Learning Objectives

- Understanding the different components of differential expression analysis in the context of DESeq2
- Exploring different objects in DESeq2
- Building results tables for comprison of different sample classes
- MA plot
- Volcano plot

DESeq2: Differential expression analysis

Getting setup

Let's get started by opening RStudio and opening up the project that we created last lesson.

1. Go to the File menu and select 'Open project ...'
2. Navigate to `~/Desktop/DEanalysis/` and double click on the `DEanalysis.Rproj` file

You should see your environment become populated with all of the variables created last lesson. The only thing that we will need to do is reload the required libraries:

```
library(ggplot2)
library(RColorBrewer)
library(DESeq2)
library(pheatmap)
```

Normalization

To normalize the count data DESeq2 calculates size factors for each sample, using the *median of ratios method*. Let's take a quick look at size factor values we have for each sample:

```
> sizeFactors(dds)
Mov10_kd_2 Mov10_kd_3 Mov10_oe_1 Mov10_oe_2 Mov10_oe_3 Irrel_kd_1 Irrel_kd_2 Irrel_kd_3
 1.5646728  0.9351760  1.2016082  1.1205912  0.6534987  1.1224020  0.9625632  0.7477715
```

These numbers should be identical to those we generated initially when we had run the function `estimateSizeFactors(dds)`. Take a look at the total number of reads for each sample using `colSums(counts(dds))`. *How do the numbers correlate with the size factor?*

Now take a look at the total depth after normalization using `colSums(counts(dds, normalized=T))`, how do the values across samples compare with the total counts taken for each sample?

> *NOTE:* it can be advantageous to calculate gene-specific normalization factors (size factors) to account for further sources of technical biases such as differing dependence on GC content, gene length or the like, and these can be supplied to DESeq2 instead of using the median of ratios method.

> *NOTE:* A very popular normalization metric that is cited in the literature and used quite frequently is **RPKM/FPKM** (Reads Per Kilobase Million). Recent studies have shown that these metrics **should not be used**. This video by StatQuest is a great resource for understanding why not. Briefly, RNA-seq differential expression is about comparing proportions of expression. Because of the order of operations when computing RPKM/FPKM (scale by sequencing depth, followed by gene length), each sample becomes a pie of different size. Therefore, when we try to compare proportions from those pies to one another it is no longer a fair comparison (i.e 1/3 of a 6" pie is much smaller than 1/3 of a 10" pie).

