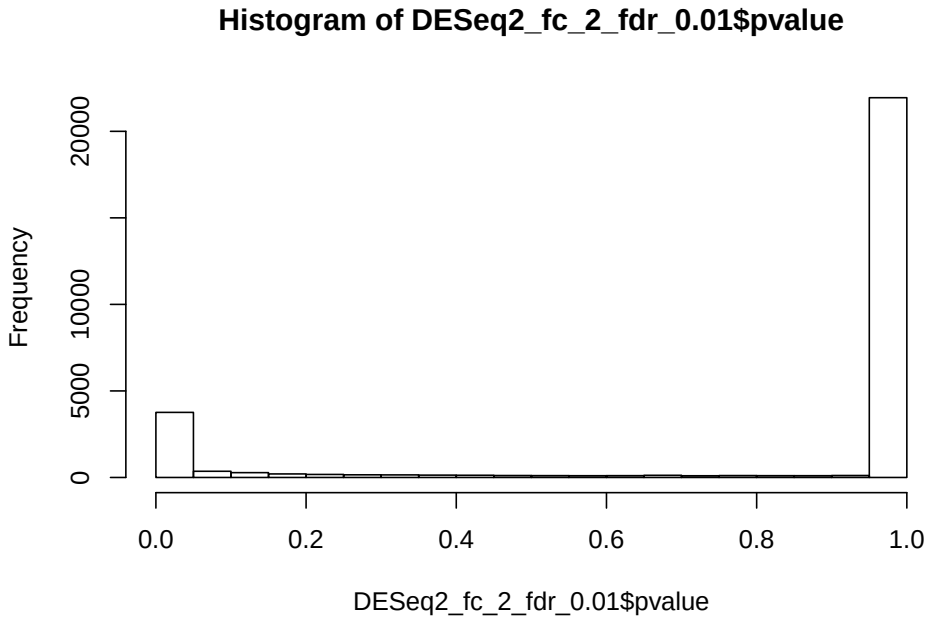
```
# Testing for at least 2-fold difference at FDR 0.001
DESeq2_fc_2_fdr_0.01 <- results(ddx, lfcThreshold=1, alpha=0.01)
DESeq2_fc_2_fdr_0.01
```

```
## log2 fold change (MLE): er Positive vs Negative
## Wald test p-value: er Positive vs Negative
## DataFrame with 28362 rows and 6 columns
##                            baseMean     log2FoldChange             lfcSE
##                           <numeric>          <numeric>         <numeric>
## ENSG00000000003.13 3254.88256154068 -0.690175380631575  0.142786200450243
## ENSG00000000005.5    56.40674633187   1.83304965161698  0.344713183710671
## ENSG00000000419.11 2311.61509753107 -0.373419784953396 0.0828390167004984
## ENSG00000000457.12 2069.88515433684  0.376458293887623 0.0860125820091379
## ENSG00000000460.15 864.522147486874 -0.812238896522597  0.112023122992459
## ...                             ...                ...                ...
## ENSG00000281883.1  3.63933518666046 0.0230481676343476  0.217488426209903
## ENSG00000281896.1  53.2908748093281  0.268939774867321  0.152935139447986
## ENSG00000281903.1  24.7506455382964  0.770290213483771  0.175801373989849
## ENSG00000281912.1   98.450301361596 -0.423446403741261  0.139085533004298
## ENSG00000281920.1    5.641455151427   1.29108849263691  0.253834348705963
```

```
# Check proportion of differentially expressed genes
summary(DESeq2_fc_2_fdr_0.01)
```

```
##
## out of 28362 with nonzero total read count
## adjusted p-value < 0.01
## LFC > 1.00 (up)    : 1118, 3.9%
## LFC < -1.00 (down) : 1358, 4.8%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

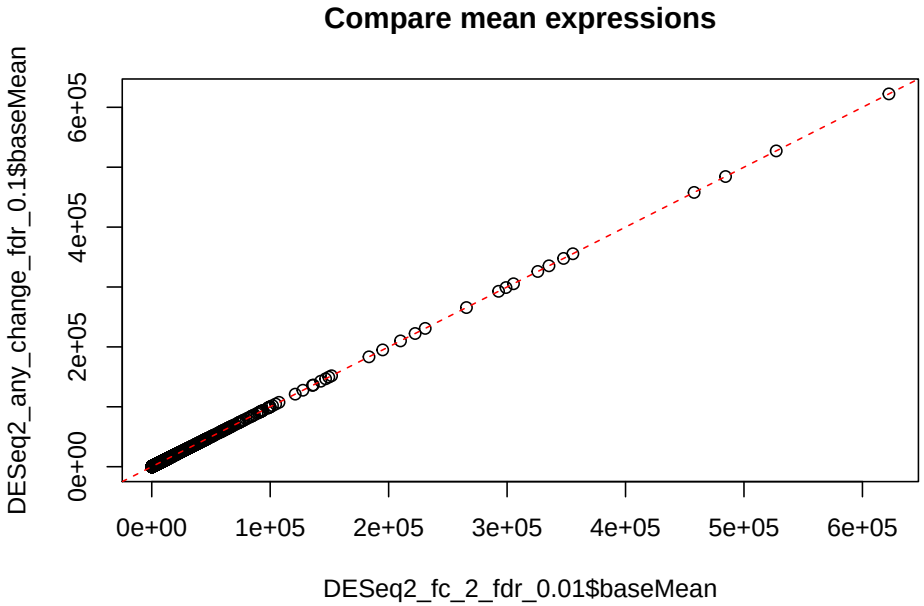
```
# Distribution of p-values:
# not uniform, but without the huge inflation either
hist(DESeq2_fc_2_fdr_0.01$pvalue)
```



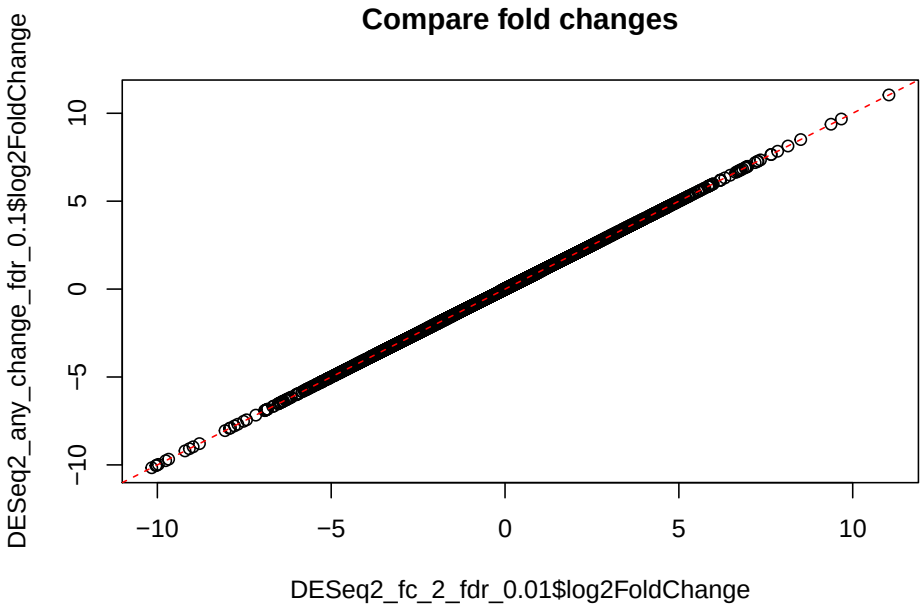
5.5.3 Compare results with different thresholds

Importantly, the above result with adjusted thresholds is NOT equivalent to mere selecting of genes with fold change >2 from the result obtained earlier using the default thersholds. While the fold-change and mean expression, of course, remain the same, the p-values and FDR have been changed, because they are calculated against a different null hypothesis. See help for **DESEq2::results** function for more details.

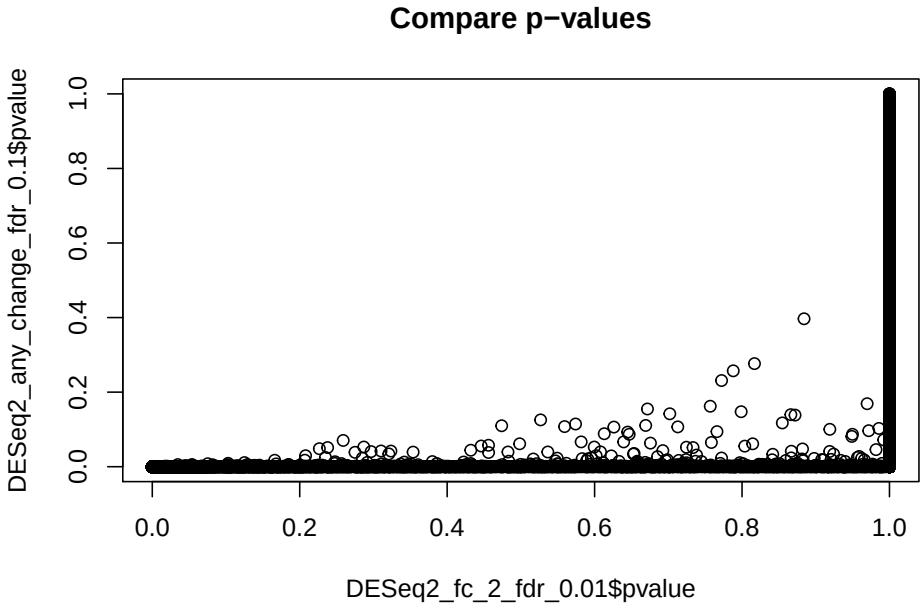
```
plot(DESeq2_fc_2_fdr_0.01$baseMean, DESeq2_any_change_fdr_0.1$baseMean,
     main="Compare mean expressions")
abline(0,1, col="red", lty=2)
```




```
plot(DESeq2_fc_2_fdr_0.01$log2FoldChange, DESeq2_any_change_fdr_0.1$log2FoldChange,
     main="Compare fold changes")
abline(0,1, col="red", lty=2)
```




```
plot(DESeq2_fc_2_fdr_0.01$pvalue, DESeq2_any_change_fdr_0.1$pvalue,
     main="Compare p-values")
```



5.6 Exploring results

In this tutorial, we will explore the results with the modifyed thresholds: at least 2 fold change with FDR 0.01.

These thresholds where selected iteratively, checking the result against a list of selcted known genes, which were reported previously by others for ER-positive breast cancers.

