# Performing differential gene expression analysis

Read counts to DGE, Part III

*Friederike Dündar, ABC, WCM*

*February 25, 2020*

## Contents



## Preparing for DE

```
library(DESeq2)
library(magrittr)
# should have the DESeq object, DESeq.ds
load("RNAseqGierlinski.RData")
```

We need to ensure that the fold change will be calculated using the WT as the base line. `DESeq` used the levels of the condition to determine the order of the comparison.

```
DESeq.ds$condition
```

```
##  [1] SNF2 SNF2 SNF2 SNF2 SNF2 WT   WT   WT   WT   WT
## Levels: SNF2 WT
```

```
DESeq.ds$condition <- relevel(DESeq.ds$condition, ref="WT")
DESeq.ds$condition
```

```
##  [1] SNF2 SNF2 SNF2 SNF2 SNF2 WT   WT   WT   WT   WT
## Levels: WT SNF2
```

Analysis design – check that the contrast is set up correctly (we want to compare the samples based on their condition):

```
design(DESeq.ds)
```

```
## ~condition
```

## Running the DE analysis

```
DESeq.ds <- DESeq(DESeq.ds)
```

This one line of code is equivalent to these three lines of code:

```
# sequencing depth normalization between the samples
DESeq.ds <- estimateSizeFactors(DESeq.ds)
# gene-wise dispersion estimates across all samples
DESeq.ds <- estimateDispersions(DESeq.ds)
# fit a neg. binomial GLM and apply Wald statistics to each gene
DESeq.ds <- nbinomWaldTest(DESeq.ds)
```

You can see that the `DESeq` object now has additional entrees in the `rowData` slot:

```
DESeq.ds
```

```
## class: DESeqDataSet
## dim: 6394 10
## metadata(1): version
## assays(6): counts log.counts ... H cooks
## rownames(6394): YAL012W YAL068C ... YMR325W YMR326C
## rowData names(22): baseMean baseVar ... deviance maxCooks
## colnames(10): SNF2_1 SNF2_2 ... WT_4 WT_5
## colData names(2): condition sizeFactor
```

These are the per-gene estimates such as their average expression across all samples and the p-value results from the Wald test comparing the conditions as specified in Intercept, condition_SNF2_vs_WT:

## Adjusting for multiple hypothesis testing with independent filtering

Since we perform thousands of tests, the number of times that the p-value represents a random event rather than a systematic one become a cause for concern (e.g., if we allow for 5% error, we can expect 350 false positives if we test 7,000 times). It is therefore important that we adjust the p-values in the light of our numerous hypotheses; this is typically done either via the false discovery rate as described by Benjamini and Hochberg (see `?p.adjust()`). In general, the more tests we perform, the more strongly the individual p-values will be "punished".

For RNA-seq, the argument has been made that genes with very low read counts can be ignored for downtream analyses as their read counts are often too unreliable and variable to be accurately assessed with only 3-5 replicates (see, for example, [Love et al., 2014; Stephens 2017]. Consider the following example of a gene with very low read counts:

| Gene | WT_1 | WT_2 | WT_3 | KO_1 | KO_2 | KO_3 |
|---|---|---|---|---|---|---|
| GeneX | 0 | 1 | 3 | 0 | 1 | 0 |

If we assume that the sequencing depths are similar across the individual samples, these values are very close to one another and there is little certainty as to whether the expression is robustly decreased in the KO condition or not. These kinds of genes often have no chance of being identified as statistically significantly changed between the conditions; therefore it may make sense to remove them **before the multiple testing adjustments**. This means effectively that these genes are ignored for all further analyses – not because they aren't biologically interesting, but because we cannot make accurate enough assessments of their expression values.

Once you're on board with removing lowly expressed genes, however, you will quickly run into the question of "how low is too low?". The `results()` function of `DESeq2` will try to find the optimal expression cut-off to maximize the absolute number of genes that pass the adjusted p-value threshold. You can read more about the details in their vignette.

The `results()` functions allows you to extract the base means across samples, log2 fold changes, standard errors, and test statistics for every gene in one go. If you keep `independentFiltering = TRUE`, the search for the expression threshold will be performed by comparing the numbers of passing genes for an FDR cut-off (defined via the `alpha` parameter) over different expression level cut-offs.

```
DGE.results <- results(DESeq.ds, independentFiltering = TRUE, alpha = 0.05)

# the first line will tell you which comparison was done to achieve the log2FC
head(DGE.results)
```

```
## log2 fold change (MLE): condition SNF2 vs WT
## Wald test p-value: condition SNF2 vs WT
## DataFrame with 6 rows and 6 columns
##                    baseMean     log2FoldChange             lfcSE
##                   <numeric>          <numeric>         <numeric>
## YAL012W    5542.39136255275  0.316250301955328 0.162134956519878
## YAL068C   0.967854292906838  0.402411385706722  1.22661538009516
## YAL067C    40.8786754337591   1.08962978678884 0.229556215359416
## YAL066W   0.140275942926642  0.688449617485271  3.15993647220664
## YAL065C    5.16394170653055 -0.554858827975471 0.649272027599774
```

```
## YAL064W.B  8.35600217065942  -0.19250559525655 0.439820610433718
##                         stat               pvalue                padj
##                    <numeric>            <numeric>           <numeric>
## YAL012W     1.95053743340385   0.0511120962285505   0.100943105212298
## YAL068C    0.328066476449613    0.742861400414189                  NA
## YAL067C     4.74667952284717 2.06783194400164e-06 1.26456667938648e-05
## YAL066W    0.217868182965246     0.82753181797371                  NA
## YAL065C   -0.854586066223538    0.392780344104072    0.50704221694258
## YAL064W.B -0.437691164738086    0.661610175087731    0.747885991555858
```

```
summary(DGE.results)
```

```
##
## out of 6394 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 1289, 20%
## LFC < 0 (down)     : 1455, 23%
## outliers [1]       : 0, 0%
## low counts [2]     : 248, 3.9%
## (mean count < 3)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
# the DESeqResult object can basically be handled like a data.frame
table(DGE.results$padj < 0.05)
```

```
##
## FALSE  TRUE
##  3402  2744
```

NAs in the `padj` column (but values in both `log2FC` and `pvalue`) are indicative of that gene being filtered out by the independent filtering (because it didn't make the expression cut-off).