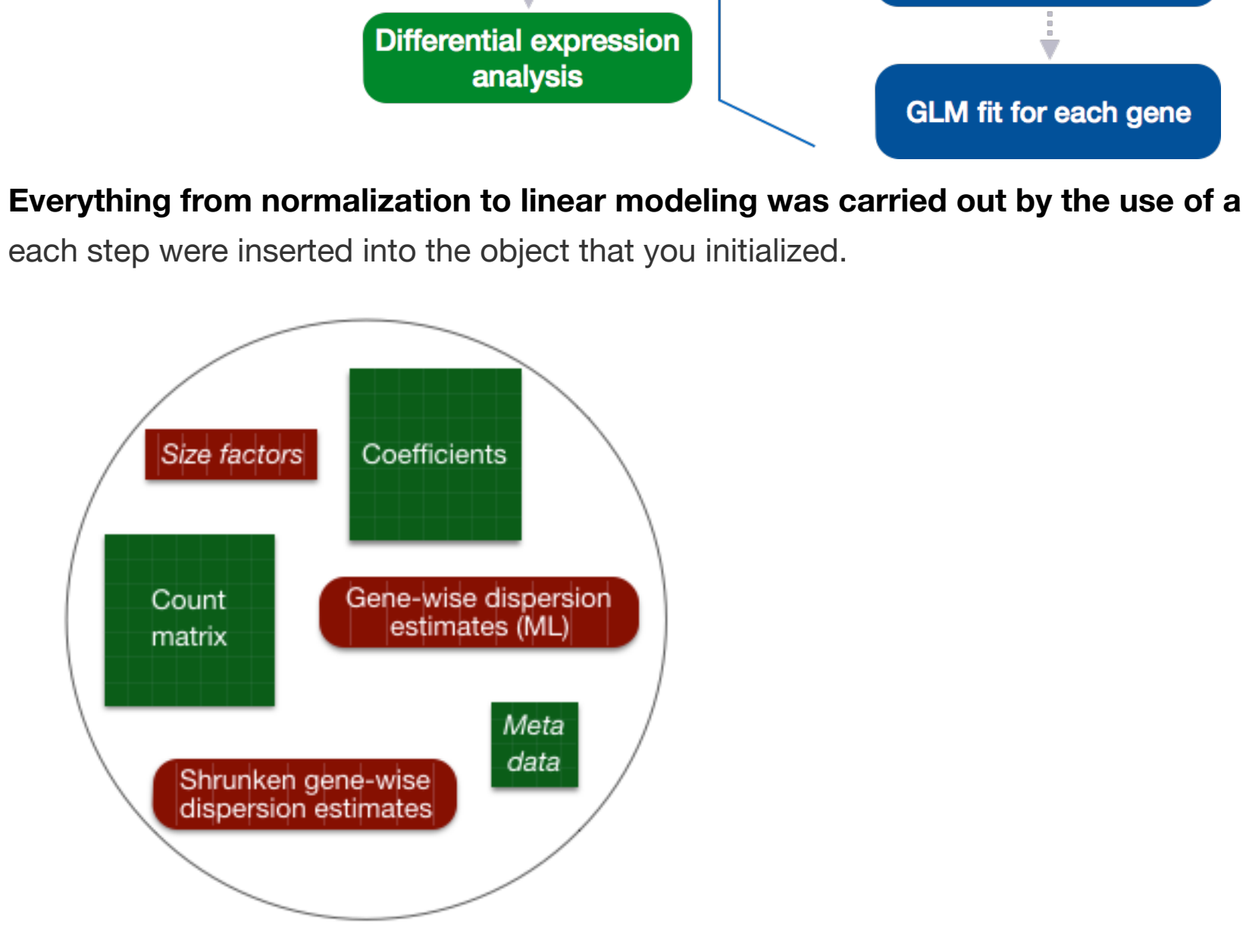
Running DESeq2

To run the differential expression pipeline on the raw counts in DESeq2, we use a **single call to the function `DESeq()`**. The required input is the `DESeqDataSet` object that we created in the last lesson. By re-assigning the results of the function back to the same variable name, we can continue to fill in the `slots` of our `DESeqDataSet` object.

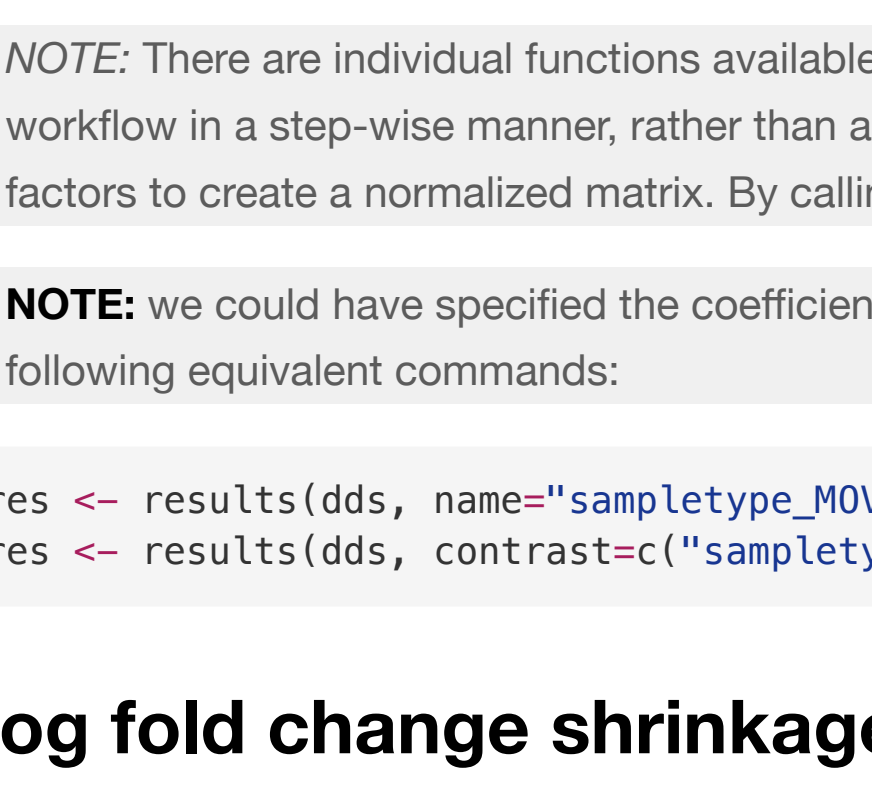
```
##Run analysis
dds <- DESeq(dds)
```

This function will print out a message for the various steps it performs:

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



Everything from normalization to linear modeling was carried out by the use of a single function! The results of each step were inserted into the object that you initialized.



