Introduction to Bulk RNAseq data analysis

Gene Set Testing for RNA-seq

Last modified: 23 Oct 2023

Contents

The list of differentially expressed genes is sometimes so long that its interpretation becomes cumbersome and time consuming. It may also be very short while some genes have low p-value yet higher than the given threshold.

A common downstream procedure to combine information across genes is gene set testing. It aims at finding pathways or gene networks the differentially expressed genes play a role in.

Various ways exist to test for enrichment of biological pathways. We will look into over representation and gene set enrichment analyses.

A gene set comprises genes that share a biological function, chromosomal location, or any other relevant criterion.

To save time and effort there are a number of packages that make applying these tests to a large number of gene sets simpler, and which will import gene lists for testing from various sources.

Today we will use `clusterProfiler`.

Over-representation

Method

This method tests whether genes in a pathway are present in a subset of our data in a higher number than expected by chance (explanations derived from the clusterProfiler manual).

Genes in the experiment are split in two ways:

- annotated to the pathway or not
- differentially expressed or not

We can then create a contingency table with:

- rows: genes in pathway or not
- columns: genes differentially expressed or not

And test for independence of the two variables with the Fisher exact test.

`clusterProfiler`

`clusterprofiler` (Yu et al. 2012) supports direct online access of the current KEGG database (KEGG: Kyoto Encyclopedia of Genes and Genomes), rather than relying on R annotation packages. It also provides some nice visualisation options.

We first search the resource for mouse data:

```
library(tidyverse)
library(clusterProfiler)

search_kegg_organism('mouse', by='common_name')
```

```
##      kegg_code                            scientific_name
## 20        mmur                          Microcebus murinus
## 22         mmu                                Mus musculus
## 23        mcal                                  Mus caroli
## 24        mpah                                  Mus pahari
## 26        mcoc                             Mastomys coucha
## 29        pleu                          Peromyscus leucopus
## 85        mmyo                                Myotis myotis
## 5095       asf Candidatus Arthromitus sp. SFB-mouse-Japan
## 5096       asm   Candidatus Arthromitus sp. SFB-mouse-Yit
## 5097       aso    Candidatus Arthromitus sp. SFB-mouse-NL
##                                    common_name
## 20                            gray mouse lemur
## 22                                 house mouse
## 23                                Ryukyu mouse
## 24                                 shrew mouse
## 26                 southern multimammate mouse
## 29                         white-footed mouse
## 85                      greater mouse-eared bat
## 5095 Candidatus Arthromitus sp. SFB-mouse-Japan
## 5096   Candidatus Arthromitus sp. SFB-mouse-Yit
## 5097    Candidatus Arthromitus sp. SFB-mouse-NL
```

We will use the 'mmu' 'kegg_code'.

KEGG enrichment analysis

The input for the KEGG enrichment analysis is the list of gene IDs of significant genes.

We now load the R object keeping the outcome of the differential expression analysis for the d11 contrast.

```
shrink.d11 <- readRDS("RObjects/Shrunk_Results.d11.rds")
```

We will only use genes that have:

- an adjusted p-value (FDR, False Discovery Rate) of less than 0.05
- and an absolute fold change greater than 2.

We need to remember to eliminate genes with missing values in the FDR as a result of the independent filtering by DESeq2.