# Assessing the DE results

Once you've extracted the test results, you need to ensure that the results make sense, i.e. that the results meet reasonable expectations in terms of the number of genes passing the adj. p-value threshold.

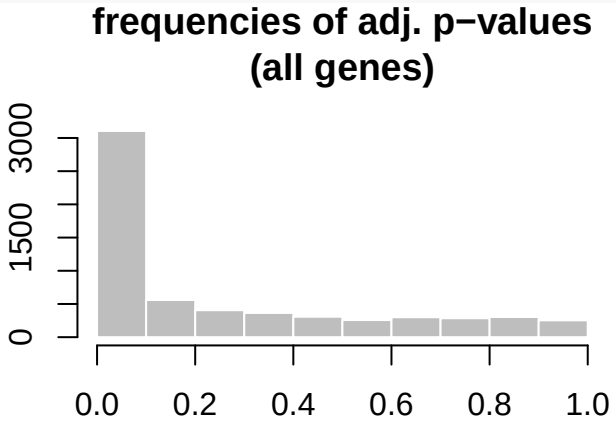
Typical assessments following the performance of the statistical tests include:

- **histogram** of p-values (What is your expectation?)
- individual **plots of selected genes** of interest for which the direction and magnitude of change are known (e.g. a gene that was knocked-out should show significantly reduced expression levels)
- **heatmaps** of lists of genes of interest, e.g. the most strongly changing ones
- **MA-plots** and **volcano plots** visualizing the magnitude of the change in comparison to the magnitude of baseline expression

## Visual assessments using normalized read counts

Histogram of adjusted p-values:

```
hist(DGE.results$padj,
  col="grey", border="white", xlab="", ylab="",
  main="frequencies of adj. p-values\n(all genes)",
  cex = 0.4)
```