```
rm(DESeq2_any_change_fdr_0.1)
```

5.6.1 Check genes with no p-value

Because of the large number of samples, no genes have been excluded from analysis by DESeq in this example. However, for a smaller number of samples, DESeq2 may exclude some genes, which are very unlikely to be differentially expressed, to improve conditions for multiple testing correction. For such genes the **padj** (FDR) value would be set to **NA**.

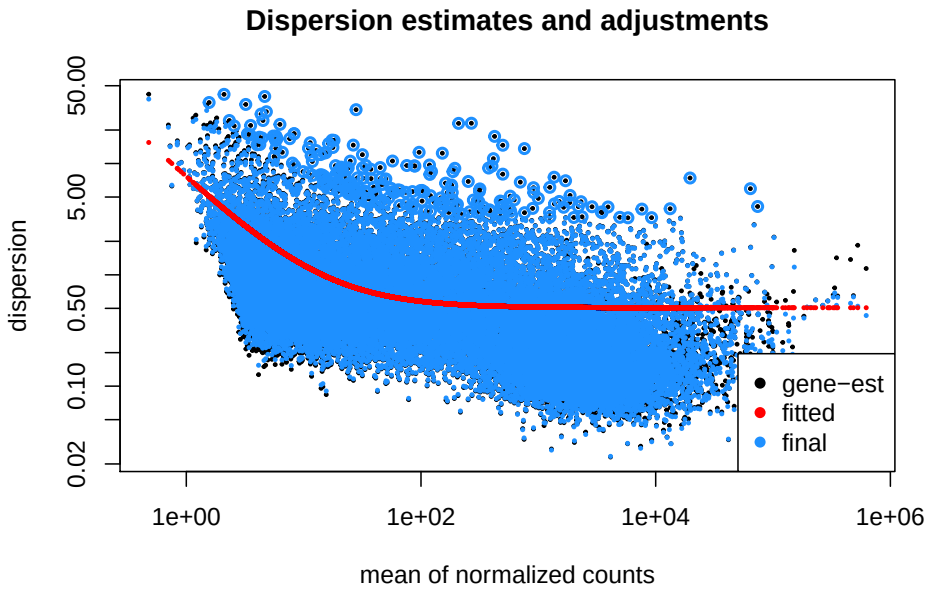
```
sum(is.na(DESeq2_fc_2_fdr_0.01$padj))
```

```
## [1] 0
```

5.6.2 Plot dispersions

Adjustment of dispersions is one of the key statistical tasks in DSEq2. However, in this analysis, because of the large number of samples, the dispersion adjustments were very modest (blue dots overlay the black ones).

```
plotDispEsts(ddx, main="Dispersion estimates and adjustments")
```

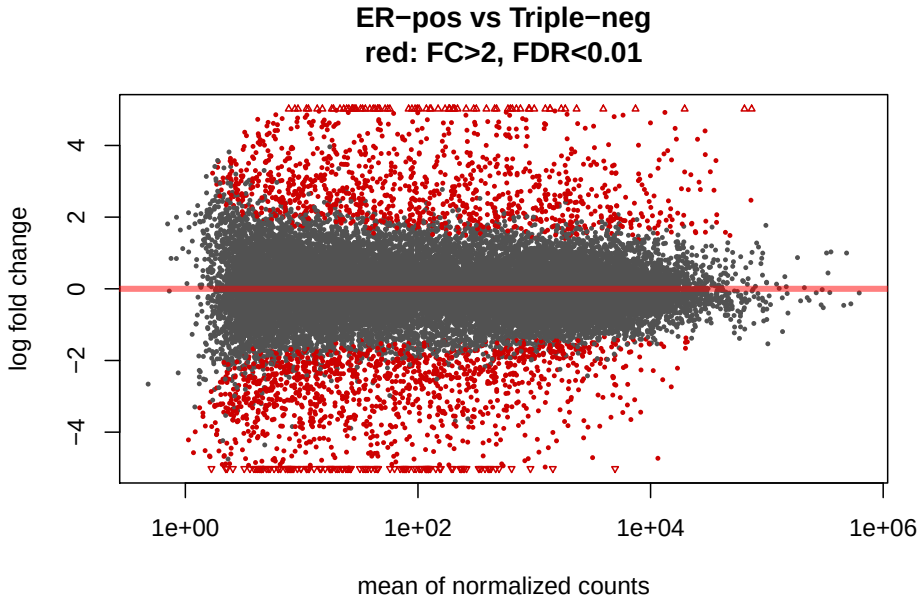


5.6.3 MA plots

MA plot shows fold change along the y-axis, mean expression along the x-axis and significance by colour coding (in red). Each dot represents a gene. The genes equally expressed in both groups (ER-positive and Triple-negative tumours) are located along the middle zero-line. The differentially expressed genes will be located above- or below- the zero-line.

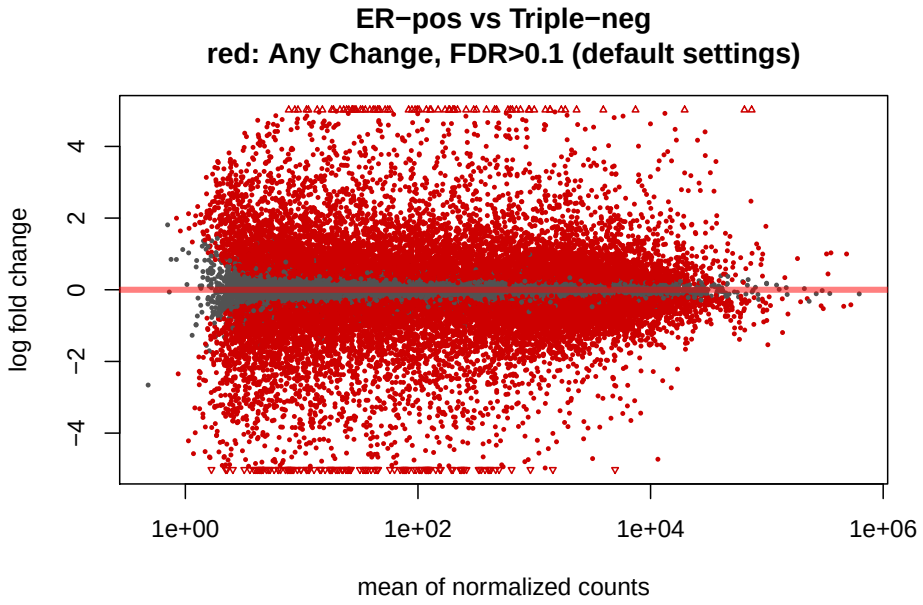
Because of the large number of available samples, a large number of genes have reached statistical "significance" at the default FDR < 0.1 threshold (red dots). This is consistent with the strong biological difference between ER-positive and ER-negative breast cancers. A smaller, but still very large number of genes have reached "significance" at FDR $<$ 10-6 level.

