Performing differential gene expression (DGE) analysis

Friederike Dündar, ABC

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.

```
str(DESeq.ds$condition)
```

```
##  Factor w/ 2 levels "SNF2","WT": 1 1 1 1 1 2 2 2 2 2
```

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

```
##  Factor w/ 2 levels "WT","SNF2": 2 2 2 2 2 1 1 1 1 1
```

Analysis design

```
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)
# this fits a negative binomial GLM and applies Wald statistics to each gene's
# estimated logFC values comparing the conditions/groups of interest
DESeq.ds <- nbinomWaldTest(DESeq.ds)
```

Extract the base means across samples, log2 fold changes, standard errors, test statistics, p-values and adjusted p-values for every gene using `results()`.

```
resultsNames(DESeq.ds) # tells you which types of values can be extracted with results()
```

```
## [1] "Intercept"           "condition_SNF2_vs_WT"
```

```
DGE.results <- results(DESeq.ds,
                       independentFiltering = TRUE,
                       alpha = 0.05)
head(DGE.results) # first line indicates which comparison was done for the log2FC
```

```
## 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>
## YDL248W    74.4718697762033  0.129095527093782 0.167085403420346
## YDL247W.A 0.647376806025316   1.59097150906673  1.70514818230034
```

```
## YDL247W     4.4875284684919 -0.752929436683315  0.58828323808097
## YDL246C    2.86787145342621  0.641505335792843 0.734205386009157
## YDL245C    5.07165502474423 -0.472798016529886 0.603781328037129
## YDL244W    31.6256425299895   1.51361325204893 0.267459102601866
##                       stat               pvalue                 padj
##                  <numeric>            <numeric>            <numeric>
## YDL248W     0.77263198610479      0.4397402089234    0.564150226073357
## YDL247W.A  0.933040028767717     0.35079930123511                   NA
## YDL247W    -1.27987572642633    0.200588845762314    0.308670301391428
## YDL246C    0.873740983132535    0.382259326661834    0.503897995585226
## YDL245C   -0.783061672455051    0.433590896902652    0.557729921289971
## YDL244W     5.65923252311984 1.52051427751018e-08 1.51367785672758e-07
```

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

```
##
## out of 6680 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 1306, 20%
## LFC < 0 (down)     : 1464, 22%
## outliers [1]       : 0, 0%
## low counts [2]     : 259, 3.9%
## (mean count < 1)
## [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)
```

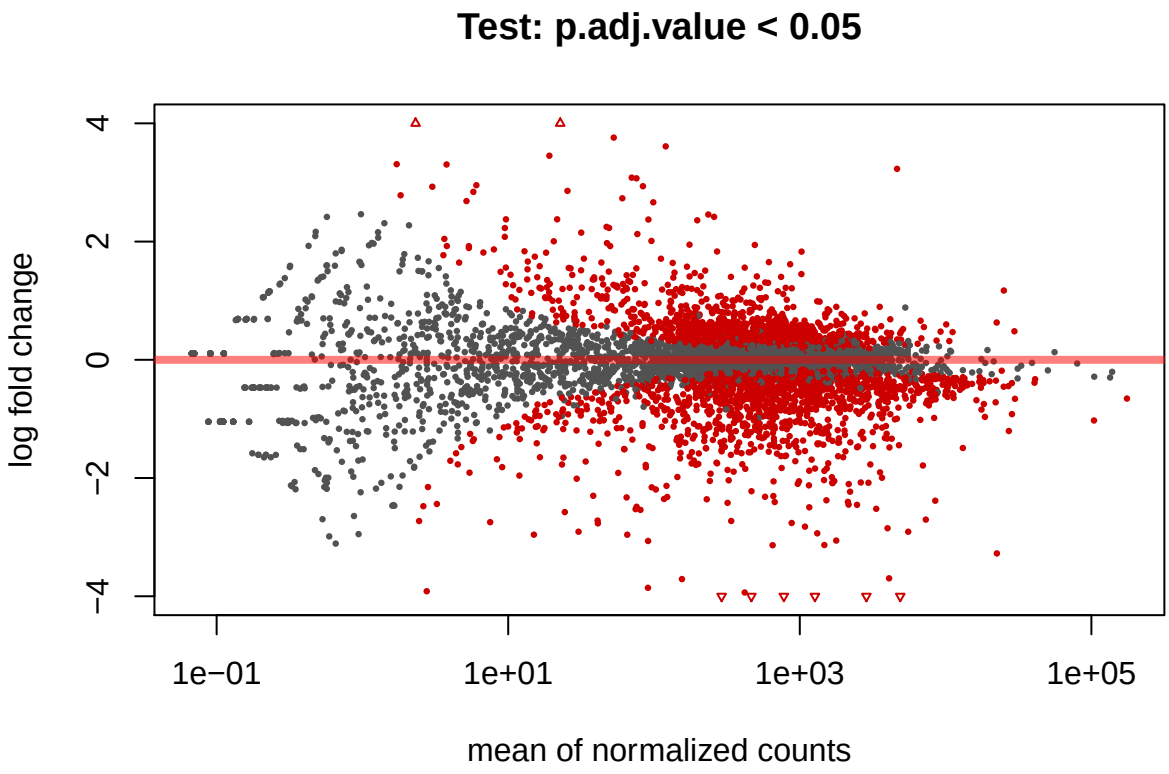
```
##
## FALSE  TRUE
##  3651  2770
```