For this tool we need to use Entrez IDs, so we will also need to eliminate genes with a missing Entrez ID (NA values in the 'Entrez' column).

```
sigGenes <- shrink.d11 %>%
    drop_na(Entrez, padj) %>%
    filter(padj < 0.05 & abs(log2FoldChange) > 1) %>%
    pull(Entrez)

keggRes <- enrichKEGG(gene = sigGenes, organism = 'mmu')
as_tibble(keggRes)
```

```
## # A tibble: 73 x 9
##    ID      Description GeneRatio BgRatio   pvalue p.adjust   qvalue geneID Count
##    <chr>   <chr>       <chr>     <chr>      <dbl>    <dbl>    <dbl> <chr>  <int>
##  1 mmu046~ Antigen pr~ 40/346    90/9370 6.03e-34 1.42e-31 1.01e-31 14991~    40
##  2 mmu051~ Epstein-Ba~ 56/346    231/93~ 5.84e-31 6.89e-29 4.89e-29 12502~    56
##  3 mmu053~ Graft-vers~ 32/346    63/9370 1.12e-29 8.84e-28 6.27e-28 14939~    32
##  4 mmu049~ Type I dia~ 33/346    70/9370 3.39e-29 2.00e-27 1.42e-27 16160~    33
##  5 mmu041~ Phagosome ~ 48/346    182/93~ 2.31e-28 1.09e-26 7.73e-27 16414~    48
##  6 mmu053~ Allograft ~ 31/346    63/9370 3.22e-28 1.26e-26 8.97e-27 16160~    31
##  7 mmu051~ Influenza ~ 45/346    173/93~ 2.43e-26 8.20e-25 5.81e-25 21706~    45
##  8 mmu045~ Cell adhes~ 45/346    183/93~ 3.21e-25 9.48e-24 6.72e-24 16414~    45
##  9 mmu054~ Viral myoc~ 32/346    88/9370 5.29e-24 1.39e-22 9.85e-23 16414~    32
## 10 mmu051~ Leishmania~ 28/346    70/9370 1.79e-22 4.21e-21 2.99e-21 16414~    28
## # i 63 more rows
```

Visualise a pathway in a browser

`clusterProfiler` has a function `browseKegg` to view the KEGG pathway in a browser, highlighting the genes we selected as differentially expressed.

We will show one of the top hits: pathway 'mmu04612' for 'Antigen processing and presentation'.

```
browseKEGG(keggRes, 'mmu04612')
```

Visualise a pathway as a file

The package `pathview` (Luo et al. 2013) can be used to generate figures of KEGG pathways.

One advantage over the `clusterProfiler` browser method `browseKEGG` is that genes can be coloured according to fold change levels in our data. To do this we need to pass `pathview` a named vector of fold change values (one could in fact colour by any numeric vector, e.g. p-value).

The package plots the KEGG pathway to a `png` file in the working directory.

```
library(pathview)
logFC <- shrink.d11$log2FoldChange
names(logFC) <- shrink.d11$Entrez
pathview(gene.data = logFC,
         pathway.id = "mmu04612",
         species = "mmu",
         limit = list(gene=20, cpd=1))
```

mmu04612.pathview.png:

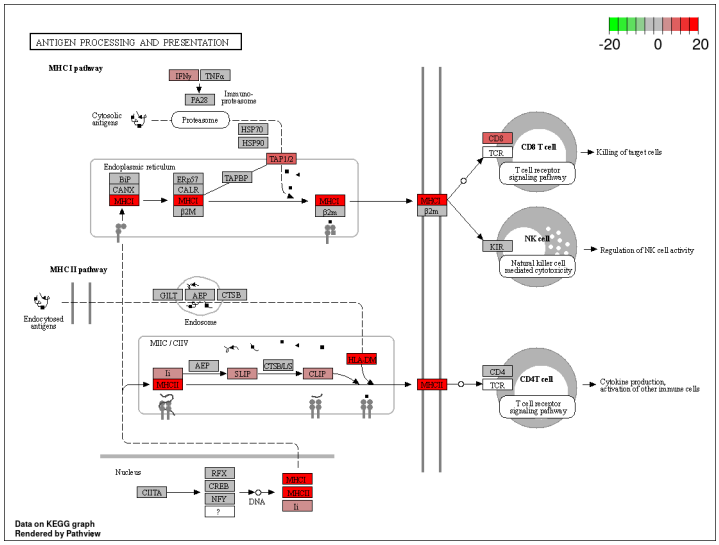


Figure 1: mmu04612 - Antigen processing and presentation

Exercise 1

1. Use `pathview` to export a figure for "mmu04659" or "mmu04658", but this time only use genes that are statistically significant at padj < 0.01

GO term enrichment analysis

`clusterProfiler` can also perform over-representation analysis on GO terms using the command `enrichGO`. For this analysis we will use Ensembl gene IDs instead of Entrez IDs and in order to do this we need to load another package which contains the mouse database called `org.Mm.eg.db`.