```
plotMA(ddx, lfcThreshold=1, alpha=0.01, main="ER-pos vs Triple-neg\nred: FC>2, FDR<0.0
```



This would be the MA-plot for the **default** DESeq2 settings:

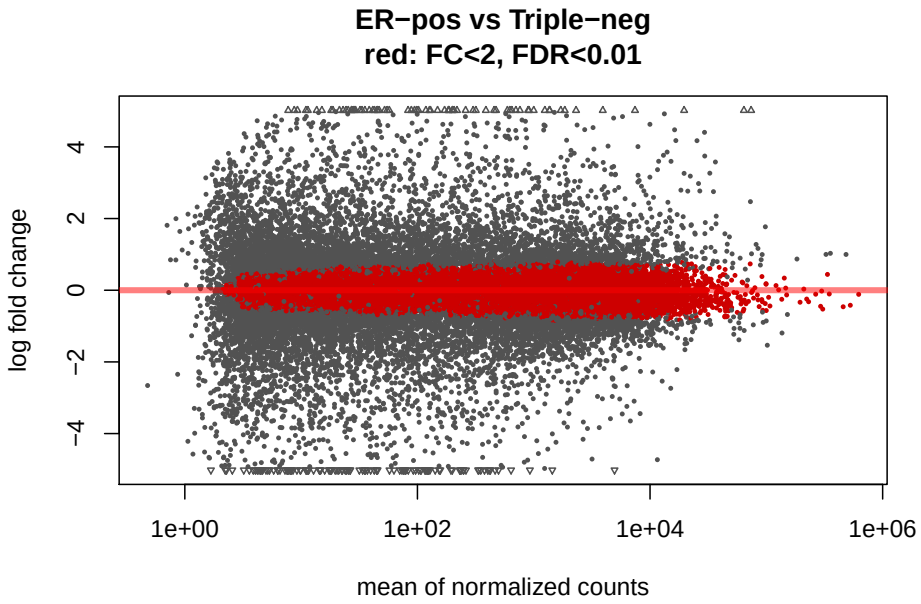
```
plotMA(ddx, main="ER-pos vs Triple-neg\nred: Any Change, FDR>0.1 (default settings)")
```



By default, of course, the null hypothesis is either of no difference, or that the difference is below the specified logFC threshold. That means the alternative hypothesis is that the changes are above the threshold.

A very interesting and unusual feature of DESeq2 is that it allows to test for alternative hypothesis that the change is **LESS** than a specified threshold. In other words, DESeq2 provides **a statistical framework to look for unchanged genes**:

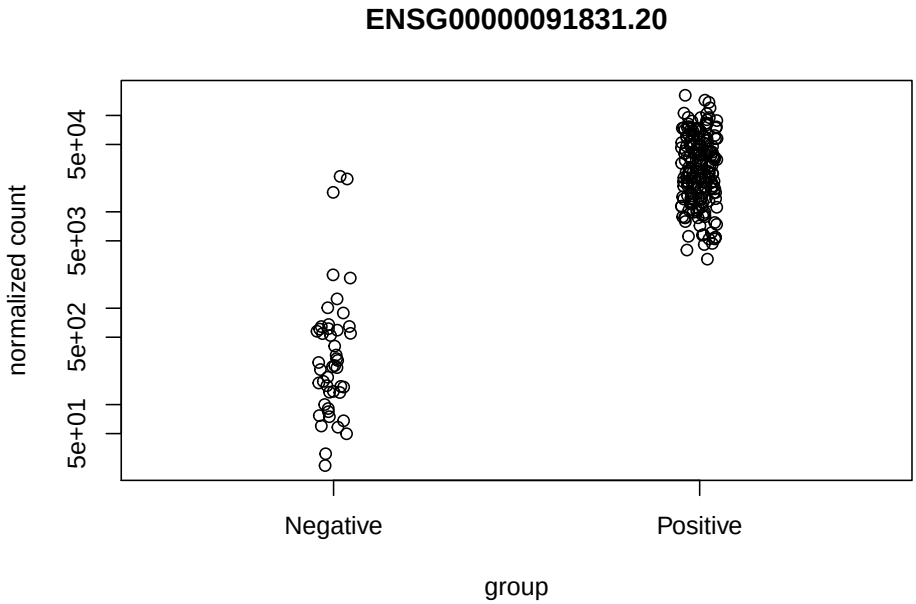
```
plotMA(ddx, lfcThreshold=1, altHypothesis="lessAbs", alpha=0.01,
       main="ER-pos vs Triple-neg\nred: FC<2, FDR<0.01")
```



5.6.4 Plot counts for an individual gene

In this tutorial we will plot counts for Estrogen Receptor Alpha (ESR1) comparing ER-positive and Triple-negative groups.

```
# Get gene ID for ESR1
ESR1_gene_id <- genes.df %>%
  filter(gene_name == "ESR1") %>%
  select(gene_id)

# Make the plot
plotCounts(ddx, gene=as.character(ESR1_gene_id), intgroup = "er")
```




```
# Clean-up
rm(ESR1_gene_id)
```

5.6.5 Convert results to data-frame

```
# The results can be converted to data.frame
DESeq2_fc_2_fdr_0.01_all_genes.df <- as.data.frame(DESeq2_fc_2_fdr_0.01)
head(DESeq2_fc_2_fdr_0.01_all_genes.df)
```

```
##                      baseMean log2FoldChange      lfcSE     stat     pvalue      padj
## ENSG00000000003.13 3254.88256     -0.6901754 0.14278620 0.000000 1.00000000 1.0000000
## ENSG00000000005.5    56.40675      1.8330497 0.34471318 2.416646 0.01566425 0.1357792
## ENSG00000000419.11 2311.61510     -0.3734198 0.08283902 0.000000 1.00000000 1.0000000
## ENSG00000000457.12 2069.88515      0.3764583 0.08601258 0.000000 1.00000000 1.0000000
## ENSG00000000460.15  864.52215     -0.8122389 0.11202312 0.000000 1.00000000 1.0000000
## ENSG00000000938.11  621.25238     -0.1733090 0.14869825 0.000000 1.00000000 1.0000000
```

```
# Copy gene ID-s from the rownames to a column
DESeq2_fc_2_fdr_0.01_all_genes.df$gene_id=rownames(DESeq2_fc_2_fdr_0.01_all_genes.df)
head(DESeq2_fc_2_fdr_0.01_all_genes.df)
```

```
##                      baseMean log2FoldChange      lfcSE     stat     pvalue      padj
## ENSG00000000003.13 3254.88256     -0.6901754 0.14278620 0.000000 1.00000000 1.0000000 ENSG0000
## ENSG00000000005.5    56.40675      1.8330497 0.34471318 2.416646 0.01566425 0.1357792  ENSG000
## ENSG00000000419.11 2311.61510     -0.3734198 0.08283902 0.000000 1.00000000 1.0000000 ENSG0000
## ENSG00000000457.12 2069.88515      0.3764583 0.08601258 0.000000 1.00000000 1.0000000 ENSG0000
```

```
## ENSG00000000460.15  864.52215     -0.8122389 0.11202312 0.000000 1.00000000 1.000000
## ENSG00000000938.11  621.25238     -0.1733090 0.14869825 0.000000 1.00000000 1.000000
```

```
# Add gene names (using information obtained from the GTF file earlier)
DESeq2_fc_2_fdr_0.01_all_genes.df <-left_join(DESeq2_fc_2_fdr_0.01_all_genes.df, genes
head(DESeq2_fc_2_fdr_0.01_all_genes.df)
```

```
##     baseMean log2FoldChange      lfcSE     stat     pvalue      padj            gene
## 1 3254.88256     -0.6901754 0.14278620 0.000000 1.00000000 1.0000000 ENSG00000000003
## 2   56.40675      1.8330497 0.34471318 2.416646 0.01566425 0.1357792  ENSG00000000005
## 3 2311.61510     -0.3734198 0.08283902 0.000000 1.00000000 1.0000000 ENSG00000000419
## 4 2069.88515      0.3764583 0.08601258 0.000000 1.00000000 1.0000000 ENSG00000000457
## 5  864.52215     -0.8122389 0.11202312 0.000000 1.00000000 1.0000000 ENSG00000000460
## 6  621.25238     -0.1733090 0.14869825 0.000000 1.00000000 1.0000000 ENSG00000000938
```

```
dim(DESeq2_fc_2_fdr_0.01_all_genes.df)
```

```
## [1] 28362     8
```

Make a separate dataframe with the differentially expressed genes only: expression change at least 2-fold with FDR 0.01

```
# Select Differentially Expressed Genes only
DESeq2_fc_2_fdr_0.01_DEGs.df <- DESeq2_fc_2_fdr_0.01_all_genes.df %>%
  filter(-log10(padj) > 2, abs(log2FoldChange) > 1) %>%
  select(gene_name, gene_id, baseMean, log2FoldChange, padj) %>%
  arrange(desc(log2FoldChange))

# Check results
dim(DESeq2_fc_2_fdr_0.01_DEGs.df)
```

```
## [1] 2476    5
```

```
head(DESeq2_fc_2_fdr_0.01_DEGs.df)
```

```
##       gene_name            gene_id   baseMean log2FoldChange         padj
## 1        CARTPT  ENSG00000164326.4   760.7236      11.041924 5.602005e-26
## 2          CPB1 ENSG00000153002.10 64177.1952       9.673386 3.075352e-48
## 3  RP11-53O19.2  ENSG00000248779.1   301.5455       9.376603 3.960013e-46
## 4 RP11-473L15.3  ENSG00000249203.1   171.1401       8.504287 3.198874e-43
## 5  RP11-680B3.2  ENSG00000240521.1   118.1529       8.138256 6.406333e-18
## 6        CLEC3A  ENSG00000166509.9 19635.8020       7.839747 2.696858e-24
```

```
# Clean-up
rm(DESeq2_fc_2_fdr_0.01)
```

5.6.6 Check known genes

The list of genes up-regulated in ER-positive breast cancers is not yet definitevely established. Numerous papers report different lists of genes in cell lines or clinical biopsies, using different study designs, experimental methods etc. ESR1, PGR, TFF1, TFF3, FOXA1, GATA3 are amongst the most consistently mentioned genes up-regulated in ER-positive cancers. Reports about the genes down-regulated in ER-positive breast cancers are even less consistent, although FOXC1, MIA had been mentioned earlier in this context.

The changes in expression of selected known ER-associated genes confirm that our findings are considtent with previous reports. In additon, our analysis suggests many other genes, not yet reported in context of ER signalling in breast cancer. The newly reported genes need to be validated in an independent dataset before making definitive conclusions.