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 was very lowly expressed].

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.

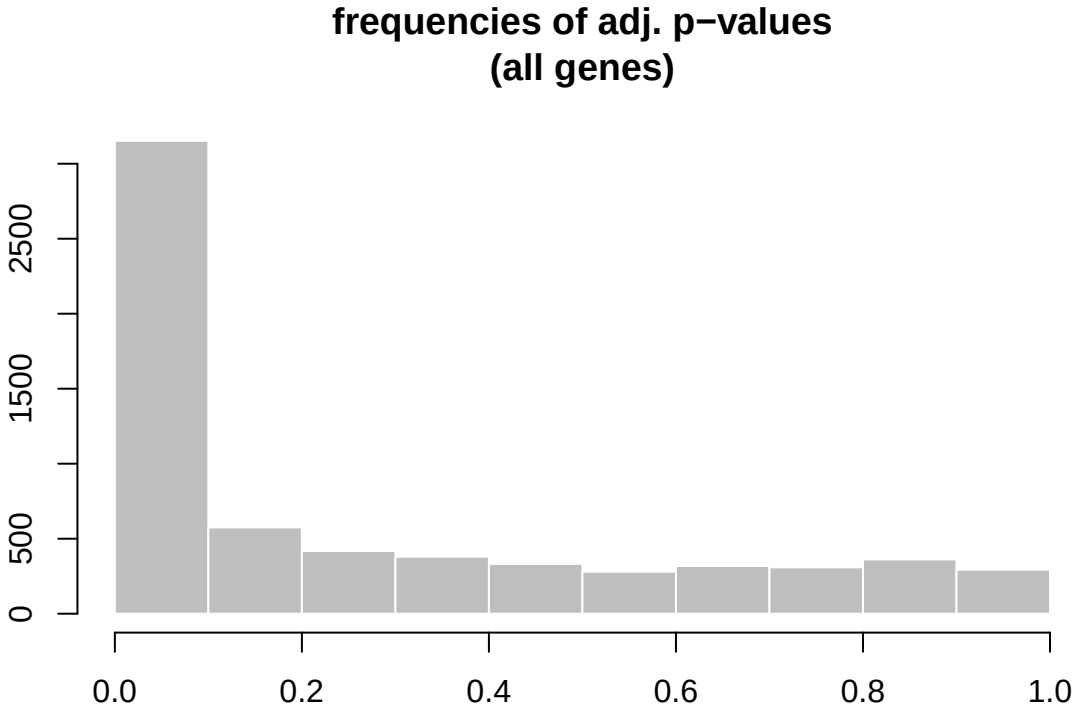
Genes that pass the significance threshold (adjusted p.value <0.05) are colored in red.