To run the GO enrichment analysis, this time we also need a couple of extra things. Firstly, we should provide a list of the 'universe' of all the genes in our DE analysis not just the ones we have selected as significant.

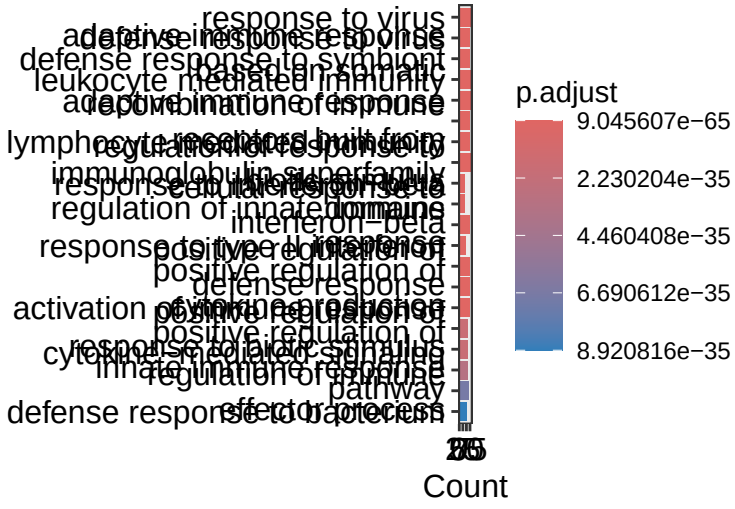
Gene Ontology terms are divided into 3 categories. - Metabolic Functions - Biological Processes - Cellular Components

For this analysis we will narrow our search terms in the 'Biological Processes' Ontology so we can add the parameter "BP" with the 'ont' argument (the default is Molecular Functions).

```
library(org.Mm.eg.db)

sigGenes_GO <-  shrink.d11 %>%
    drop_na(padj) %>%
    filter(padj < 0.01 & abs(log2FoldChange) > 2) %>%
    pull(GeneID)

universe <- shrink.d11$GeneID

ego <- enrichGO(gene          = sigGenes_GO,
                universe      = universe,
                OrgDb         = org.Mm.eg.db,
                keyType       = "ENSEMBL",
                ont           = "BP",
                pvalueCutoff  = 0.01,
                readable      = TRUE)
```

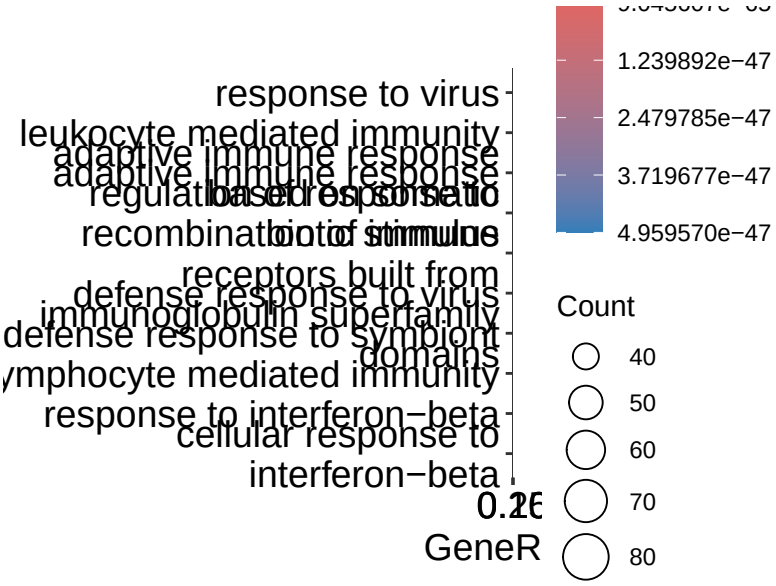
We can use the `barplot` function to visualise the results. Count is the number of differentially expressed in each gene ontology term.

```
barplot(ego, showCategory=20)
```



