GO term enrichment analyses of DEG

Friederike Dündar

We need the results of the DESeq2 analysis (`DGE.results`) and the vector of differentially expressed genes (`DGE.genes`).

```
load("~/Documents/Teaching/2019_RNA-seq/Rclass.RData")
DESeq.ds <- DESeq(DESeq.ds)
DGE.results <- results(DESeq.ds,independentFiltering = TRUE, alpha = 0.05)
DGEgenes <- rownames(subset(DGE.results, padj < 0.05))
head(DGEgenes)
```

```
## [1] "YDL244W" "YDL241W" "YDL233W" "YDL229W" "YDL227C" "YDL222C"
```

GO term enrichment using goseq

Among our list of DE genes, which GO terms are enriched?

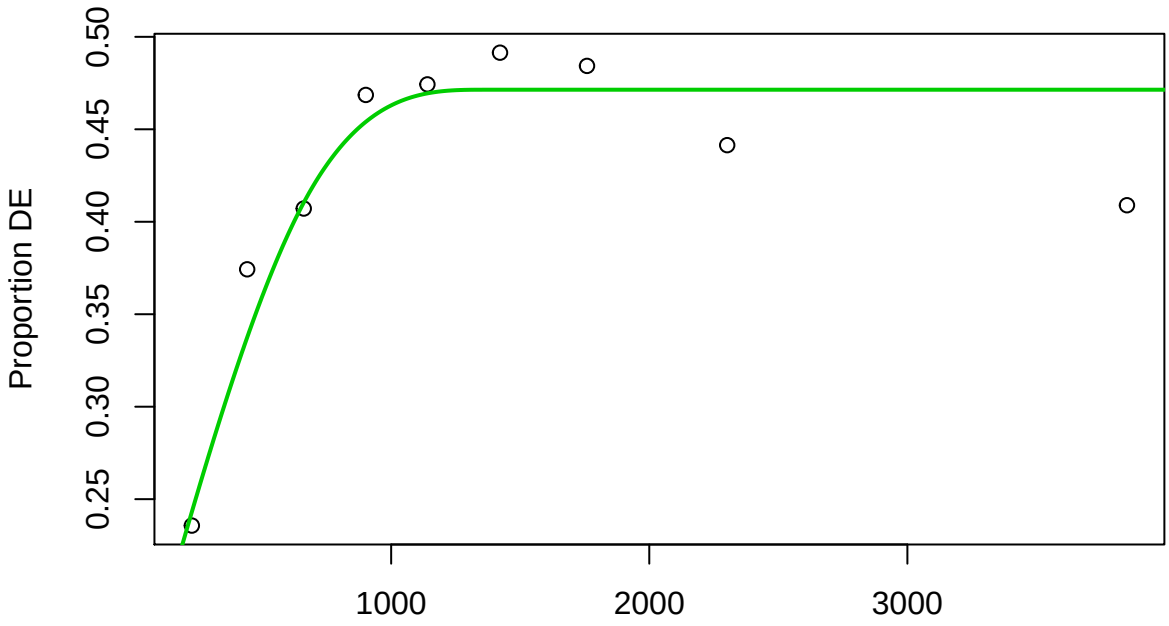
Transcripts that are longer or more highly expressed give more statistical power for detecting differential expression between samples

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

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

# 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 700 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), '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] 437178      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)
```

REVIGO generates multiple plots; the easiest way to obtain them in high quality is to download the R scripts that it offers and to run those yourself.