```
# Make a list of selected previously known genes of interest
selected_known_genes=c("ESR1", "PGR", "TFF1", "TFF3", "GATA3", "FOXA1", "FOXC1", "MIA")

# Look at the genes of interest in the top DEGs in our dataset
DESeq2_fc_2_fdr_0.01_DEGs.df %>%
  filter(gene_name %in% selected_known_genes)
```

```
##   gene_name            gene_id   baseMean log2FoldChange         padj
## 1      TFF1  ENSG00000160182.2 16259.9568       4.476394 9.765610e-15
## 2      ESR1 ENSG00000091831.20 29402.2500       4.406085 3.070201e-44
## 3      TFF3 ENSG00000160180.15 16669.3405       4.175744 1.344767e-18
## 4       PGR ENSG00000082175.13 11860.4295       3.597820 2.272716e-14
## 5     FOXA1  ENSG00000129514.5 18425.8831       2.792643 8.023961e-16
## 6     GATA3 ENSG00000107485.14 35098.3567       2.750493 1.204231e-22
## 7     FOXC1  ENSG00000054598.6  1822.2110      -3.702730 2.674219e-47
## 8       MIA  ENSG00000261857.5   296.8554      -3.865913 3.768407e-17
```

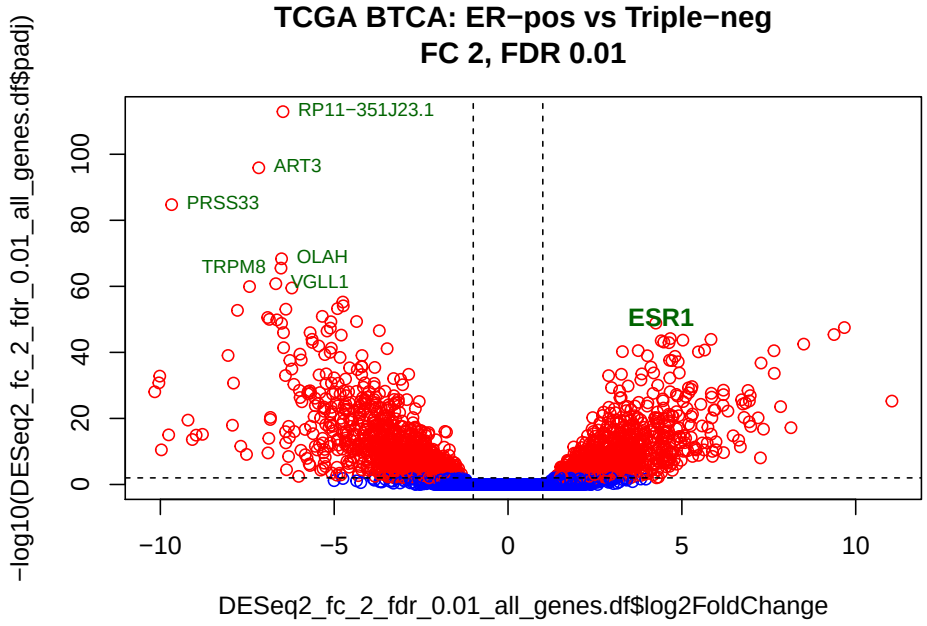
```
# Clean-up
rm(selected_known_genes)
```

5.6.7 Volcano plot

A **volcano plot** allows simultaneous visualisation of significance (y axis) and fold change (x axis).

```
# Prepare colour-coding for genes: highlit the DEGs in red
genes_color <- rep("blue",nrow(DESeq2_fc_2_fdr_0.01_all_genes.df))
"red" -> genes_color[DESeq2_fc_2_fdr_0.01_all_genes.df$gene_id %in% DESeq2_fc_2_fdr_0.01_DEGs.df$

# Make plot
plot(x=DESeq2_fc_2_fdr_0.01_all_genes.df$log2FoldChange,
```

```
     y=-log10(DESeq2_fc_2_fdr_0.01_all_genes.df$padj),
     col=genes_color,
     main="TCGA BTCA: ER-pos vs Triple-neg\nFC 2, FDR 0.01")

# Add lines to show thresholds for the top genes
abline(h=2, lty=2) # FDR 0.01
abline(v=-1, lty=2) # FC < -2
abline(v=1, lty=2) # FC > 2

# Label Estrogen Receptor Alpha gene
esr1.df <- DESeq2_fc_2_fdr_0.01_all_genes.df %>% filter(gene_name == "ESR1")
text(esr1.df$log2FoldChange, -log10(esr1.df$padj), "ESR1", font=2, col="darkgreen", pos

# Label most down-regulated genes
down_regulated_genes.df <- DESeq2_fc_2_fdr_0.01_all_genes.df %>%
                  filter(-log10(padj)>60) %>%
                  select(gene_name, log2FoldChange,padj)

text(down_regulated_genes.df$log2FoldChange,
     -log10(down_regulated_genes.df$padj),
     down_regulated_genes.df$gene_name,
     col="darkgreen", pos=c(4,4,2,4,4,4), cex=0.75)
```



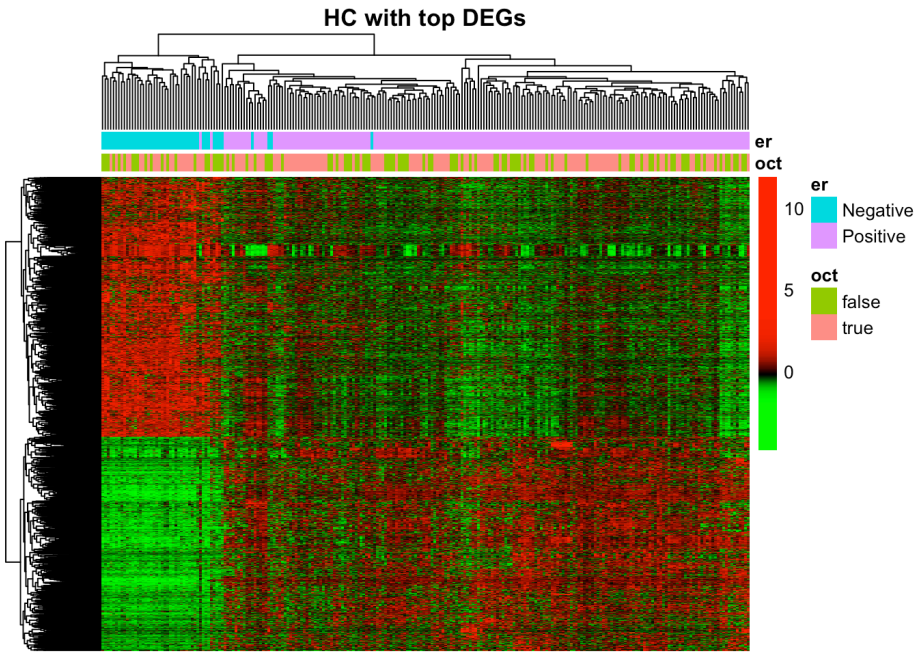

```
# Clean-up
rm(genes_color, esr1.df, down_regulated_genes.df)
```

5.6.8 Hierarchical clustering using DEGs

Hierarchical clustering is also informative to show relations between samples and top differentially expressed genes

```
# Libraries for drawing heatmap
#install.packages("RColorBrewer")
#install.packages("pheatmap")
#library(RColorBrewer)
#library(pheatmap)

# Matrix for clustering and heatmap
vst_counts <- assay(vst_dds)
DEGs <- rownames(vst_counts) %in% DESeq2_fc_2_fdr_0.01_DEGs.df$gene_id
vst_counts <- vst_counts[DEGs,]

# Scale and center by rows
vst_counts <- t(scale(t(vst_counts)))

# Change genes IDs to genes Names
rownames(DESeq2_fc_2_fdr_0.01_DEGs.df) <- DESeq2_fc_2_fdr_0.01_DEGs.df$gene_id
rownames(vst_counts) <- DESeq2_fc_2_fdr_0.01_DEGs.df[rownames(vst_counts),"gene_name"]

# Data frame with ER- and OCT- status
cases.df <- as.data.frame(colData(vst_dds))
cases.df <- cases.df %>% select(oct, er)

# Make palette of 99 colours for heatmap
gbr_99 <- colorRampPalette(c("green", "black", "red"))(n = 99)

# Make breaks between colours by 100 quantiles
breaks_100 <- quantile(vst_counts, probs=seq(0,1,length.out=100))

# Plot the heatmap and dendrograms
# Row and column names couls be shown if there was a smaller number of genes and samples
# Options "fontsize_row" and "fontsize_col" could be used for row/column labels adjustment
pheatmap(
  mat = vst_counts, # matrix with data
  color = gbr_99, # user-defined palette
  breaks = breaks_100, # user-defined breaks between colours
  annotation_col = cases.df, # ER- and OCT- status
  scale = "none", # has been scaled manually
  border_color = NA, # dont draw borders in heatmap
  show_rownames = FALSE, # too many genes to show names
  show_colnames = FALSE, # too many samples to show names
  main = "HC with top DEGs")
```

```
# Clean-up
rm(vst_counts, gbr_99, breaks_100, cases.df, DEGs)
```



5.7 Save results

You can use **write.table** function to save DEGs (and results for all genes) to a text file:

```
# Save DEGs
write.table(DESeq2_fc_2_fdr_0.01_DEGs.df,
            file=file.path(base_folder,"analysis","results","DESeq2_fc_2_fdr_0_01_DEGs
            quote=F, sep="\t", row.names = F)

# Save all genes
write.table(DESeq2_fc_2_fdr_0.01_all_genes.df,
            file=file.path(base_folder,"analysis","results","DESeq2_fc_2_fdr_0_01_all_g
            quote=F, sep="\t", row.names = F)
```

Chapter 6

edgeR analysis

First, we load edgeR library:

```
#install.packages("BiocManager")
#BiocManager::install("edgeR")
suppressWarnings(suppressMessages(library(edgeR)))
```

6.1 Read HTSeq counts to DGEList

Prior the analysis **edgeR** needs to collect all necessary information. **edgeR** collects the counts, samples and genes data in a single list called **DGEList**. To facilitate the data input **edgeR** provides a function called **readDGE** that makes a single matrix from multiple files, allows adding sample information etc.

The function can read tabulated text files of arbitrary format, as long as one of the file's columns contains genes names and another column contains the counts. Format of an HTSeq counts file can be explored using **Bash** functions **head** and **tail**:

```
## ENSG00000000003.13   2569
## ENSG00000000005.5    1
## ENSG00000000419.11   3180
## ENSG00000000457.12   3332
## ENSG00000000460.15   1621
## ENSG00000000938.11   530
## ENSG00000000971.14   7282
## ENSG00000001036.12   3312
## ENSG00000001084.9    2642
## ENSG00000001167.13   3322
## ...
```