> *NOTE:* There are individual functions available in DESeq2 that would allow us to carry out each step in the workflow in a step-wise manner, rather than a single call. We demonstrated one example when generating size factors to create a normalized matrix. By calling `DESeq()`, the individual functions for each step are run for you.

> **NOTE:** we could have specified the coefficient or contrast we want to build a results table for, using either of the following equivalent commands:

```
res <- results(dds, name="sampletype_MOV10_overexpression_vs_control")
res <- results(dds, contrast=c("sampletype","MOV10_overexpression","control"))
```

Log fold change shrinkage for visualization and ranking

Shrinkage of effect size (LFC estimates) is useful for visualization and ranking of genes. To shrink the LFC, we pass the dds object to the function `lfcShrink`. Below we specify to use the apeglm method for effect size shrinkage (Zhu, Ibrahim, and Love 2018), which improves on the previous estimator.

We provide the dds object and the name or number of the coefficient we want to shrink, where the number refers to the order of the coefficient as it appears in `resultsNames(dds)`.

Before using the function `lfcShrink`, install the package `apeglm` using `biocManager`.

```
BiocManager::install("apeglm")
```

```
resLFC <- lfcShrink(dds, coef="sampletype_MOV10_overexpression_vs_control", type="apeglm")
```

p-values and adjusted p-values

We can order our results table by the smallest p value:

```
resOrdered <- res[order(res$pvalue),]
```

We can summarize some basic tallies using the summary function.

```
summary(res)
```

```
out of 19748 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)       : 3582, 18%
LFC < 0 (down)     : 3847, 19%
outliers [1]       : 0, 0%
low counts [2]     : 3413, 17%
(mean count < 3)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

Excercise 1

How many adjusted p-values were less than 0.1?

sol: 7429

The `results` function contains a number of arguments to customize the results table which is generated. You can read about these arguments by looking up `?results`. Note that the `results` function automatically performs independent filtering based on the mean of normalized counts for each gene, optimizing the number of genes which will have an adjusted p value below a given FDR cutoff, `alpha`. Independent filtering is further discussed below. By default the argument `alpha` is set to `0.1`. If the adjusted p value cutoff will be a value other than `0.1`, `alpha` should be set to that value:

```
res05 <- results(dds, alpha=0.05)
summary(res05)
```

```
out of 19748 with nonzero total read count
adjusted p-value < 0.05
LFC > 0 (up)       : 3103, 16%
LFC < 0 (down)     : 3408, 17%
outliers [1]       : 0, 0%
low counts [2]     : 4171, 21%
(mean count < 5)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