```
## this will generate a PDF file named "REVIGO_treemap.pdf" in your current
## working directory
source("~/Downloads/REVIGO_treemap.r")
```

Since I personally don't like plots being immediately printed to PDF (much more difficult to include them in an Rmarkdown!), I've tweaked the function a bit; it's essentially the original REVIGO script that I downloaded minus the part where the `revigo.data` object is generated and with a couple more options to tune the resulting heatmap.

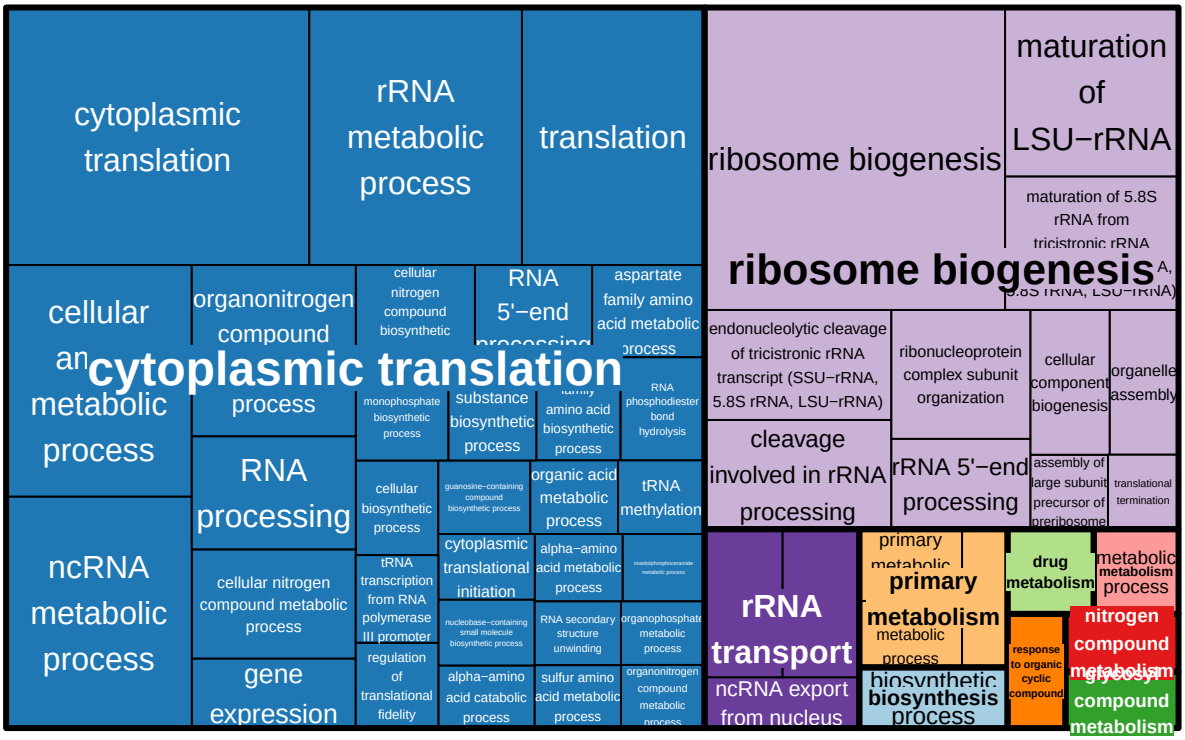
```
REVIGO_treemap <- function(revigo.data, col_palette = "Paired",
                           title = "REVIGO Gene Ontology treemap", ...){
```

```
  stuff <- data.frame(revigo.data)
  names(stuff) <- c("term_ID","description","freqInDbPercent","abslog10pvalue",
                    "uniqueness","dispensability","representative")

  stuff$abslog10pvalue <- as.numeric( as.character(stuff$abslog10pvalue) )
  stuff$freqInDbPercent <- as.numeric( as.character(stuff$freqInDbPercent) )
  stuff$uniqueness <- as.numeric( as.character(stuff$uniqueness) )
  stuff$dispensability <- as.numeric( as.character(stuff$dispensability) )

  # check the treemap command documentation for all possible parameters -
  # there are a lot more
  treemap::treemap(
    stuff,
    index = c("representative","description"),
    vSize = "abslog10pvalue",
    type = "categorical",
    vColor = "representative",
    title = title,
    inflate.labels = FALSE,
    lowerbound.cex.labels = 0,
    bg.labels = 255,
    position.legend = "none",
    fontsize.title = 22, fontsize.labels=c(18,12,8),
    palette= col_palette, ...
  )

}
```

I still need the originally downloaded script to generate the `revigo.data` object, which I will then pass onto my newly tweaked function. Using the command line (!) tools `sed` and `egrep`, I'm going to only keep the lines between the one starting with "revigo.data" and the line starting with "stuff". The output of that will be parsed into a new file, which will only generate the `revigo.data` object that I'm after (no PDF!).

```
# the system function allows me to run command line stuff outside of R
# just for legibility purposes, I'll break up the command into individual
# components, which I'll join back together using paste()
sed_cmd <- "sed -n '/^revigo\\.data.*/,/^stuff.*/p'"
fname <- "~/Downloads/REVIGO_treemap.r"
egrep_cmd <- "egrep '^rev|^c'"
out_fname <- "~/Downloads/REVIGO_myData.r"

system(paste(sed_cmd, fname, "|", egrep_cmd, ">", out_fname))