or perhaps the `dotplot` version is more informative. Gene ratio is Count divided by the number of genes in that GO term.

```
dotplot(ego, font.size = 14)
```

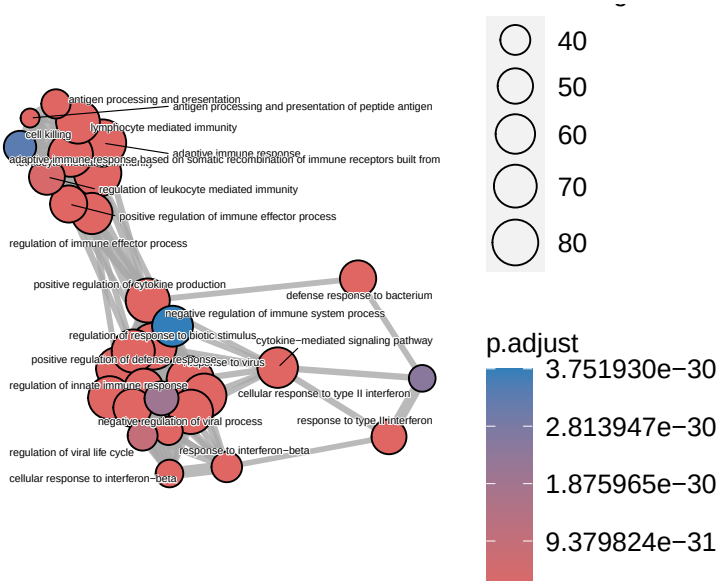


Another visualisation that can be nice to try is the `emapplot` which shows the overlap between genes in the different GO terms.

```
library(enrichplot)
ego_pt <- pairwise_termsim(ego)
emapplot(ego_pt, cex_label_category = 0.25)
```

```
## Warning in emapplot.enrichResult(x, showCategory = showCategory, ...): Use 'cex.params = list(catego
##  The cex_label_category parameter will be removed in the next version.
```

```
## Warning: ggrepel: 6 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



GSEA analysis

Gene Set Enrichment Analysis (GSEA) identifies gene sets that are enriched in the dataset between samples (Subramanian et al. 2005).

The software is distributed by the Broad Institute and is freely available for use by academic and non-profit organisations. The Broad also provide a number of very well curated gene sets for testing against your data - the Molecular Signatures Database (MSigDB). These are collections of human genes. Fortunately, these lists have been translated to mouse equivalents by the Walter+Eliza Hall Institute Bioinformatics service and made available for download. They are now also available from a recent R package msigdbr, which we will use.

Let's load `msigdbr` now.