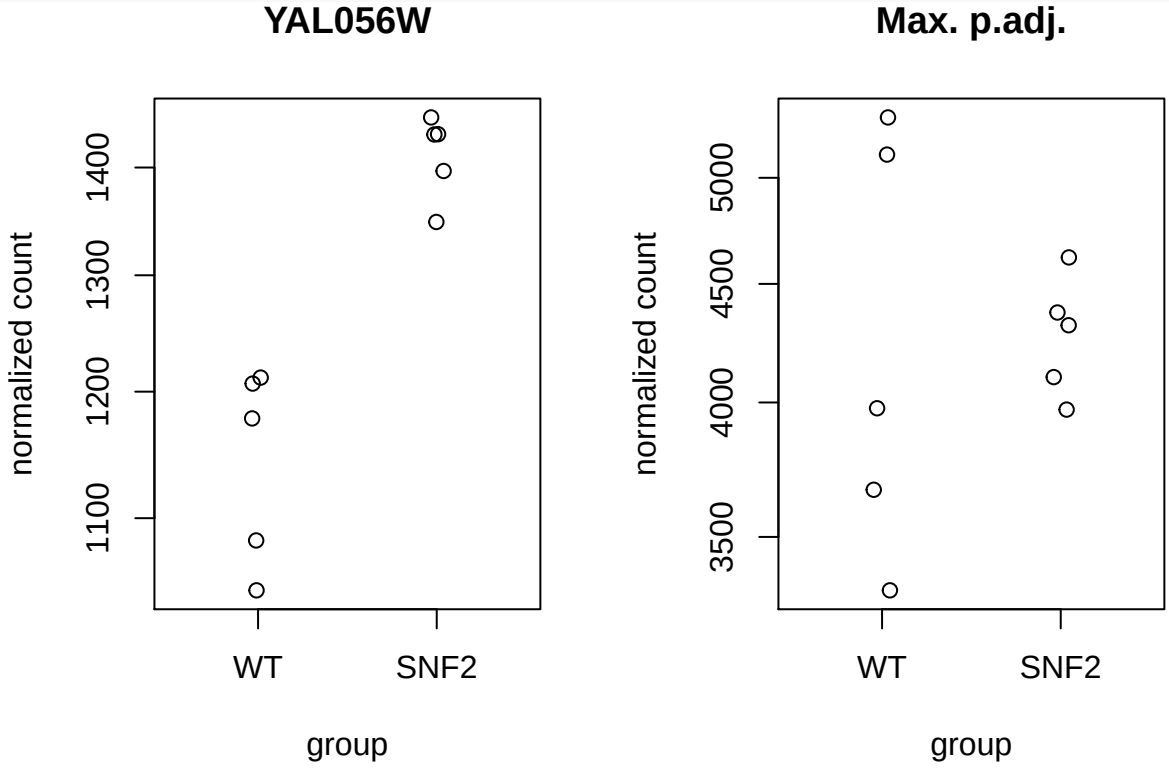

A sorted table of the results, i.e. sorted by adjusted p-values:

```
DGE.results.sorted <- DGE.results[order(DGE.results$padj),]
head(DGE.results.sorted)
```

```
## log2 fold change (MLE): condition SNF2 vs WT
## Wald test p-value: condition SNF2 vs WT
## DataFrame with 6 rows and 6 columns
##                 baseMean    log2FoldChange             lfcSE
##                <numeric>         <numeric>         <numeric>
## YGR234W 2866.10309210011 -4.24175662798006  0.107214528984456
## YDR033W 4101.97991571202 -3.69963733450346 0.0992421390851315
## YOR290C 778.501167786521 -6.92226218358102  0.186427956792307
## YML123C 4891.20882963984 -4.68459908667142   0.14896872257829
## YIL121W 880.417359160784 -2.76320140431385 0.0882348701220066
## YHR215W 291.347358509874 -4.52551156087238  0.153553445435413
##                     stat               pvalue                 padj
##                <numeric>            <numeric>            <numeric>
## YGR234W -39.5632631897776                    0                    0
## YDR033W -37.2788955237034 3.60710510082151e-304 1.10846339748245e-300
## YOR290C -37.1310306816958 8.87297339244115e-302 1.81777648233144e-298
## YML123C -31.4468635133086 4.63322648478396e-217 7.11895249387056e-214
## YIL121W  -31.316433066576 2.78808110363601e-215 3.42710929258938e-212
## YHR215W -29.4718985174179 6.59937593690135e-191 6.75996075136595e-188
```

Many plots, even after the DE tests, use the **normalized** and variance-adjusted **read counts** for every gene in every single sample (see the previous `Rmd` document). To inspect the expression values underlying the outcome of the statistical test for a given gene, the (seq. depth normalized, log2-transformed) counts for single genes can be plotted with a `DESeq2` wrapper function, `plotCounts()`:

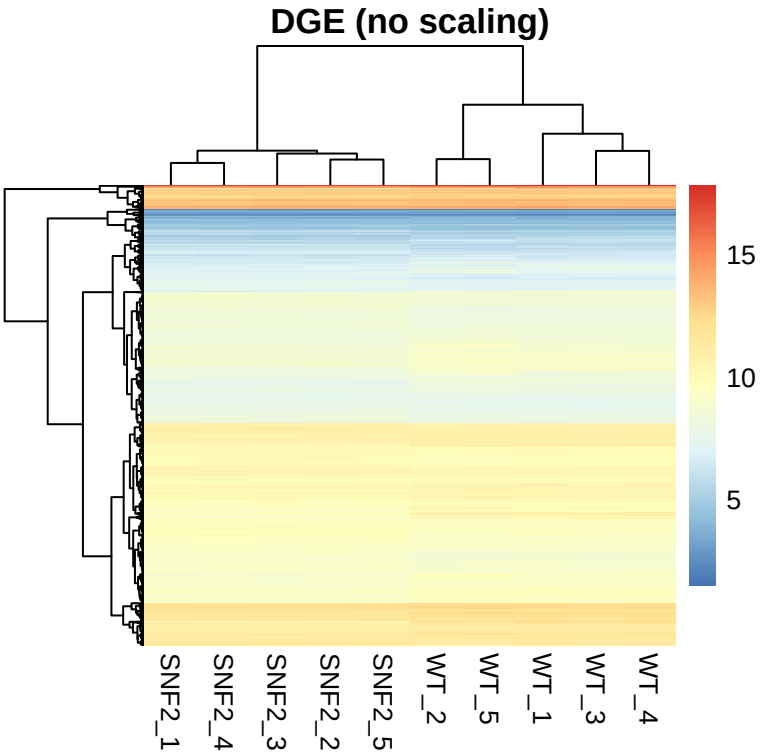
```
par(mfrow=c(1,2))
plotCounts(DESeq.ds, gene="YAL056W", normalized = TRUE)
plotCounts(DESeq.ds, gene = which.max(DGE.results$padj),
  main = "Max. p.adj.")
```



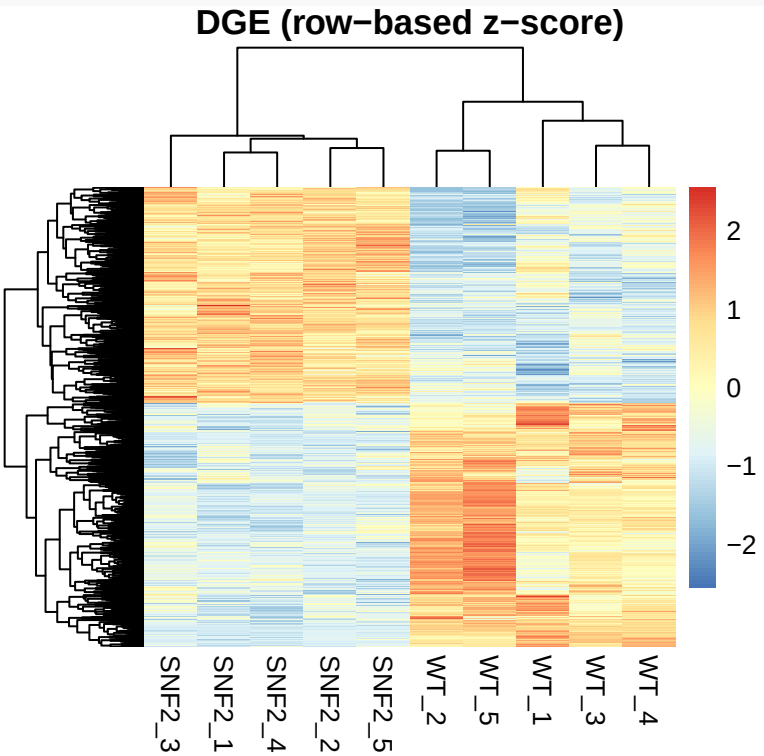

`plotCounts` simply uses `counts(dds, normalized = TRUE) + 0.5` to extract the expression values. You can also use `pcaExplorer` for individual gene plots of `rlog` values. Or write your own function.