```
## ENSGR0000275287.3    0
## ENSGR0000276543.3    0
## ENSGR0000277120.3    0
## ENSGR0000280767.1    0
## ENSGR0000281849.1    0
## __no_feature 3069305
## __ambiguous  3368739
## __too_low_aQual  0
## __not_aligned    0
## __alignment_not_unique   23748640
```

It can be seen that an **HTSeq counts file** fits **readDGE** requirements, except for some summary lines at the end of the file. Luckily, **readDGE** can take care about these summary lines with **comment.char** option. Finally, **header = FALSE** should be used to specify that the 1st row contains data, not the header:

```
# Counts folder
counts_folder=file.path(base_folder,"data","HTSeq_counts")

# Read HTSeq counts to DGE object
# ("dgl" stands for "DGEList")
dgl <- readDGE(files=samples.df$file, # files names
        path=counts_folder, # folder with count files
        columns=c(1,2), # columns with gene name (1) and count (2)
        labels=samples.df$patient, # samples names corresponding to the files
        comment.char="_", # lines starting with this character are excluded
        header = FALSE) # the 1st row contains data, not header

# Clean-up
rm(counts_folder)
```

See **readDGE** help for details of other options used in the above command.

6.2 Explore and update the DEGList object

The code chunk below

- checks content of the counts' matrix read to the DEGList object
- checks samples information and adds OCT-status as an example of potential experimental confounder
- adds genes information to the DEGList object

See **DEGList** help for details.

```
# Check content of created DEGList
names(dgl)
```

```
## [1] "samples" "counts"
```

```
# The counts matrix is identical to the counts in DESeq2DataSet
dim(dgl$counts)
```

```
## [1] 60483   238
```

```
dgl$counts[1:5,1:5]
```

```
##                     Samples
## Tags                 TCGA-A7-A0DA TCGA-D8-A1XU TCGA-D8-A143 TCGA-A7-A4SB TCGA-D8-A1XR
##   ENSG00000000003.13         2724         5645         6180         2558         3884
##   ENSG00000000005.5             7            6            1           32          109
##   ENSG00000000419.11         1962         4926         2624         1068         3337
##   ENSG00000000457.12         1973         2271         1860         1362         4953
##   ENSG00000000460.15          867          675         2359          494         2284
```

```
# Add information about the exparimental groups for DEG detection (ER status)
head(dgl$samples)
```

```
##                                                          files group lib.size norm.factors
## TCGA-A7-A0DA a33029dd-b5fa-4be0-9cbf-971d289146dd.htseq.counts.gz     1 66687895            1
## TCGA-D8-A1XU 8d54214a-1d9b-4fea-9c42-5bbb3cd11da9.htseq.counts.gz     1 93109355            1
## TCGA-D8-A143 4b19c0e2-2a61-4f0a-9257-4a528e6b320e.htseq.counts.gz     1 55477594            1
## TCGA-A7-A4SB cc233ff9-d5fb-4e9b-9007-f58d008df995.htseq.counts.gz     1 38870554            1
## TCGA-D8-A1XR bdd8c340-250b-474a-8802-7653b7884ced.htseq.counts.gz     1 82295968            1
## TCGA-BH-A18L 5bc7e90d-0fa4-4bde-bee8-4f9b92de03a2.htseq.counts.gz     1 71552576            1
```

```
# Add ER and OCT status to the samples information
dgl$samples$er <- samples.df$er
dgl$samples$oct <- samples.df$oct
head(dgl$samples)
```

```
##                                                          files group lib.size norm.factors
## TCGA-A7-A0DA a33029dd-b5fa-4be0-9cbf-971d289146dd.htseq.counts.gz     1 66687895            1
## TCGA-D8-A1XU 8d54214a-1d9b-4fea-9c42-5bbb3cd11da9.htseq.counts.gz     1 93109355            1
## TCGA-D8-A143 4b19c0e2-2a61-4f0a-9257-4a528e6b320e.htseq.counts.gz     1 55477594            1
## TCGA-A7-A4SB cc233ff9-d5fb-4e9b-9007-f58d008df995.htseq.counts.gz     1 38870554            1
## TCGA-D8-A1XR bdd8c340-250b-474a-8802-7653b7884ced.htseq.counts.gz     1 82295968            1
## TCGA-BH-A18L 5bc7e90d-0fa4-4bde-bee8-4f9b92de03a2.htseq.counts.gz     1 71552576            1
```

```
# Add genes information
genes.df <- genes.df[rownames(dgl$counts),] # Reorder genes to match the genes order in the count
dgl$genes <- genes.df
names(dgl)
```

```
## [1] "samples" "counts"  "genes"
```

6.3 Gene filtering

In **DESeq2** analysis we manually removed genes, for which less than 10 samples had count above 10. This reduced the nimber of genes from ~60k to ~28k.

edgrR has a dedicated build-in function **filterByExpr** to remove low-expressed genes. This function also keeps the genes with a count above 10 in a certain number of samples, depending in the groups size. Applying this function to our TCGA-BRCA dataset preserves ~24k of genes, which is reasonably close to the manual filtering applyed in DESeq2 analysis.

```
# Select genes with sufficien expression for comparison in ER groups
keep.exprs <- filterByExpr(dgl, group=dgl$samples$er)
sum(keep.exprs)
```

```
## [1] 24361
```

```
# Remove low-expressed genes
dgl <- dgl[keep.exprs,, keep.lib.sizes=FALSE]
dim(dgl)
```

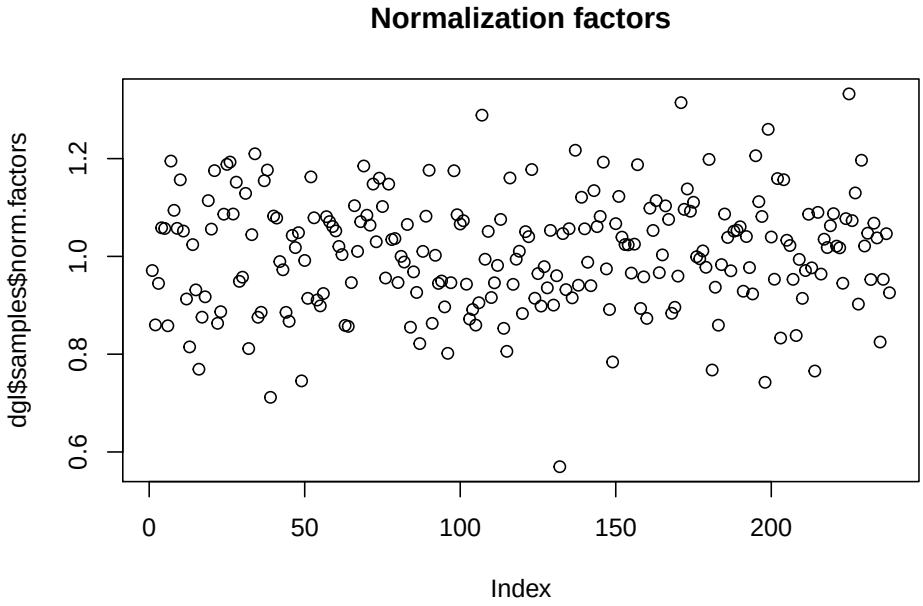
```
## [1] 24361   238
```

```
# Clean-up
rm(keep.exprs)
```

6.4 Normalizing by TMM

edgeR's preferred method of normalization is **Trimmed Means of M-values** (TMM)

```
# Aply normalization
dgl <- calcNormFactors(dgl, method = "TMM")

# Look at the calculated TMM notmalization factors
plot(dgl$samples$norm.factors, main="Normalization factors")
```



6.5 Exploring source data

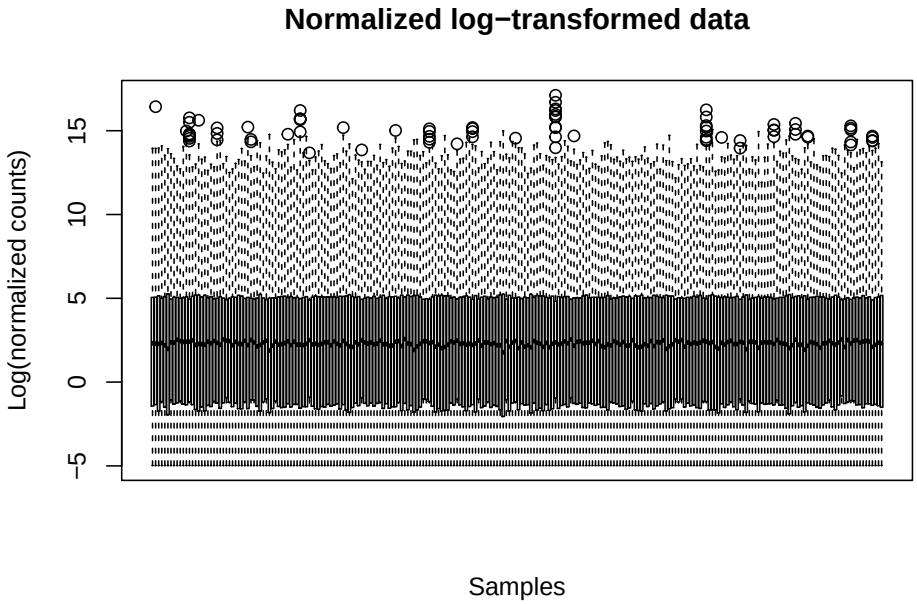
Like in **DESeq2** analysis, the source data is explored in a normalized form. For this, **edgeR** allows to calculate log-transformed counts per million:

```
lcpm <- cpm(dgl, log=TRUE)
```

6.5.1 Boxplot

The boxplot shows that counts range and central positions are similar in different samples after the normalization:

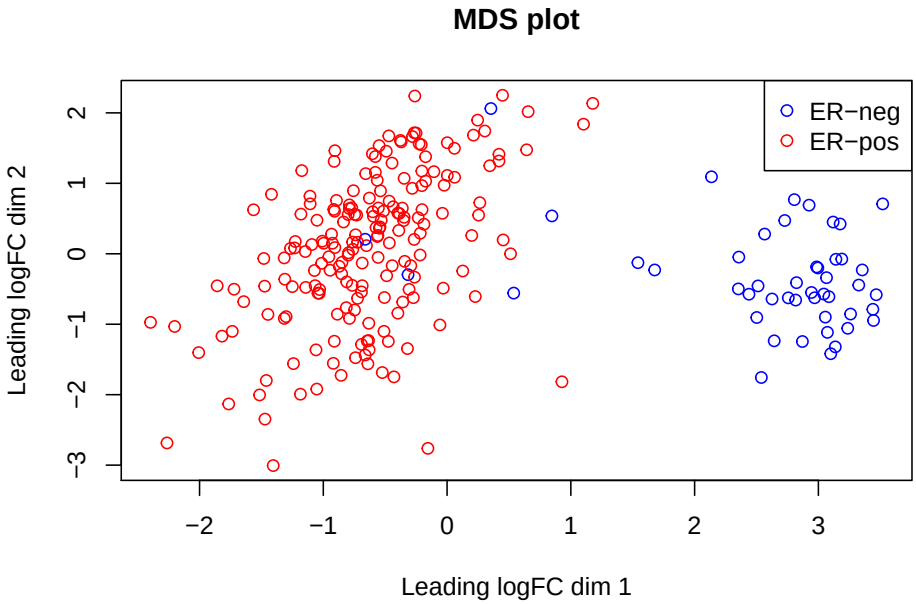
```
boxplot(lcpm, xaxt="n", main="Normalized log-transformed data",
        xlab="Samples",ylab="Log(normalized counts)")
```



6.5.2 MDS plot

Multi Dimentional Scaling (MDS) is similar to PCA: it places similar samples close to each other in a 2-dimention plot. **edgeR** provides a function for making a MDS plot. The plot shows a clear separation of ER-positive and Triple-negative breast cancers by the genes expression.

```
# Colour code ER status
col_er <- as.factor(dgl$samples$er)
levels(col_er) <- c("blue","red")
col_er <- as.character(col_er)

# Make MDS plot
plotMDS(lcpm, labels = NULL, pch = 1, col=col_er,
        main="MDS plot")

legend("topright", # the location of the legend on the plot
       legend = c("ER-neg", "ER-pos"), # labels
       col = c("blue","red"),
       pch = 1) # colours
```




```
# Clean-up
rm(lcpm, col_er)
```

6.6 Calculate DEGs

6.6.1 Specify design

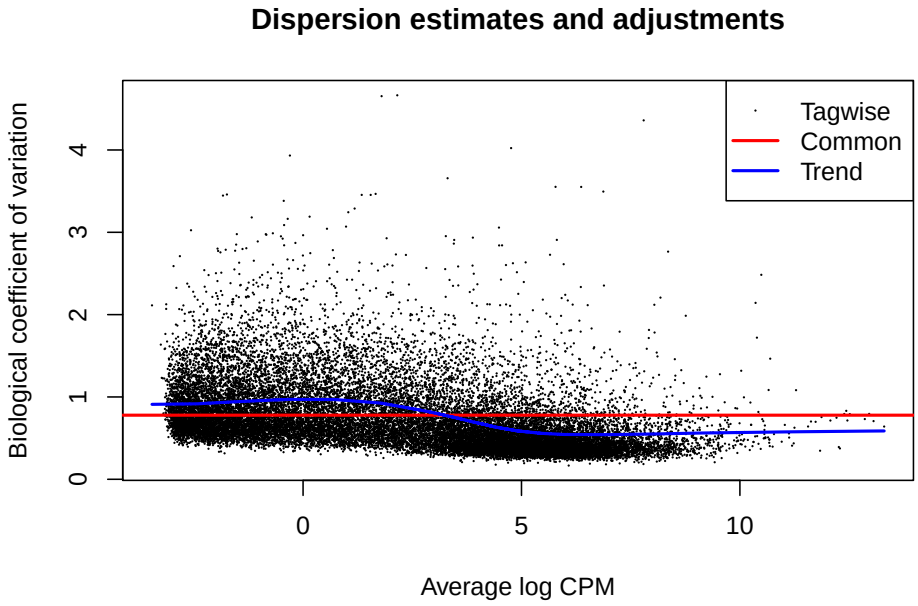
Like in **DESeq2** the design in **edgeR** is defined by a text string that follows the format of a **glm** formula in R: ~ **oct** + **er**. It is used to specify the variable of interest for differential expression ("er"" in our case) and the confounding variables, if any ("oct" in our case). By convention, the variable of interest should be placed is at the end of the design formula.

```
design <- model.matrix(~ oct + er, data = dgl$samples)
```

6.6.2 Estimate dispersions

Like in **DESeq2**, the variance estimation and adjustment is a core element of building the **edgeR** statistical model:

```
dgl <- estimateDisp(dgl, design)
plotBCV(dgl, main="Dispersion estimates and adjustments")
```



6.6.3 Fit the model

```
fit <- glmQLFit(dgl, design)
```

6.7 Extract results

The gene-wise statistical significance is extracted from the model, which has been fitted in the previous code chunk.

6.7.1 Default settings

By default edgeR calculates tests against the null hypothesis of no change (i.e. **alt hypothesis of any change**). The key results are written in the **table** slot of the specialised **TopTags** result object.

Note that in **glmQLFTest** function call the **coef=ncol(design)** points to the last column in the design table. You may remember that,by convention, the variable of interest was placed at the end of the design formula.

The **topTags** function actually prepared the table with results. By default, though, it will only show the top 10 genes… To extract results for all the genes **n = nrow(x)** option is used.

```
# Calculate the test statistics
edgeR_any_change <- glmQLFTest(fit, coef = ncol(design))

# Extract statistics for all genes
edgeR_any_change_all_genes <- topTags(edgeR_any_change, n = nrow(dgl), sort.by = "none")
names(edgeR_any_change_all_genes)
```

```
## [1] "table"         "adjust.method" "comparison"    "test"
```

```
# Explore table with results
dim(edgeR_any_change_all_genes$table)
```

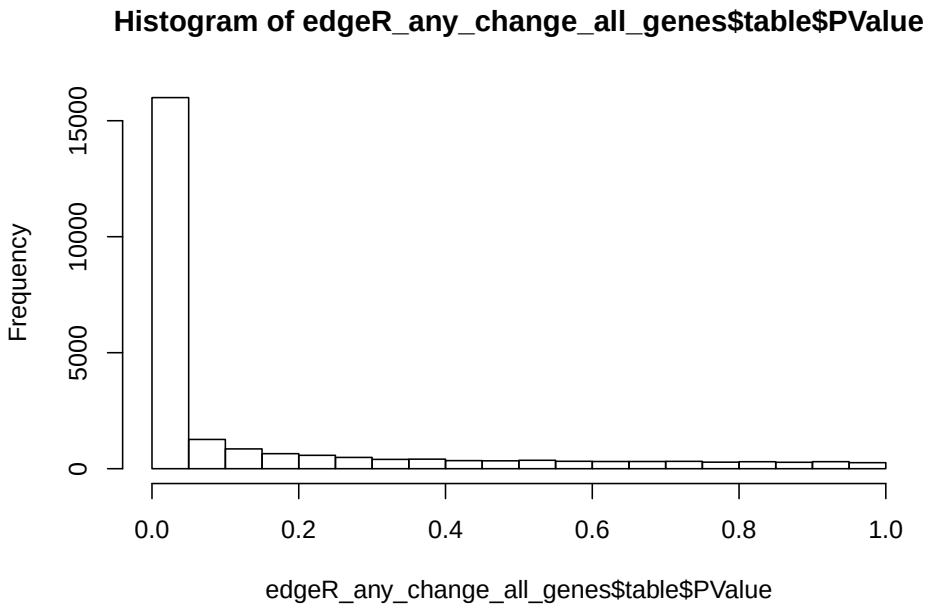
```
## [1] 24361     7
```

```
head(edgeR_any_change_all_genes$table)
```

```
##                                gene_id gene_name      logFC      logCPM        F       PValue
## ENSG00000000003.13 ENSG00000000003.13    TSPAN6 -0.6784647  5.73670542 23.82900 1.925960e-06 6
## ENSG00000000005.5   ENSG00000000005.5      TNMD  1.8310633 -0.08050533 18.95597 1.984714e-05 5
## ENSG00000000419.11 ENSG00000000419.11      DPM1 -0.3582450  5.25058484 18.56443 2.400762e-05 6
## ENSG00000000457.12 ENSG00000000457.12     SCYL3  0.3891598  5.09050998 18.72840 2.216723e-05 6
## ENSG00000000460.15 ENSG00000000460.15  C1orf112 -0.8041035  3.83330744 54.05200 3.092747e-12 2
## ENSG00000000938.11 ENSG00000000938.11       FGR -0.1688074  3.34761965  1.27565 2.598416e-01 3
```

Like in **DESeq2** analysis, the default settings lead to an inflated number of **significant** p-values (if compared to the expected uniform distribution of p-values under the null):