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



A adj. p-value histogram:

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



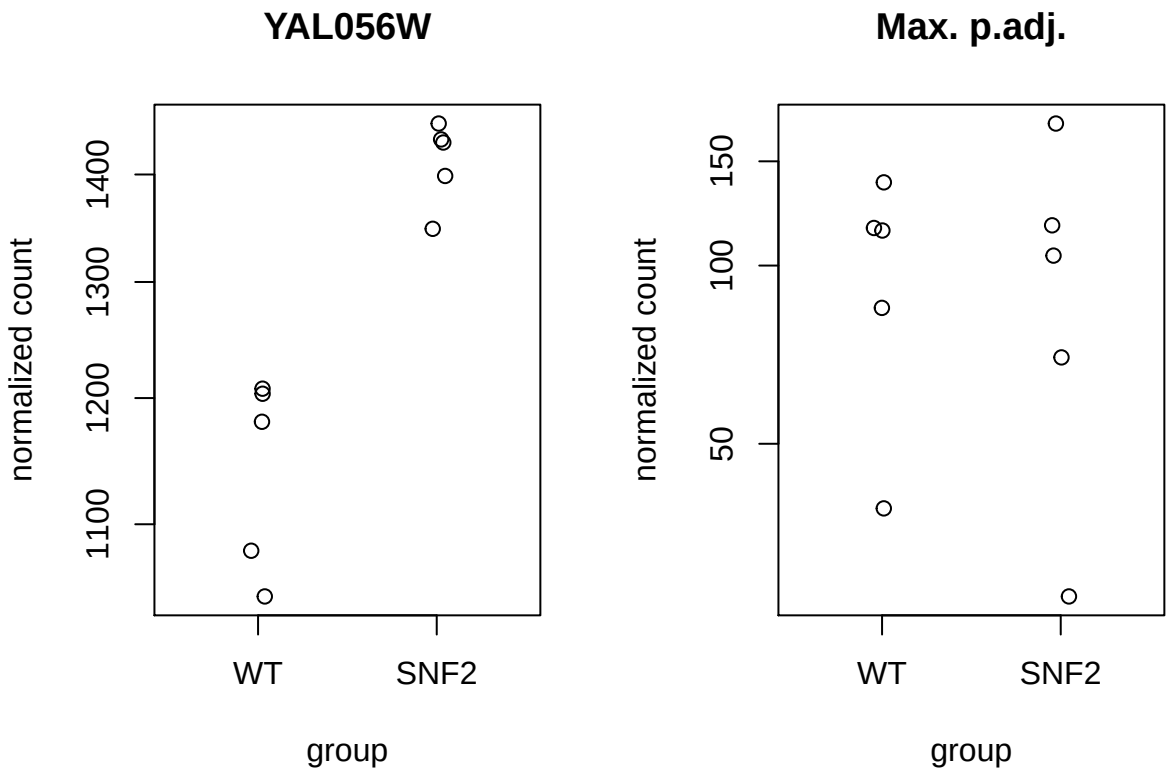
A sorted results table so that we can immediately see which genes come up as the best candidates:

```
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 2862.51247211677 -4.23894250364827  0.107457023017786
## YDR033W 4096.70096241307 -3.69679758799383 0.0989610470227823
## YOR290C 777.496644010726 -6.91948752004275  0.186534916249946
## YIL121W 879.508777465599 -2.76056917919252  0.088499594119629
## YML123C 4886.51910770664 -4.68229556156666  0.150245451570332
## YHR215W 290.985494848713  -4.5227277739132  0.153532465471962
##                      stat                pvalue                 padj
##                 <numeric>             <numeric>            <numeric>
## YGR234W -39.4477939608159                     0                    0
## YDR033W -37.3560880691043 2.01938473217832e-305 6.48323468265851e-302
## YOR290C -37.0948649140359 3.39945859447714e-301 7.27597454504592e-298
## YIL121W -31.1930151392664 1.32513905669646e-213   2.127179470762e-210
## YML123C -31.1643082211698 3.24618197323416e-213 4.16874689002731e-210
## YHR215W -29.4577942196801 1.00044864893231e-190 1.07064679579906e-187
```