## upon sourcing; no treemap PDF will be generated, but the revigo.data object
## should appear in your R environment
source("~/Downloads/REVIGO_myData.r")
REVIGO_treemap(revigo.data)
```

REVIGO Gene Ontology treemap



Gene set enrichment of KEGG pathways using `ClusterProfiler`

ClusterProfiler actually offers access to a fairly sprawling set of R packages that all deal with over-representation analyses and gene set enrichment analyses. Since we just showed you how to do over-representation analyses using `goseq` and REVIGO, we're going to focus on highlighting a function that allows you to perform GSEA using the pathways annotated and curated by KEGG.

To this end, you will need to provide a full list of all genes that you analyzed. Those genes should be sorted by some sort of metric that you feel is a meaningful representation of the signal you're after, e.g. log2FC or a combination of log2FC and p-value or the like. This sorted vector of gene-value pairs is then compared to the KEGG pathways and those pathways that seem to have a consistent enrichment for positive (or negative) fold changes across all of the pathway's genes will be highlighted.

The function that clusterProfiler relies on for GSEA is the `fgsea` function of the package of the same name, which, in turn, is a more efficient implementation of the original GSEA algorithm introduced by Subramanian et al. (2005).

The main advantage of using clusterProfiler comes from its implementation of numerous helpful plots. In addition, it takes care of downloading the appropriate annotation from KEGG (or any other data base for which it offers functionalities).

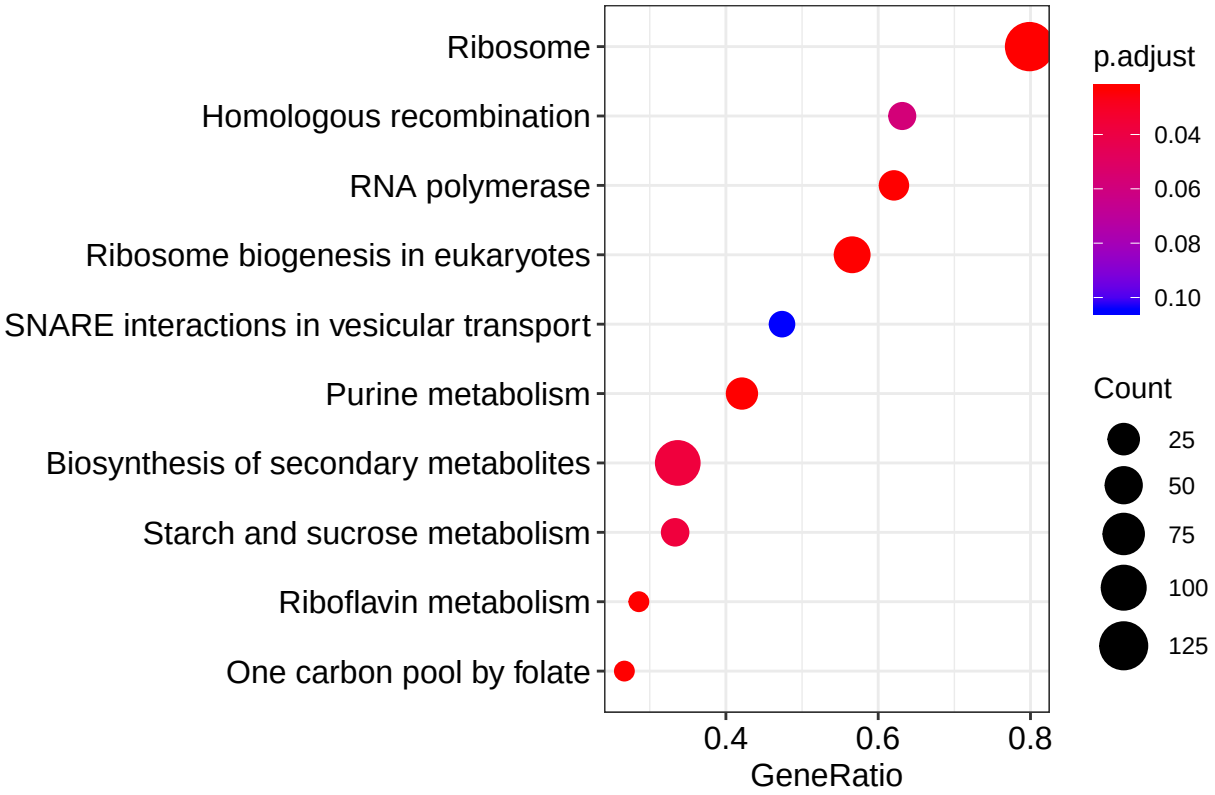
```
library(clusterProfiler)

## clusterProfiler requires a sorted vector where the values correspond
## to the measure used for the sorting
DGE.results <- DGE.results[order(-1*DGE.results$log2FoldChange),]
genes_for_cp <- DGE.results$log2FoldChange
names(genes_for_cp) <- row.names(DGE.results)
```