To expand the same type of information for numerous genes at once, **heatmaps** have proven useful. A heatmap of the genes that show differential expression with adjusted p-value <0.05 can be generated so:

```
# identify genes with the desired adjusted p-value cut-off
DGEgenes <- rownames(subset(DGE.results.sorted, padj < 0.05))

# extract rlog-transformed values into a matrix
rlog.dge <- DESeq.rlog[DGEgenes,] %>% assay

library(pheatmap)
# heatmap of DEG sorted by p.adjust
pheatmap(rlog.dge, scale="none",
  show_rownames = FALSE, main = "DGE (no scaling)")
```




```
pheatmap(rlog.dge, scale="row",
  show_rownames = FALSE, main = "DGE (row-based z-score)")
```

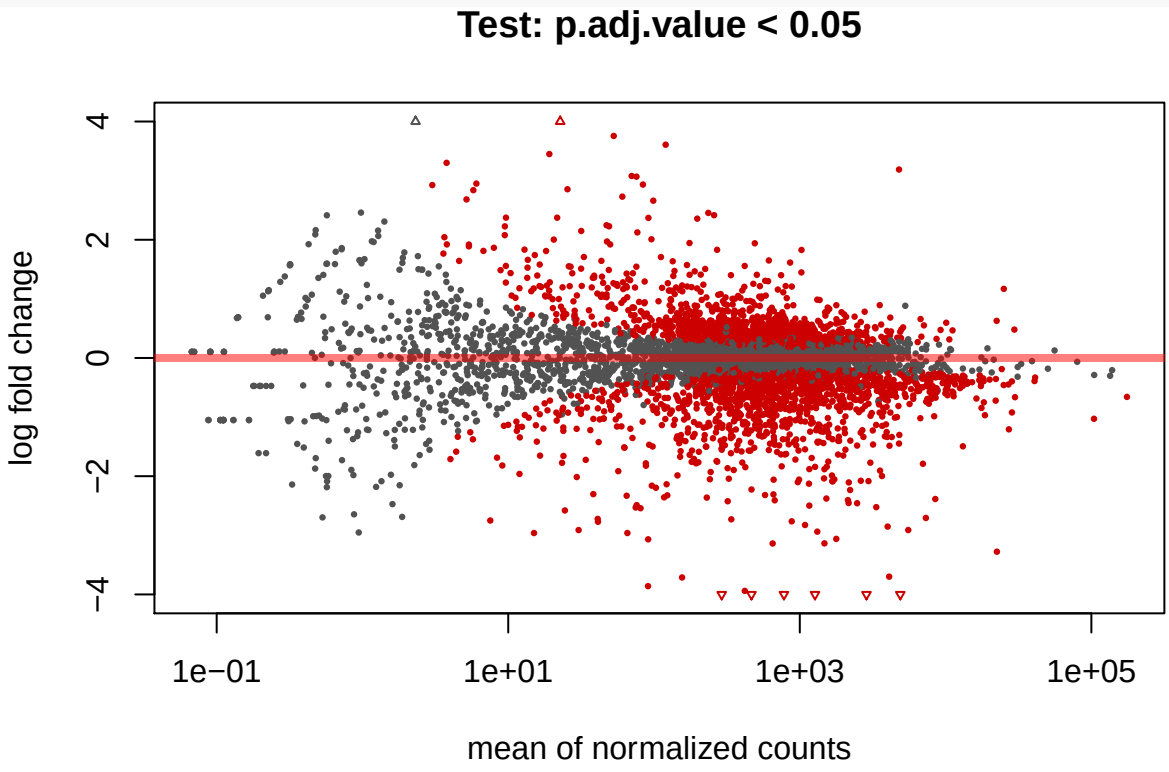


## Shrinking the logFC values

Despite the fact that many visualizations will use the normalized read counts per sample, e.g. by simply focusing on that subset of genes that passed the adjusted p-value threshold, it is often useful to interrogate the **fold change** values that were estimated (i.e., the **magnitude**) of change as well. Many researchers believe that subtle fold changes are more difficult to interpret than dramatic differences in expression; therefore you may find yourself wanting to rank the genes by the logFC values in addition to their statistical significance as implied by their adjusted p-values.

The **MA-plot** provides a global view of the differential genes, with the log2 fold change on the y-axis over the mean of normalized counts.
Genes that pass the significance threshold (adjusted p.value <0.05) are colored in red.

```
plotMA(DGE.results, alpha = 0.05,
       main = "Test: p.adj.value < 0.05", ylim = c(-4,4))
```