Plotting counts for single genes (seq. depth normalized, log2-transformed)

```
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.")
```

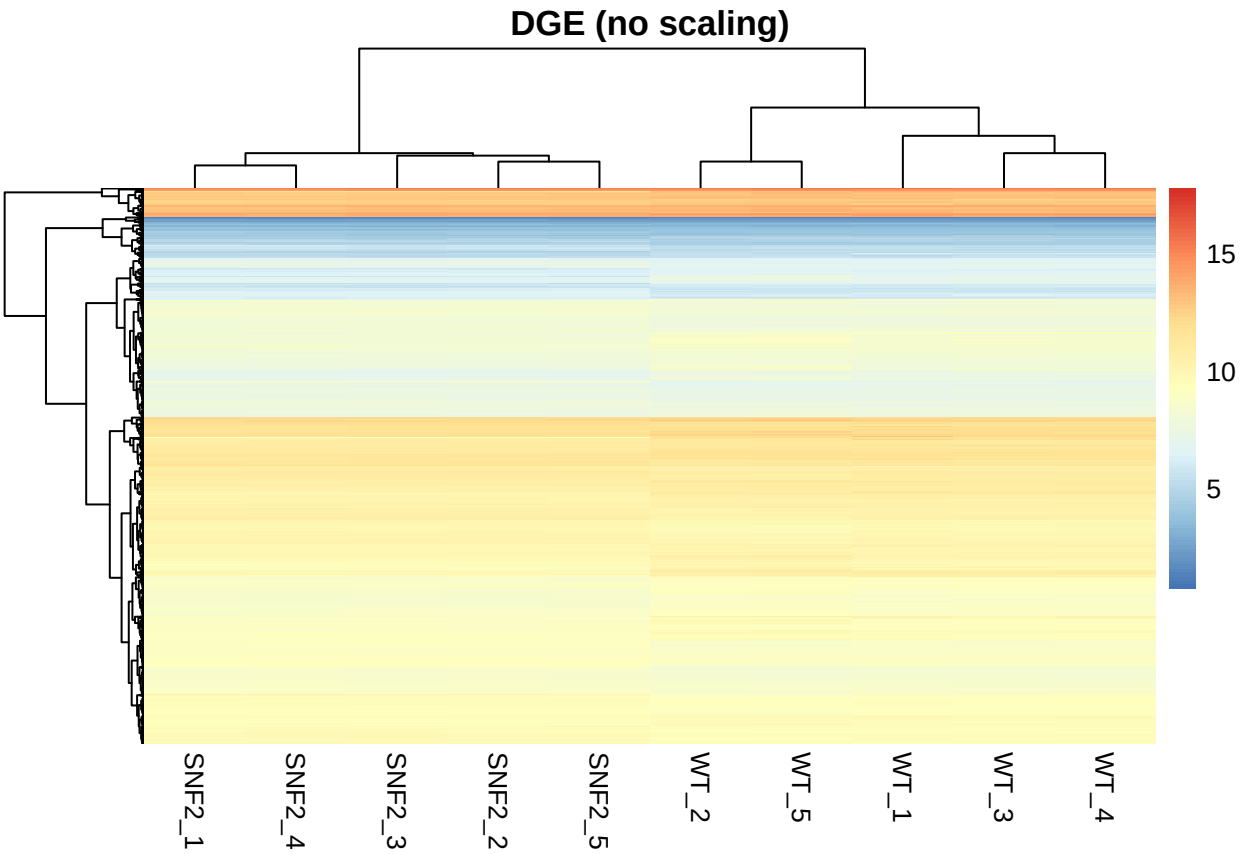


`plotCounts` simply uses `counts(dds, normalized = TRUE) + 0.5`.