```
## run the gene set enrichment analysis
gsea_kegg <- clusterProfiler::gseKEGG(geneList = genes_for_cp, organism = 'sce',
                                      nPerm = 1000, minGSSize = 10,
                                      pvalueCutoff = 1, verbose = FALSE)
```

Dot plots:

Dot plots depict the enrichment scores and gene counts per gene set (for the most significant gene sets).

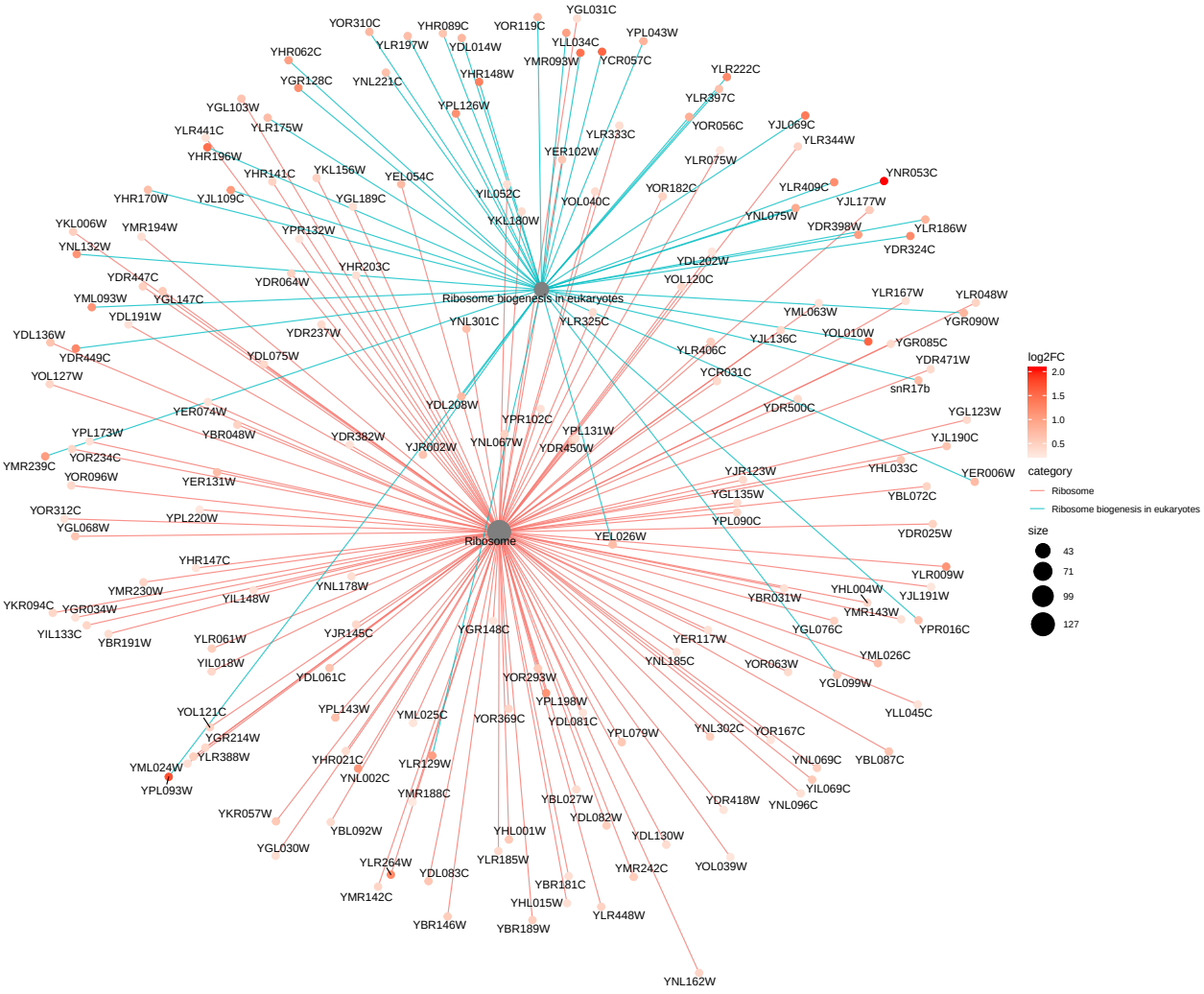
```
dotplot(gsea_kegg)
```



Cnetplots depict the linkages of genes and biological concepts (e.g. GO terms or KEGG pathways) as a network.

- nodes = genes (of the top 5 most sign. GO terms by default)
- edges = indicate whether a gene belongs to a given gene set
- size of the GO terms = number of genes belong to a given category

```
cnetplot(gsea_kegg,
  showCategory = 2, colorEdge = TRUE, foldChange = genes_for_cp) +
 scale_colour_gradient2(name = "log2FC",
   low = "navyblue", high = "red", mid = "white")
```



This result shows one of the reasons why the gene sets “Ribosome biogenesis in eukaryotes” and “Ribosome” both show up at the top of our list: they have many genes in common. In fact, it seems as if almost all genes of “Ribosome biogenesis in eukaryotes” are also part of “Ribosome”, which makes sense, but is an important detail to note.

See the documentation of clusterProfiler for more types of analyses and plots.