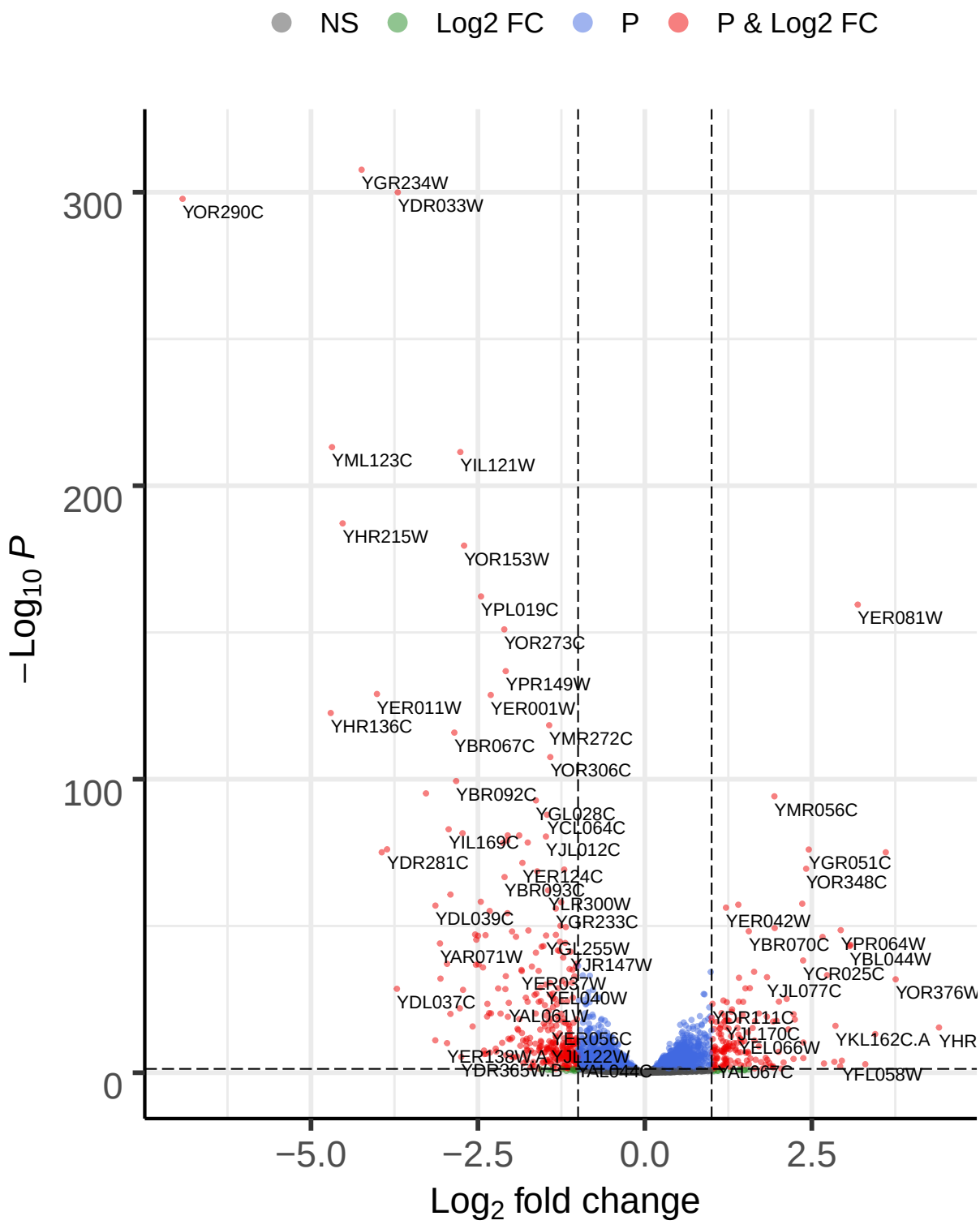



A very similar concept is shown with "volcano plots". The library `EnhancedVolcano` comes with numerous parameters to tweak these plots. Check out their documentation!

```
#BiocManager::install("EnhancedVolcano")
library(EnhancedVolcano)
vp1 <- EnhancedVolcano(DGE.results,
    lab = rownames(DGE.results),
    x = 'log2FoldChange',
    y = 'padj', pCutoff = 0.05,
  title = "SNF2 / WT")
print(vp1)
```