```
# Plot histogram of p-values
hist(edgeR_any_change_all_genes$table$PValue)
```



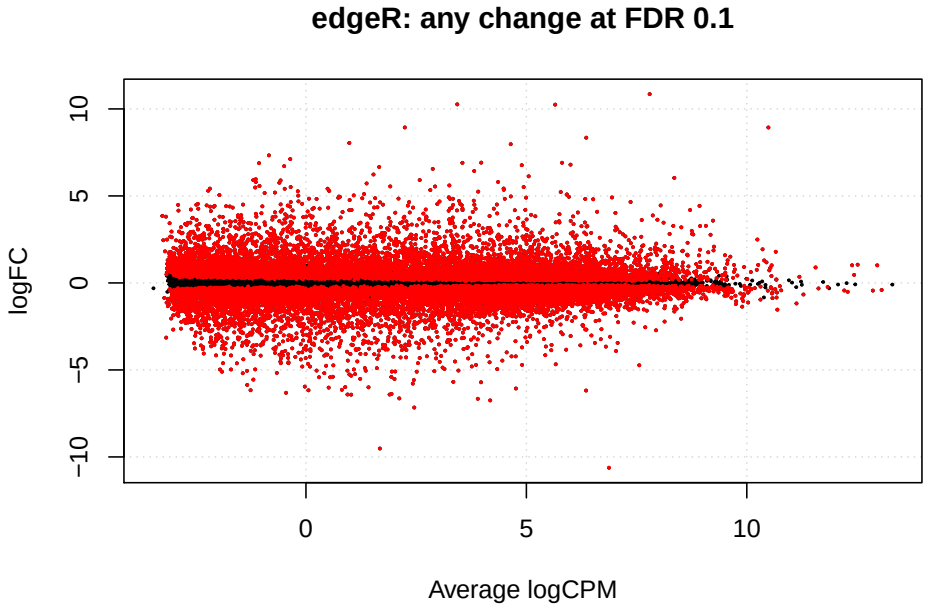
By default, the **topTags** function does not apply a filter by p-value. However, such option exists. In the chunk below we apply a FDR 0.1 filter, by analogy with the default FDR threshold in **DESeq2**. Like in **DESeq2** analysis such thersholds would suggest that more than half of the genes are differentially expressed:

```
# Select genes with FDR 0.1
edgeR_any_change_fdr_0.1 <- topTags(edgeR_any_change, n = nrow(dgl), p.value = 0.1)
dim(edgeR_any_change_fdr_0.1$table)
```

```
## [1] 16575     7
```

MA plot in **edgeR** is plotted by **plotSmear** function:

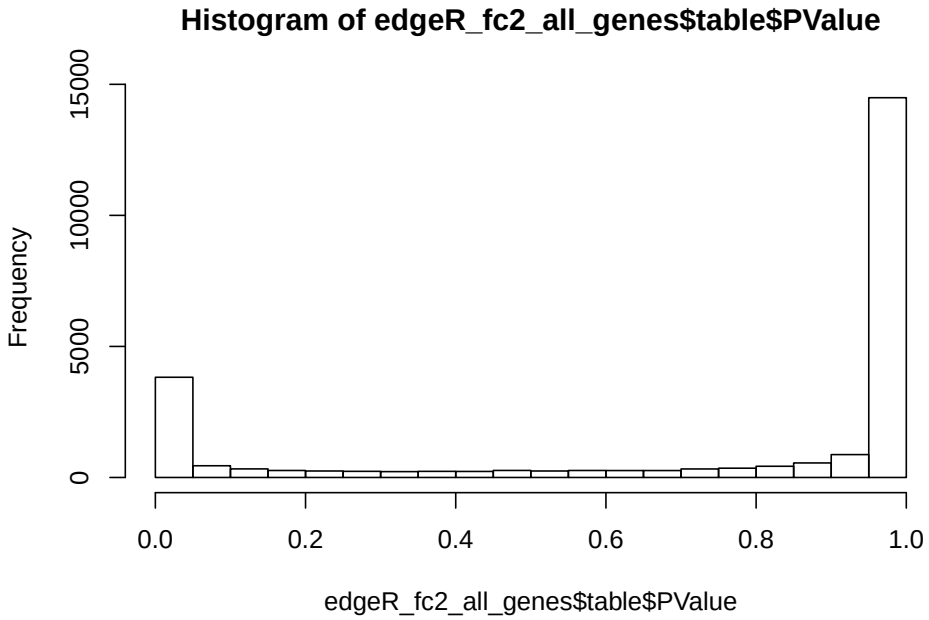
```
plotSmear(edgeR_any_change, de.tags = edgeR_any_change_fdr_0.1$table$gene_id,
          main="edgeR: any change at FDR 0.1")
```



6.7.2 Customised thresholds

Like in the **DESeq2** analysis, **edgeR** allows to calculate p-values for the alternative hypothesis of at least **2 fold difference** at **FDR 0.01**. Like in the **DESeq2** analysis, this improves the p-values distribution and reduces the number of suggested DEGs to less than 10% of the genes:

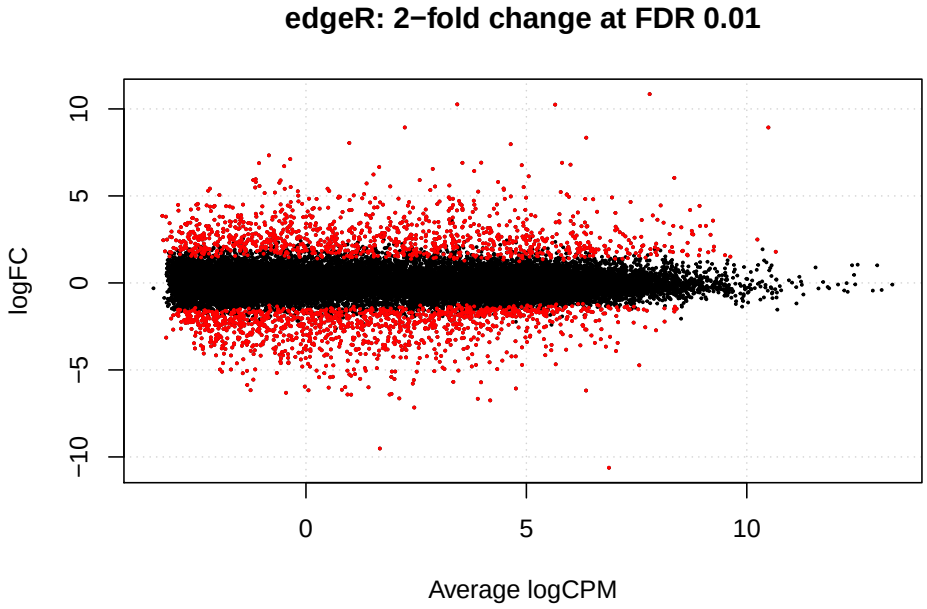
```
# Calculate significance for 2-fold change
edgeR_fc2 <- glmTreat(fit, coef = ncol(design), lfc = 1)

# Extract data for all genes
edgeR_fc2_all_genes <- topTags(edgeR_fc2, n = nrow(dgl), sort.by = "none")

# Check distribution of all p-values
hist(edgeR_fc2_all_genes$table$PValue)
```




```
# Select DEGs with FDR<0.01
edgeR_fc2_fdr_0.01 <- topTags(edgeR_fc2, n = nrow(dgl), p.value = 0.01)

# Count DEGs
dim(edgeR_fc2_fdr_0.01$table)
```

```
## [1] 2345    7
```

```
# MA plot
plotSmear(edgeR_fc2, de.tags = edgeR_fc2_fdr_0.01$table$gene_id,
          main="edgeR: 2-fold change at FDR 0.01")
```




```
# Clean-up
rm(edgeR_any_change, edgeR_any_change_all_genes, edgeR_any_change_fdr_0.1)
```

6.7.3 Check known genes

Like in **DESeq2** analysis, the list of DEGs suggested by **edgeR** includes selected known genes, which expression is associated with ER in breast cancer:

```
# List of selected previously known genes of interest
selected_known_genes=c("ESR1", "PGR", "TFF1", "TFF3", "GATA3", "FOXA1", "FOXC1", "MIA")

# Look at the genes of interest in the edgeR DEGs
edgeR_fc2_fdr_0.01$table %>%
  filter(gene_name %in% selected_known_genes)
```

```
##              gene_id gene_name     logFC unshrunk.logFC   logCPM       PValue          FDR
## 1  ENSG00000054598.6     FOXC1 -3.702670      -3.702971 4.900554 5.479376e-43 8.342693e-40
## 2 ENSG00000091831.20      ESR1  4.428545       4.428643 8.928694 4.484183e-23 8.152178e-21
## 3  ENSG00000261857.5       MIA -3.856874      -3.858896 2.278226 1.154317e-19 1.434710e-17
## 4 ENSG00000107485.14     GATA3  2.766902       2.766926 9.178419 1.539914e-17 1.506580e-15
## 5  ENSG00000129514.5     FOXA1  2.814738       2.814787 8.249821 8.375199e-13 4.445060e-11
## 6 ENSG00000082175.13       PGR  3.617757       3.617892 7.617047 2.641659e-10 9.206504e-09
## 7  ENSG00000160182.2      TFF1  4.454900       4.455097 8.046632 2.002155e-09 6.058944e-08
## 8 ENSG00000160180.15      TFF3  3.462370       3.462466 8.112775 5.373016e-08 1.289577e-06
```

```
# Clean-up
rm(selected_known_genes)
```

6.8 Save results

write.table function can be used to save DEGs (and results for all genes) to text files:

```
# Save DEGs
write.table(edgeR_fc2_fdr_0.01$table,
            file=file.path(base_folder,"analysis","results","edgeR_fc_2_fdr_0_01_DEGs.t
            quote=F, sep="\t", row.names = F)

# Save all genes
write.table(edgeR_fc2_all_genes$table,
            file=file.path(base_folder,"analysis","results","edgeR_fc_2_fdr_0_01_all_ge
            quote=F, sep="\t", row.names = F)
```

Chapter 7

Compare DEG lists

In this section we will compare DEGs, sugested by DESeq2 and edgeR for at least 2 fold changes at FDR 0.01.

7.1 Venn diagram

Venn diagram allows a visual assessment of the lists overlap.