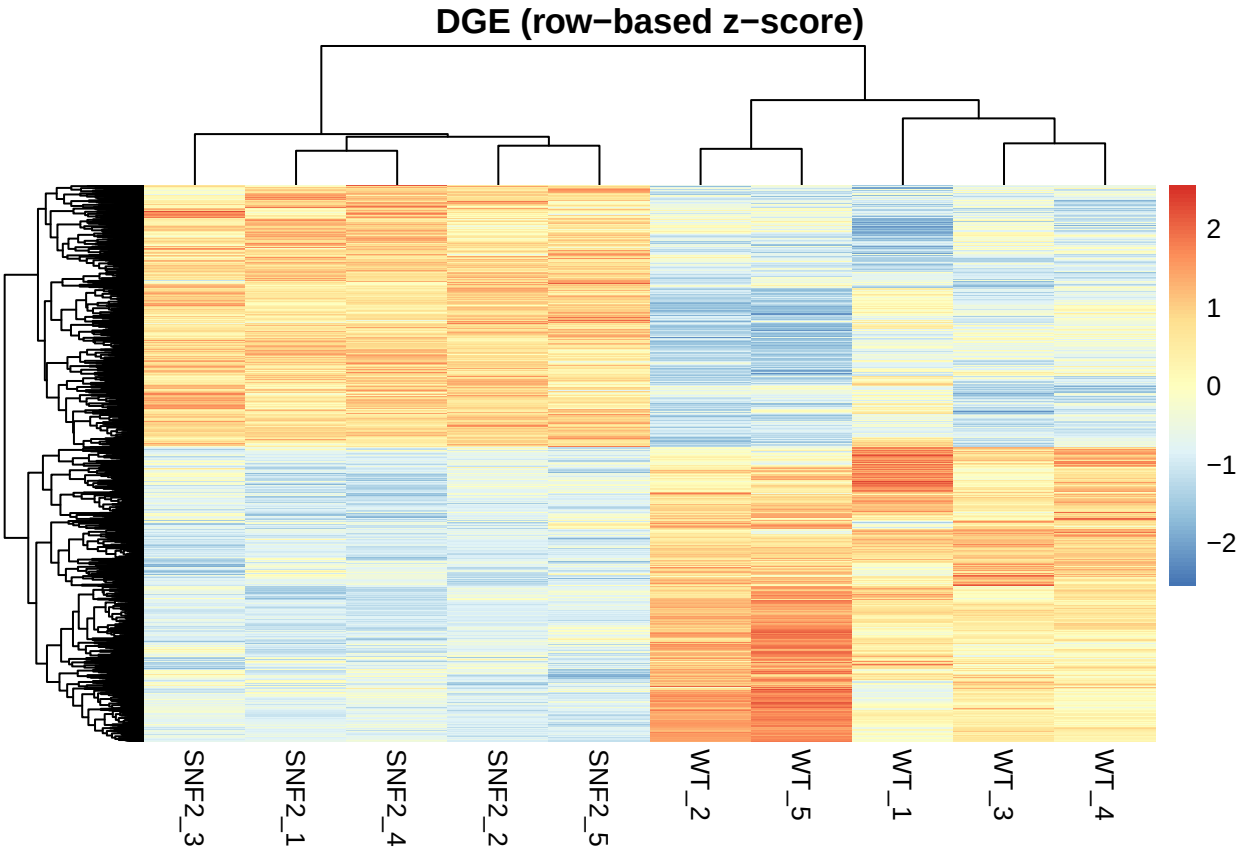
You can also use `pcaExplorer` for individual gene plots of `rlog` values.

A heatmap of the genes that show differential expression with adjusted p-value <0.05 :

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

# extract rlog-transformed values of DE genes 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)")
```



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 bioconductor.

We will go with `org.Sc.sgd.db`.