Exercise 2

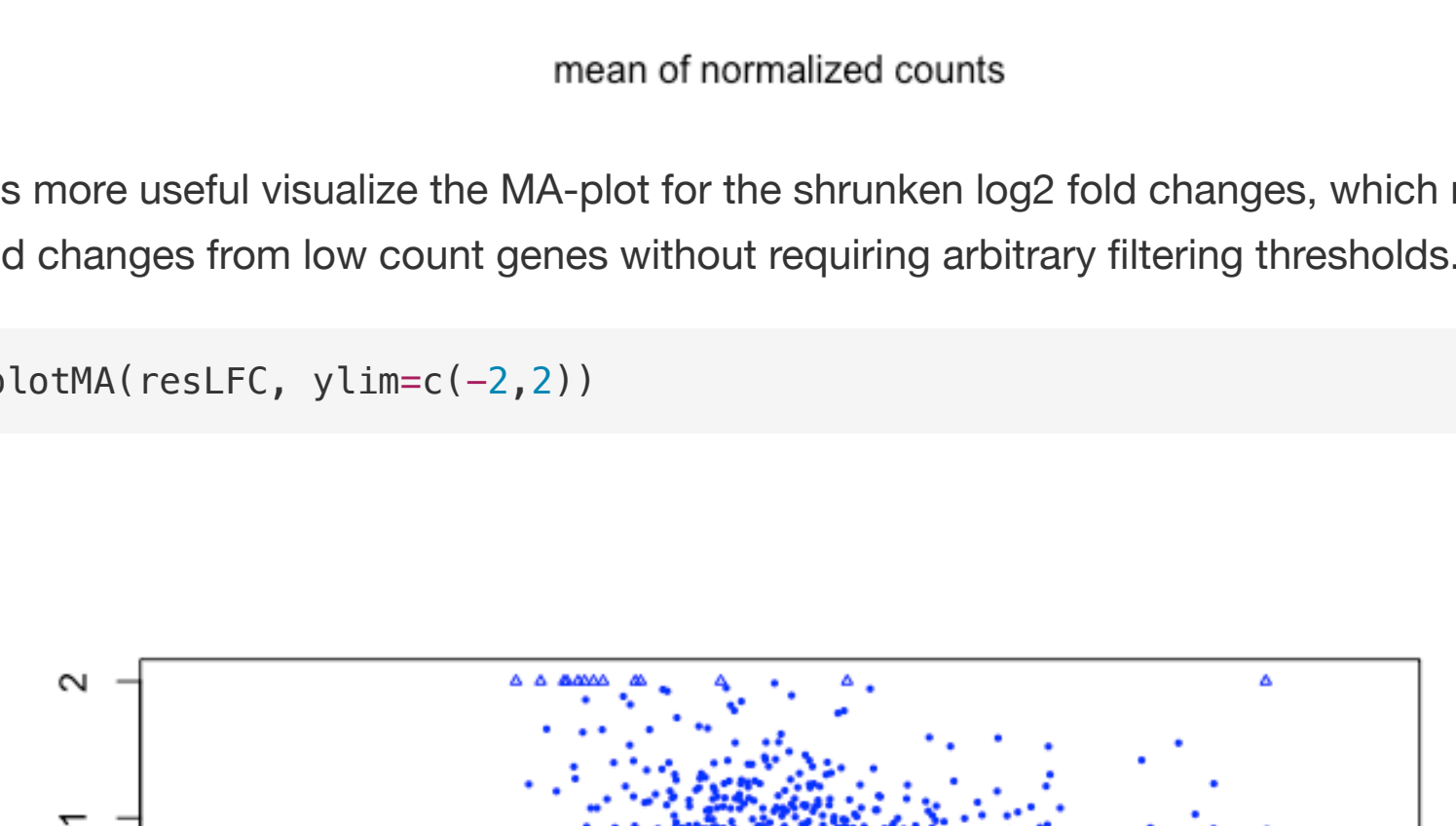
Explore the results table summary for the **Mov10_knockdown comparison to control**. How many genes are differentially expressed at FDR < 0.1? How many fewer genes do we find at FDR < 0.05?