```
# Library for plotting Venn diagram
# install.packages("VennDiagram")
suppressWarnings(suppressMessages(library(VennDiagram)))

# Prepare data
venn_data <- list(DESEq2=DESeq2_fc_2_fdr_0.01_DEGs.df$gene_id,
                  edgeR=edgeR_fc2_fdr_0.01$table$gene_id)
# Make plot
# (use a real file name to direct the output into a file)
venn.plot <- venn.diagram( venn_data, filename = NULL,
                           main="DESeq2 vs edgeR",
                           main.fontface="bold",
                           sub="FC>2, FDR<0.01",
                           col=c("red","blue"),
                           fill=c("red","blue"),
                           alpha=0.3)
grid.newpage()
grid.draw(venn.plot)
```


```
grid.newpage()

# Clean-up
rm(venn_data, venn.plot)
```

7.2 Compare FC and FDR estimates

The plots show a good agreement between fold-change estimates between **DESeq2** and **edgeR** packages. The FDR estimates also show reasonable correlation. However, it could be noted that, overall, **edgeR** FDR estimates are more conservative for some genes in our dataset.

```
# Estimate the size of DESEq and edgeR data for all genes
dim(DESeq2_fc_2_fdr_0.01_all_genes.df)
```

```
## [1] 28362     8
```

```
dim(edgeR_fc2_all_genes$table)
```

```
## [1] 24361     7
```

```
# Merge DESEq and edgeR data for all genes
all_genes_intersect.df <- inner_join(DESeq2_fc_2_fdr_0.01_all_genes.df,
                   edgeR_fc2_all_genes$table,
```

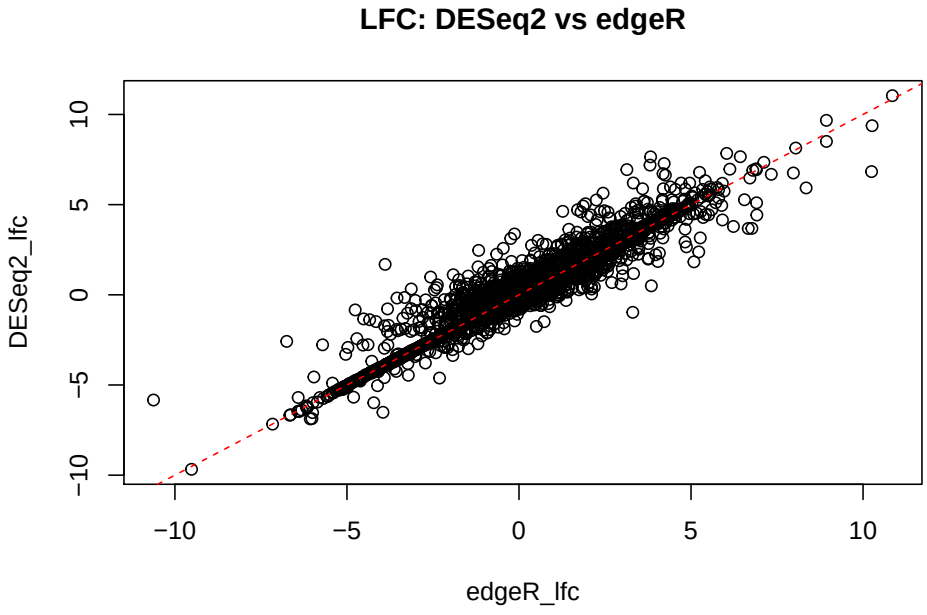
```
                    by="gene_id")

# Check result
dim(all_genes_intersect.df)
```

```
## [1] 24361    14
```

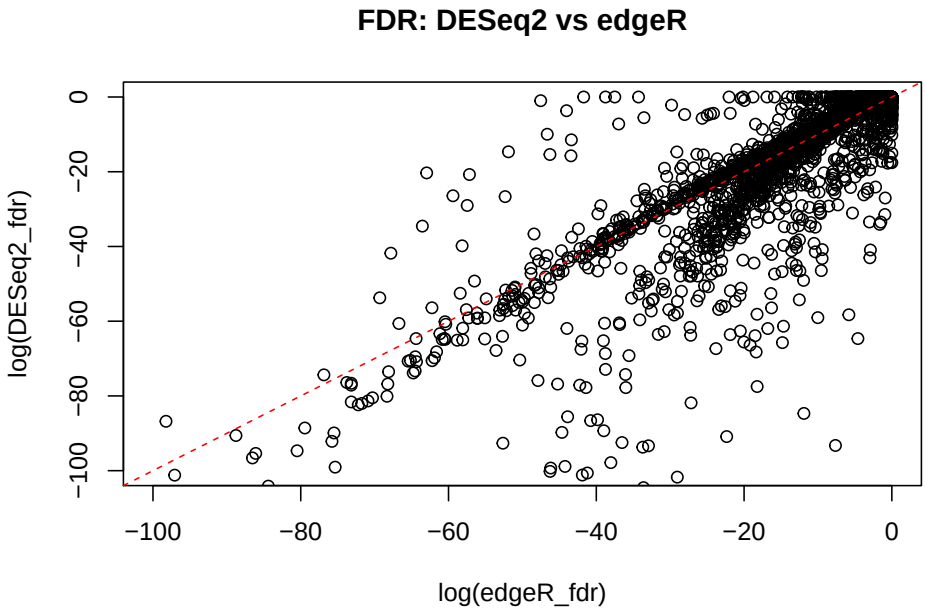
```
colnames(all_genes_intersect.df)
```

```
##  [1] "baseMean"       "log2FoldChange" "lfcSE"          "stat"           "pvalue"         "pad
```

```
# Select columns
all_genes_intersect.df <- all_genes_intersect.df %>%
  select(gene_name=gene_name.x,
         gene_id,
         DESeq2_lfc=log2FoldChange,
         DESeq2_fdr=padj,
         edgeR_lfc=logFC,
         edgeR_fdr="FDR")

# Compare fold change
plot(DESeq2_lfc~edgeR_lfc,
     data=all_genes_intersect.df,
     main="LFC: DESeq2 vs edgeR")
abline(0,1, lty=2, col="red")
```



```
# Compare adjusted p (FDR)
plot(log(DESeq2_fdr)~log(edgeR_fdr),
     data=all_genes_intersect.df,
     xlim=c(-100,0),ylim=c(-100,0),
     main="FDR: DESeq2 vs edgeR")
abline(0,1, lty=2, col="red")
```




```
# Save the intersect into a text file
write.table(all_genes_intersect.df,
            file=file.path(base_folder,"analysis","results","all_genes_intersect.txt")
            quote=F, sep="\t", row.names = F)
```

7.3 DEGs intersect list

It is reasonble to assume that the intersect between DEseq2 and edgeR (when using the same thresholds!) shows the most robust DEGs between ER-positive and Triple-negative breast cancer in the studied dataset.

Importantly, an external dataset is needed for an independent confirmation of the detected DEGs.

```
# Merge DESEq and edgeR data for DEGs
DEGs_intersect.df <- inner_join(DESeq2_fc_2_fdr_0.01_DEGs.df,
                   edgeR_fc2_fdr_0.01$table,
                   by="gene_id")
```

```
# Check result
dim(DEGs_intersect.df)
```

```
## [1] 1886   11
```

```
colnames(DEGs_intersect.df)
```

```
##  [1] "gene_name.x"    "gene_id"        "baseMean"       "log2FoldChange" "padj"           "ger
```

```
# Select columns
DEGs_intersect.df <- DEGs_intersect.df %>%
  select(gene_name=gene_name.x,
         gene_id,
         DESeq2_lfc=log2FoldChange,
         DESeq2_fdr=padj,
         edgeR_lfc=logFC,
         edgeR_fdr="FDR")

# Save the DEGs intersect into a text file
write.table(DEGs_intersect.df,
            file=file.path(base_folder,"analysis","results","DEGs_intersect.txt"),
            quote=F, sep="\t", row.names = F)
```

Chapter 8

Final section

Record time and objects in the environment at the end of analysis:

```
Sys.time()
```

```
## [1] "2020-04-08 16:34:22 BST"
```

```
ls()
```

```
##  [1] "all_genes_intersect.df"            "base_folder"                       "colorize"
```

```
sessionInfo()
```

```
## R version 3.6.2 (2019-12-12)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Catalina 10.15.4
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8
##
## attached base packages:
##  [1] grid      parallel  stats4    stats     graphics  grDevices utils     datasets  methods
##
## other attached packages:
##  [1] VennDiagram_1.6.20          futile.logger_1.4.3         edgeR_3.28.1                limma
##
## loaded via a namespace (and not attached):
##  [1] bitops_1.0-6             bit64_0.9-7              RColorBrewer_1.1-2       tools_3.6.2
## [40] RCurl_1.98-1.1           magrittr_1.5             GenomeInfoDbData_1.2.2   Formula_1.2-3
```

```
## [79] AnnotationDbi_1.48.0     memoise_1.1.0            cluster_2.1.0            elli
```