```
library(msigdbr)
```

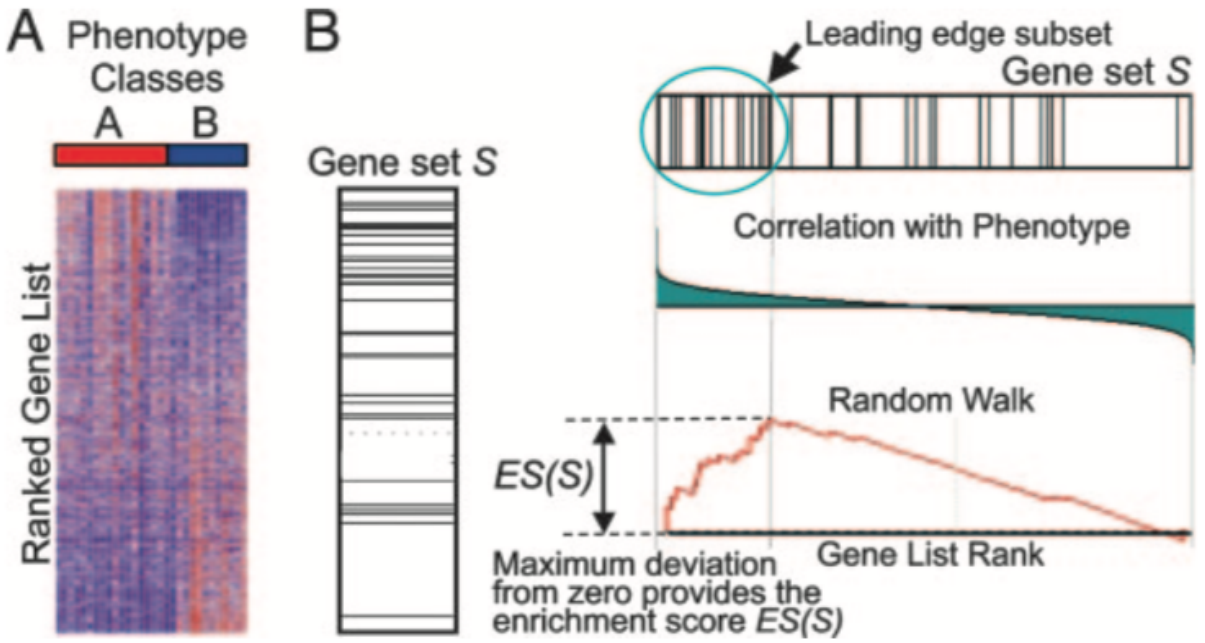
Method

The analysis is performed by:

1. ranking all genes in the data set

2. identifying in the ranked data set the rank positions of all members of the gene set
3. calculating an enrichment score (ES) that represents the difference between the observed rankings and that which would be expected assuming a random rank distribution.

The article describing the original software is available here, while this commentary on GSEA provides a shorter description.



We will use `clusterProfiler`'s `GSEA` package (Yu et al. 2012) that implements the same algorithm in R.

Rank genes

We need to provide `GSEA` with a vector containing values for a given gene mtric, e.g. log(fold change), sorted in decreasing order.

To start with we will simply use a rank the genes based on their fold change.

We must exclude genes with no Ensembl ID.

Also, we should use the shrunk LFC values.

```
rankedGenes <- shrink.d11 %>%
  drop_na(GeneID, padj, log2FoldChange) %>%
  mutate(rank = log2FoldChange) %>%
  arrange(desc(rank)) %>%
  pull(rank, GeneID)
```

Load pathways

We will load the MSigDB Hallmark gene set with `msigdbr`, setting the `category` parameter to 'H' for **H**allmark gene set. The object created is a `tibble` with information on each {gene set; gene} pair (one per row). We will only keep the the gene set name, gene Ensembl ID.

```
term2gene <- msigdbr(species = "Mus musculus", category = "H") %>%
  dplyr::select(gs_name, ensembl_gene)
term2name <- msigdbr(species = "Mus musculus", category = "H") %>%
  dplyr::select(gs_name, gs_description) %>%
  distinct()
```

Conduct analysis

Arguments passed to `GSEA` include:

- ranked genes
- pathways
- gene set minimum size
- gene set maximum size

```
gseaRes <- GSEA(rankedGenes,
                TERM2GENE = term2gene,
                TERM2NAME = term2name,
                pvalueCutoff = 1.00,
                minGSSize = 15,
                maxGSSize = 500)
```

```
## preparing geneSet collections...