Exploring and exporting results

MA-plot

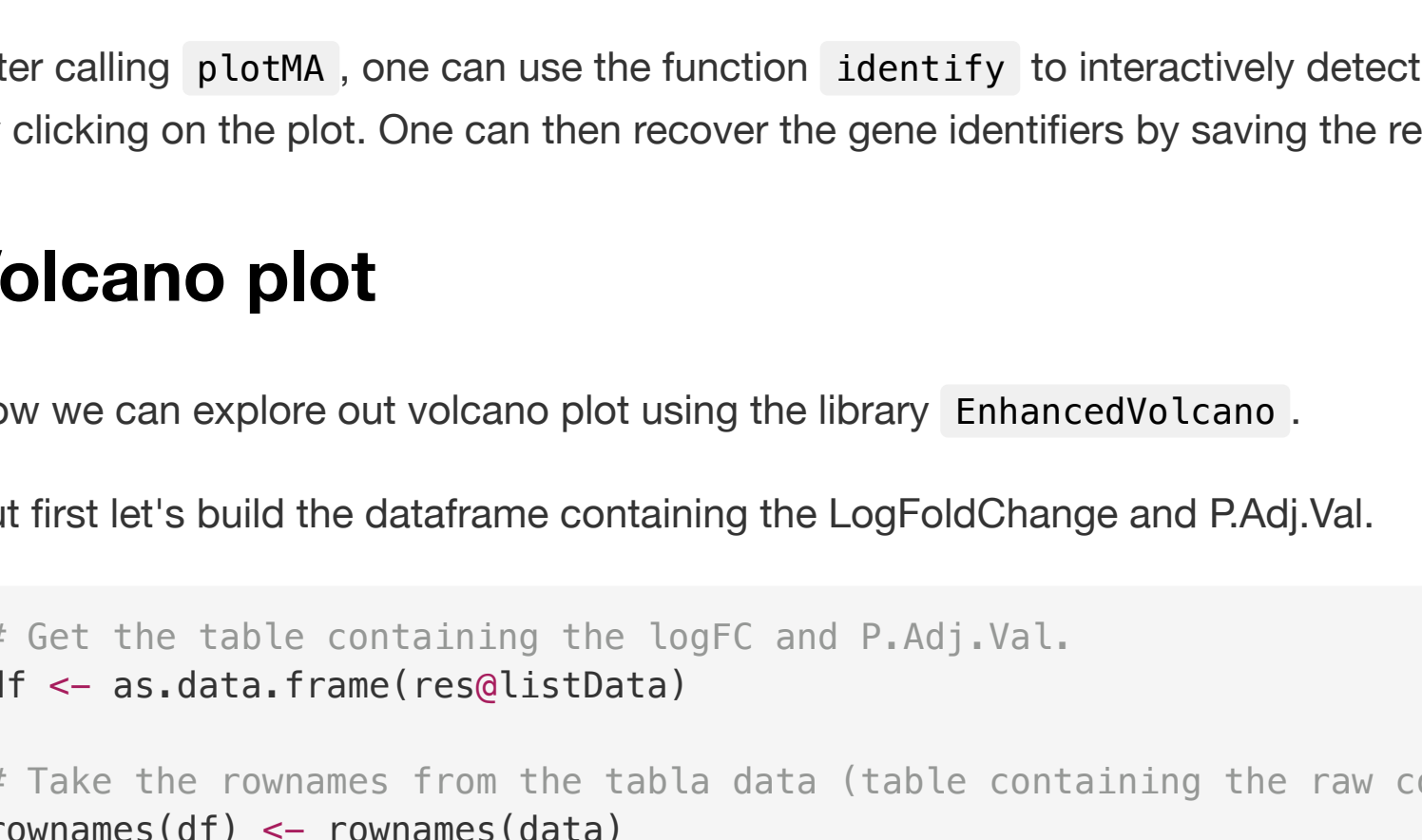
In DESeq2, the function `plotMA` shows the log2 fold changes attributable to a given variable over the mean of normalized counts for all the samples in the `DESeqDataSet`. Points will be colored blue if the adjusted p value is less than `0.1`. Points which fall out of the window are plotted as open triangles pointing either up or down.

First of all please keep in mind that the MA (ratio intensity) plot is meant to compare two or two group of samples. It concludes how different your samples are in terms of signal intensities (in microarray) or read counts (in RNAseq experiments). If you try to plot sample A versus sample A (same sample) you will notice the data points converge to zero at Y-axis because log (A/A) is zero.



It is more useful visualize the MA-plot for the shrunken log2 fold changes, which remove the noise associated with log2 fold changes from low count genes without requiring arbitrary filtering thresholds.

```
plotMA(resLFC, ylim=c(-2,2))
```



After calling `plotMA`, one can use the function `identify` to interactively detect the row number of individual genes by clicking on the plot. One can then recover the gene identifiers by saving the resulting indices:

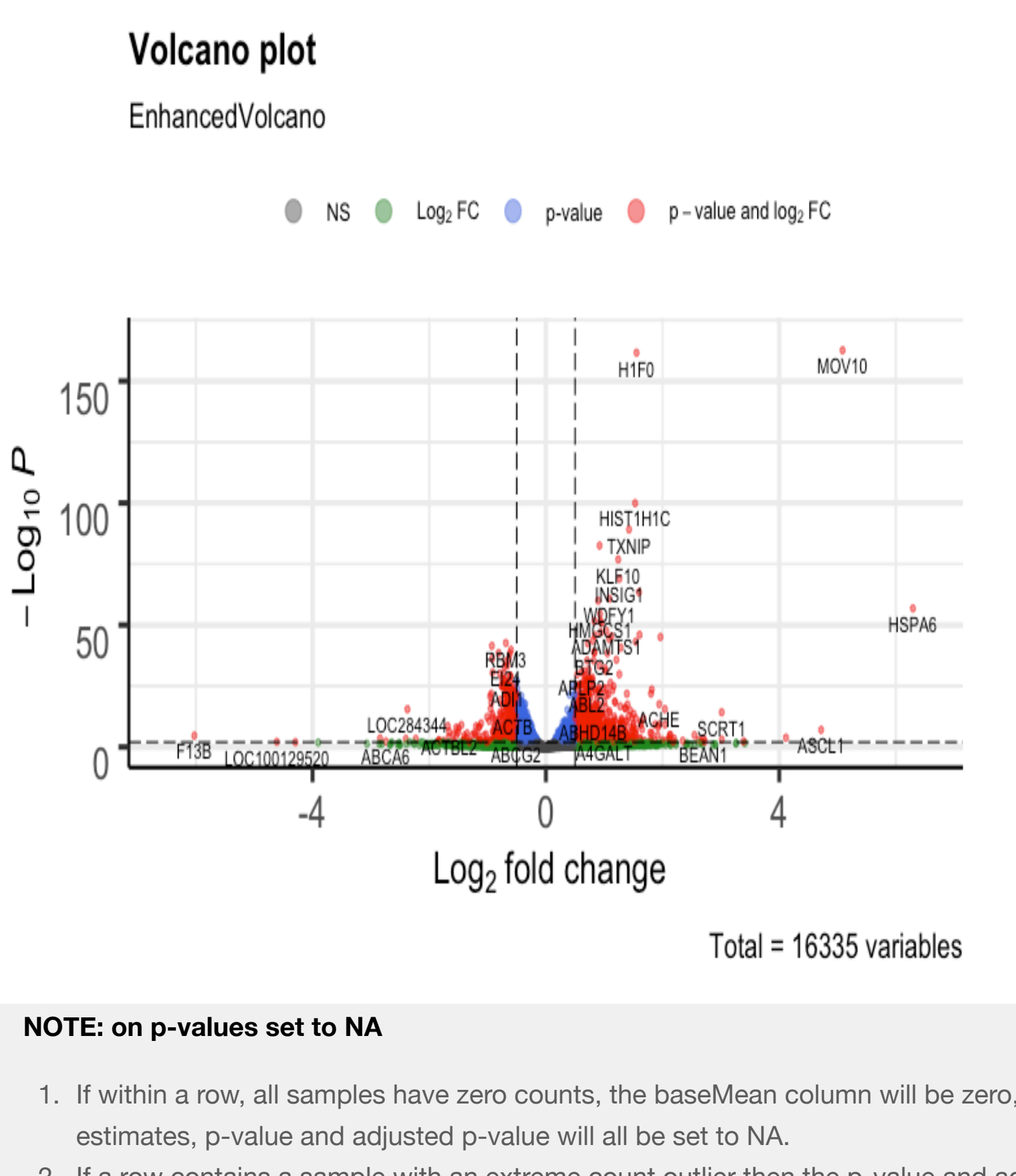
Volcano plot

Now we can explore out volcano plot using the library `EnhancedVolcano`.

But first let's build the dataframe containing the LogFoldChange and P.Adj.Val.

```
# Get the table containing the logFC and P.Adj.Val.
df <- as.data.frame(res@listData)

# Take the rownames from the tabla data (table containing the raw counts)
rownames(df) <- rownames(data)

# remove all the line with na
df <- df[complete.cases(df),]
```

```
library(EnhancedVolcano)


# Make your plot an modify cit-offs for log2FC and P value
# specify title, adjust points and label size
volano_plot <- EnhancedVolcano(df,
                               lab = rownames(df),
                               pCutoff = 0.01,
                               FCcutoff = 0.5,
                               x = "log2FoldChange",
                               y = "padj",
                               pointSize = 1,
                               legendLabSize = 10,
                               xlim = c(-6.5, 6.5 ),

                               labSize = 3.0)
volano_plot
```



> **NOTE: on p-values set to NA**
>
> 1. If within a row, all samples have zero counts, the baseMean column will be zero, and the log2 fold change estimates, p-value and adjusted p-value will all be set to NA.
> 2. If a row contains a sample with an extreme count outlier then the p-value and adjusted p-value will be set to NA. These outlier counts are detected by Cook's distance.
> 3. If a row is filtered by automatic independent filtering, for having a low mean normalized count, then only the adjusted p-value will be set to NA.

Exercise 3

Make a volcano plot of the results for the **Mov10_knockdown comparison to control** and write down a small description of your plot (max. 5 lines).