## SNF2 / WT

Bioconductor package EnhancedVolcano

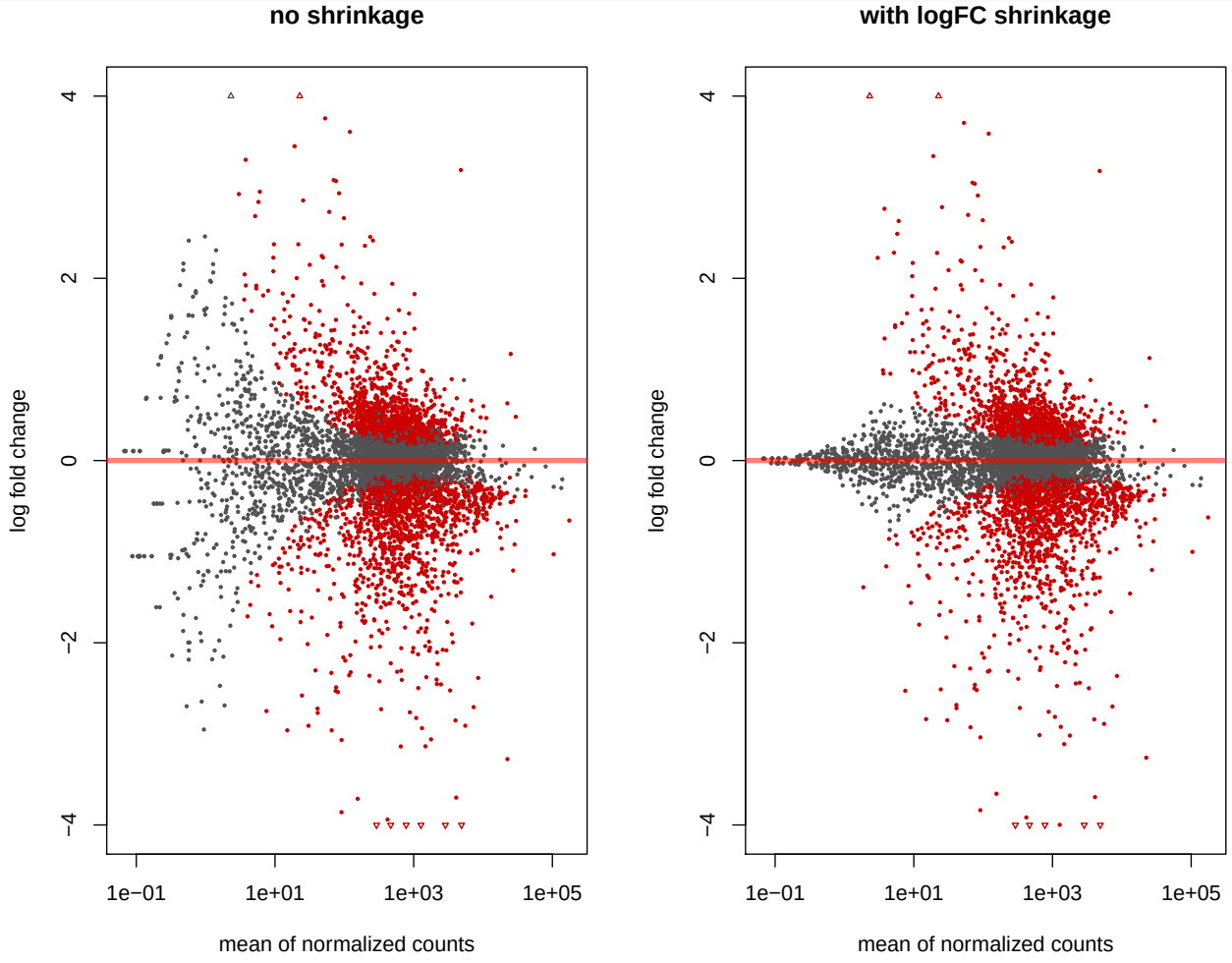



Total = 6394 variables

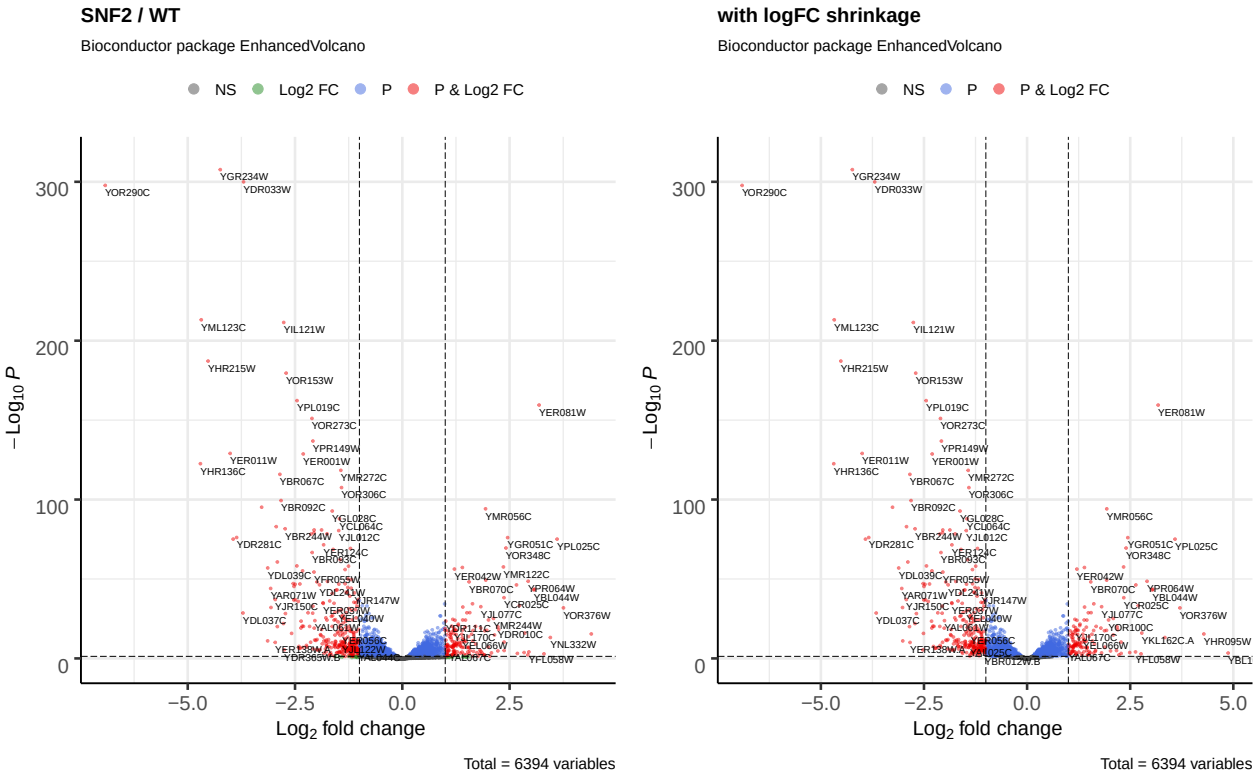
From these plots it is immediately clear that strongly expressed genes are over-represented in the population of statistically significantly changed genes.
Akin to the overdispersion problem that we observed with the raw read counts (remember heteroskedasticity vs. homoskedasticity and the reason for the `rlog` transformation), fold changes of genes with low and/or noisy expression values tend to be unreliable and often inflated. `DESeq2` provides a function that will **shrink the logFC estimates** for lowly and noisily expressed genes towards zero, therefore reducing their importance for any subsequent downstream analyses. This is particularly important for applications with ranked gene lists such as GSEA.

```
DGE.results.shrnk <- lfcShrink(DESeq.ds, coef = 2, type = "apeglm")
```

```
par(mfrow = c(1,2))
plotMA(DGE.results, alpha = 0.05,
       main = "no shrinkage", ylim = c(-4,4))

plotMA(DGE.results.shrnk,
  alpha = 0.05,
  main = "with logFC shrinkage", ylim = c(-4,4))
```




```
vp2 <- EnhancedVolcano(DGE.results.shrnk,
    lab = rownames(DGE.results.shrnk),
    x = 'log2FoldChange',
    y = 'padj', pCutoff = 0.05,
  title = "with logFC shrinkage")

library(patchwork)
vp1 + vp2
```