```
#source("http://bioconductor.org/biocLite.R")
#biocLite("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"
```

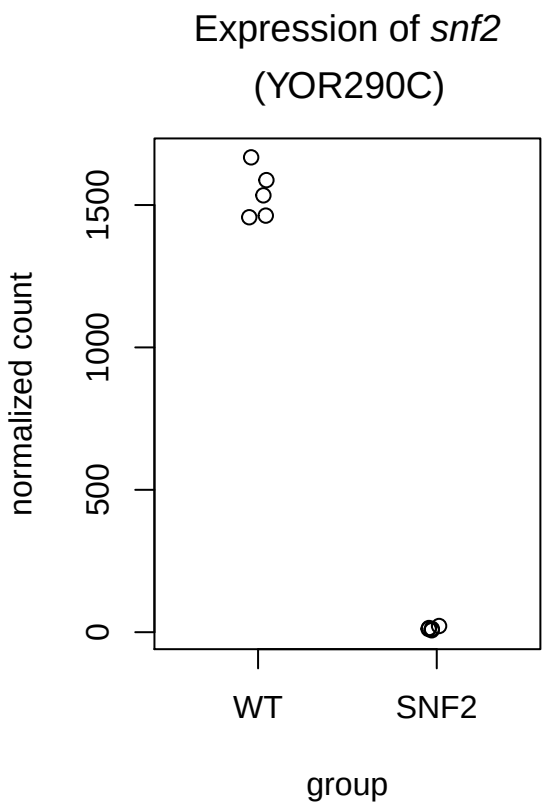
```
## [13] "GOALL"        "INTERPRO"     "ONTOLOGY"     "ONTOLOGYALL"
## [17] "ORF"          "PATH"         "PFAM"         "PMID"
## [21] "REFSEQ"       "SGD"          "SMART"        "UNIPROT"
```

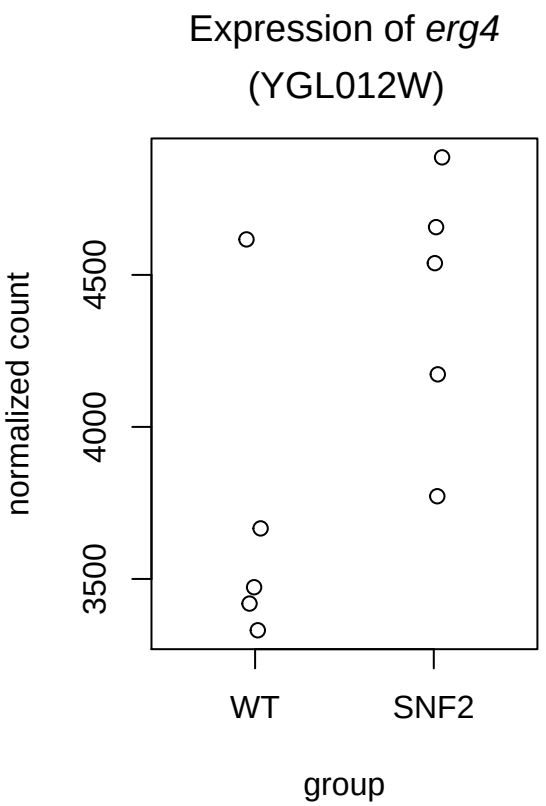
```
# make a batch retrieval for all DE genes
DGEgenes <- rownames(subset(DGE.results.sorted, padj < 0.05))

anno <- select(org.Sc.sgd.db,
               keys = rownames(DESeq.ds), # rownames
               keytype="ORF", # our rownames are ORF identifiers
               columns=c("SGD","GENENAME","ENSEMBL")) # what to return

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

```
##         ORF        SGD GENENAME ENSEMBL
## 1   YDL248W S000002407     COS7 YDL248W
## 2 YDL247W.A       <NA>     <NA>    <NA>
## 3   YDL247W S000002406     MPH2 YDL247W
## 4   YDL246C S000002405     SOR2 YDL246C
## 5   YDL245C S000002404    HXT15 YDL245C
## 6   YDL244W S000002403    THI13 YDL244W
```

To get a feeling for how the difference between WT and snf2 ko looks like for a housekeeping gene, let's repeat the exercise.

```
par(mfrow=c(1,2))
plotCounts(dds = DESeq.ds,
           gene = "YOR290C",
           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)")))
```





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

```
out.df <- merge(as.data.frame(DGE.results), 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)
```

```
save.image(file = "~/Documents/Teaching/2019_RNA-seq/Rclass.RData")
```