## GSEA analysis...

## leading edge analysis...

## done...
```

Let's look at the top 10 results.

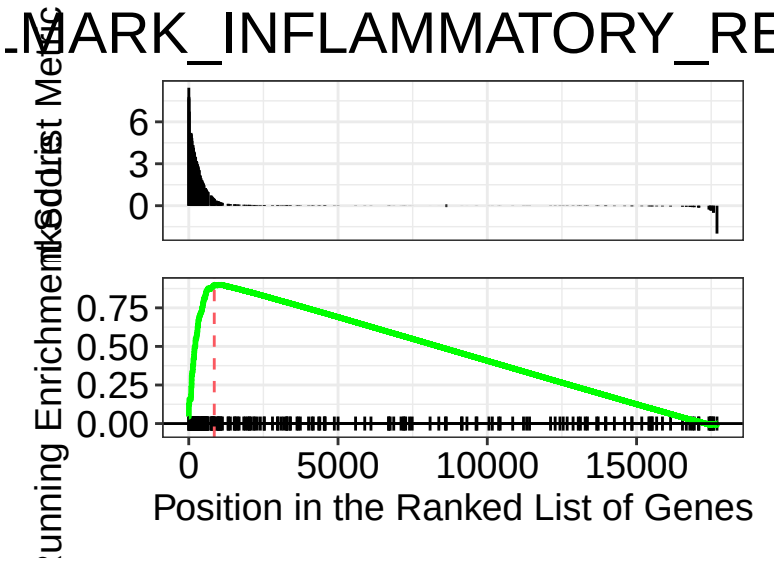
```
as_tibble(gseaRes) %>%
  arrange(desc(abs(NES))) %>%
  top_n(10, wt=-p.adjust) %>%
  dplyr::select(-core_enrichment) %>%
  mutate(across(c("enrichmentScore", "NES"), round, digits=3)) %>%
  mutate(across(c("pvalue", "p.adjust", "qvalue"), scales::scientific))
```

Enrichment score plot

The enrichment score plot displays along the x-axis that represents the decreasing gene rank:

- genes involved in the pathway under scrutiny: one black tick per gene in the pathway (no tick for genes not in the pathway)
- the enrichment score: the green curve shows the difference between the observed rankings and that which would be expected assuming a random rank distribution.

```
gseaplot(gseaRes,
         geneSetID = "HALLMARK_INFLAMMATORY_RESPONSE",
         title = "HALLMARK_INFLAMMATORY_RESPONSE")
```



Remember to check the GSEA article for the complete explanation.

Exercise 2

Another common way to rank the genes is to order by pvalue while sorting so that upregulated genes are at the start and downregulated at the end. You can do this combining the sign of the fold change and the pvalue.

1. Rank the genes by statistical significance - you will need to create a new ranking value using `-log10(pvalue) * sign(log2FoldChange)`.
2. Run GSEA using the new ranked genes and the H pathways.

3. Conduct the same analysis for the day 33 Infected vs Uninfected contrast.

References

Luo, Weijun, Brouwer, and Cory. 2013. "Pathview: An R/Bioconductor Package for Pathway-Based Data Integration and Visualization." *Bioinformatics* 29 (14): 1830–1. https://doi.org/10.1093/bioinformatics/btt285.

Subramanian, Aravind, Pablo Tamayo, Vamsi K. Mootha, Sayan Mukherjee, Benjamin L. Ebert, Michael A. Gillette, Amanda Paulovich, et al. 2005. "Gene Set Enrichment Analysis: A Knowledge-Based Approach for Interpreting Genome-Wide Expression Profiles." *Proceedings of the National Academy of Sciences* 102 (43): 15545–50. https://doi.org/10.1073/pnas.0506580102.

Yu, Guangchuang, Li-Gen Wang, Yanyan Han, and Qing-Yu He. 2012. "ClusterProfiler: An R Package for Comparing Biological Themes Among Gene Clusters." *OMICS: A Journal of Integrative Biology* 16 (5): 284–87. https://doi.org/10.1089/omi.2011.0118.