## Number 1 sanity check: is SNF2 affected in the SNF2 mutant yeast samples?

To find this out, we need to retrieve the gene names and match them to the ORF IDs that we've used so far. http://www.bioconductor.org/packages/3.1/data/annotation/ lists annotation packages that are available within R through the Bioconductor repository.
We will go with `org.Sc.sgd.db`.

```
#BiocManager::install("org.Sc.sgd.db")
library(org.Sc.sgd.db) # org.Hs.eg.db, org.Mm.eg.db

# list keytypes that are available to query the annotation data base
keytypes(org.Sc.sgd.db)
```

```
##  [1] "ALIAS"        "COMMON"       "DESCRIPTION"  "ENSEMBL"
##  [5] "ENSEMBLPROT"  "ENSEMBLTRANS" "ENTREZID"     "ENZYME"
##  [9] "EVIDENCE"     "EVIDENCEALL"  "GENENAME"     "GO"
## [13] "GOALL"        "INTERPRO"     "ONTOLOGY"     "ONTOLOGYALL"
## [17] "ORF"          "PATH"         "PFAM"         "PMID"
## [21] "REFSEQ"       "SGD"          "SMART"        "UNIPROT"
```

```
# list columns that can be retrieved from the annotation data base
columns(org.Sc.sgd.db)
```

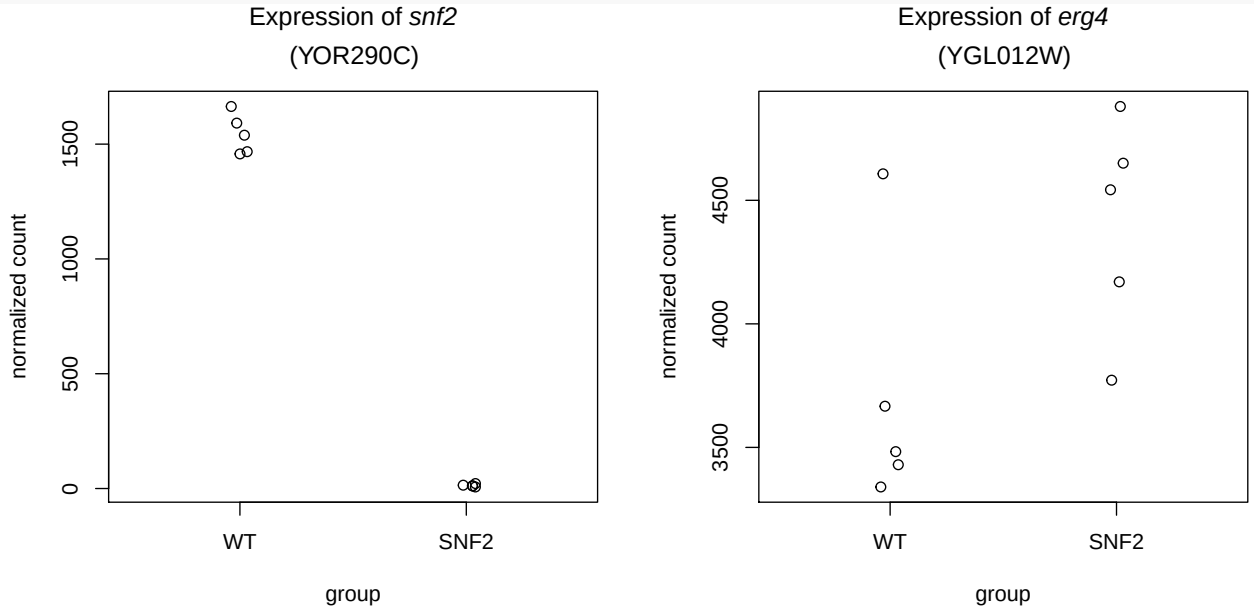
```
##  [1] "ALIAS"        "COMMON"       "DESCRIPTION"  "ENSEMBL"
##  [5] "ENSEMBLPROT"  "ENSEMBLTRANS" "ENTREZID"     "ENZYME"
##  [9] "EVIDENCE"     "EVIDENCEALL"  "GENENAME"     "GO"
## [13] "GOALL"        "INTERPRO"     "ONTOLOGY"     "ONTOLOGYALL"
## [17] "ORF"          "PATH"         "PFAM"         "PMID"
## [21] "REFSEQ"       "SGD"          "SMART"        "UNIPROT"
```

```
# make a batch retrieval for all DE genes
DGEgenes <- rownames(subset(DGE.results.sorted, padj < 0.05))

anno.DGE <- select(org.Sc.sgd.db,
  keys = DGEgenes, # original rownames of DGE results
  keytype="ORF", # our rownames are ORF identifiers
  columns=c("SGD","GENENAME")) # what to return
```

```
## check whether SNF2 pops up among the top downregulated genes
head(anno.DGE)
```

```
##       ORF        SGD GENENAME
## 1 YGR234W S000003466     YHB1
## 2 YDR033W S000002440     MRH1
## 3 YOR290C S000005816     SNF2
## 4 YML123C S000004592    PHO84
## 5 YIL121W S000001383     QDR2
## 6 YHR215W S000001258    PHO12
```

```
par(mfrow=c(1,2))
plotCounts(dds = DESeq.ds,
  gene = "YOR290C", # SNF2
  normalized = TRUE, transform = FALSE,
  main = expression(atop("Expression of "*italic("snf2"), "(YOR290C)")))

plotCounts(dds = DESeq.ds,
  gene = "YGL012W", # the last gene in DGE
  normalized = TRUE, transform = FALSE,
  main = expression(atop("Expression of "*italic("erg4"), "(YGL012W)")))
```




## Result export

Export the (shrunken) log2FC, p-values etc. into a text file:

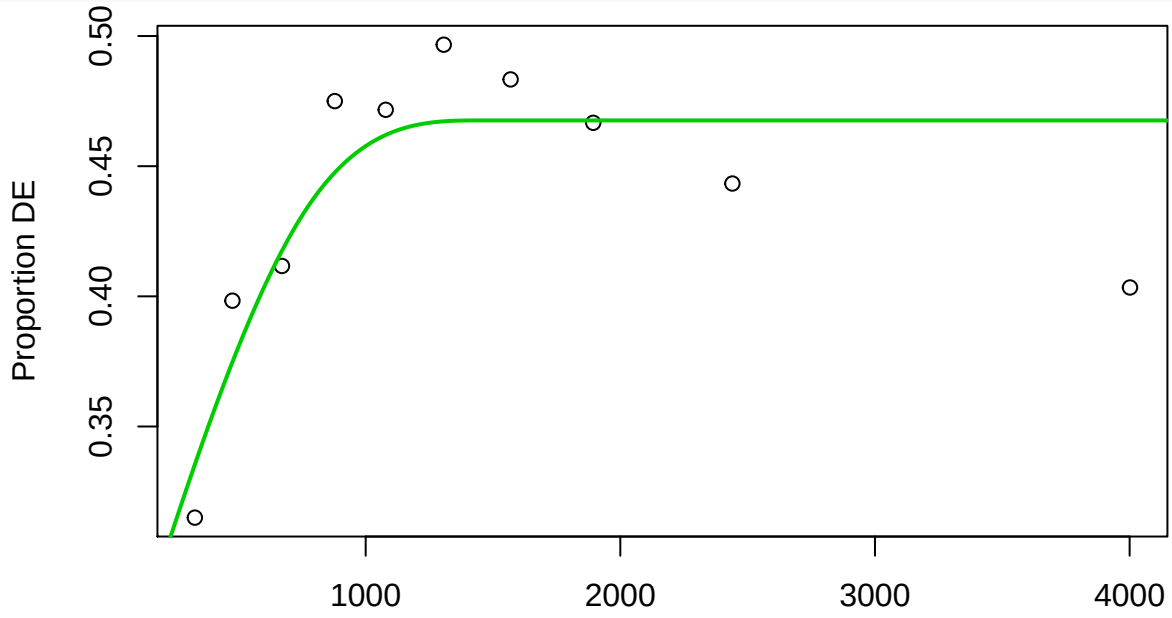
```
out.df <- merge(as.data.frame(DGE.results.shrnk),
  anno.DGE,
  by.x = "row.names",
  by.y = "ORF")

write.table(subset(out.df, padj < 0.05),
  file = "DESeq2results_WT-vs-SNF2.txt",
  sep = "\t", quote = FALSE, row.names = FALSE)
```

## Example downstream analysis: GO term enrichments

*Among our list of DE genes, which GO terms are enriched?*
Transcripts that are longer or more highly expressed give more statistical power for detecting differential expression between samples
The same bias holds true for GO categories: categories with predominantly highly expressed or long genes are more likely to be found to be over-represented within the DEG.
`GOseq`:

1. determine DEG
2. quantify likelihood of DE as a function of gene length (–> weight)
3. statistical test of each GO category's significance taking the DE probability into account

```
library(goseq) # package for GO term enrichment

# Constructing a named vector of 0 (= not DE) and 1 (= DEG)
gene.vector <- row.names(DGE.results.shrnk) %in% DGEgenes %>% as.integer
names(gene.vector) <- row.names(DGE.results.shrnk)

# Quantifying the length bias (= weight for each gene)
# using a Probability Weighting Function (PWF):
# probability of a gene being DE ~ length
# proportion of DE genes is plotted as a function of the transcript length
pwf <- nullp(gene.vector, "sacCer3", "ensGene")
```




Biased Data in 600 gene bins.

```
# do the actual test for enrichment of GO terms
GO.wall <- goseq(pwf, "sacCer3", "ensGene")
```

```
# retrieving the GO categories assigned to each gene
go_gns <- getgo( rownames(DGE.results.shrnk), 'sacCer3', 'ensGene') %>% stack
# in principle, the gene information could be added to the goseq results:
merge(GO.wall, go_gns, by.x = "category", by.y = "values") %>% dim
```

```
## [1] 438934      8
```

To summarize the results, you can use, for example, REVIGO.

```
# export a list of over-represented GO terms plus the corresponding p-value
# which is the input type that REVIGO needs
subset(GO.wall, over_represented_pvalue < 0.01,
       select = c("category","over_represented_pvalue")) %>%
  write.table(.,
```

```
    file = "~/Documents/Teaching/2019_RNA-seq/Enriched_GOterms_goseq.txt",
    quote = FALSE, row.names = FALSE, col.names = FALSE)
```

For more functions helping you with the visualization of GO term enrichments, see the GO term enrichment document `Rmd` file on github and the HBC Training Site.

## References


- Love, M.I., Huber, W., & Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology.* doi: 10.1186/s13059-014-0550-8
- Stephens M. (2017) .False discovery rates: A new deal. *Biostatistics.* doi: 10.1093/biostatistics/kxw041
- Love, M.I., Anders, S., Huber, W. (2019). Analyzing RNA-seq with DESeq2. *Bioconductor Vignette.* http://www.bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html