Enrichment analysis

Material

- MSigDB
- `clusterProfiler` vignette
- Revigo
- Signaling Pathway Impact Analysis (SPIA)
- Original paper on GSEA
- STRING for protein-protein interactions
- GO figure! for plotting GO terms and the associated paper

Exercises

Load the following packages:

If the `FindMarkers` or `FindAllMarkers` functions were used, we obtained a table listing only the significant genes, but we don't have any information of fold change for the non-significant genes. Therefore, we can use the over-representation analysis which is a threshold-based method. Using our list of significant genes, we can test if any gene set is over-represented among significant genes or not using a test similar to a Fisher test to compare differences in proportions.

The `clusterProfiler` package provides functions for over-representation analysis of Gene Ontology gene sets (among other functions, including functions for actual GSEA) or KEGG gene sets.

Genes can be labeled using different types of labels, eg symbol, Ensembl ID, Entrez ID. To list the allowed label types use:

Code starts here:

```
# BiocManager::install("org.Hs.eg.db", update = FALSE)
library(org.Hs.eg.db)
AnnotationDbi::keytypes(org.Hs.eg.db)
```

Code ends here

About OrgDb

For other organisms, you can find available OrgDbs at bioconductor

Let's select a set of genes that are downregulated in the tumor cells compared to normal:

Code starts here:

```
tum_down  <- subset(limma_de,
                    limma_de$logFC < -1
                      & limma_de$adj.P.Val <  0.05)
tum_down_genes <- rownames(tum_down)
```

Code ends here

We can do a Gene Ontology term over-representation analysis based on this set of genes. Make sure you check out the help of this function to understand its arguments:

Code starts here:

```
?enrichGO

tum_vs_norm_go <- clusterProfiler::enrichGO(tum_down_genes,
                                            "org.Hs.eg.db",
                                            keyType = "SYMBOL",
                                            ont = "BP",
                                            minGSSize = 50)
```

The results are stored in the `@result` slot:

```
View(tum_vs_norm_go@result)
```

Code ends here

	ID	Description	GeneRatio	BgRatio	pvalue	p.adjust	qvalue
GO:0007059	GO:0007059	chromosome segregation	105/809	424/18870	0	0	0
GO:0098813	GO:0098813	nuclear chromosome segregation	79/809	312/18870	0	0	0
GO:0000070	GO:0000070	mitotic sister chromatid segregation	62/809	184/18870	0	0	0
GO:0000819	GO:0000819	sister chromatid segregation	67/809	225/18870	0	0	0
GO:0140014	GO:0140014	mitotic nuclear division	71/809	274/18870	0	0	0
GO:0000280	GO:0000280	nuclear division	88/809	441/18870	0	0	0

> The columns `GeneRatio` and `BgRatio`

> The columns `GeneRatio` and `BgRatio` that are in the `enrichResult` object represent the numbers that are used as input for the Fisher's exact test.
>
> The two numbers (M/N) in the `GeneRatio` column are:
>
> - M: Number of genes of interest (in our case `tum_down_genes`) that are in the GO set
> - N: Number of genes of interest with any GO annoation.
>
> The two numbers (k/n) in the `BgRatio` column are:
>
> - k: Number of genes in the universe that are in the GO set
> - n: Number of genes in the universe with any GO annoation
>
> A low p-value resulting from the Fisher's exact means that M/N is signficantly greater than k/n.

Some GO terms seem redundant because they contain many of the same genes, which is a characteristic of Gene Ontology gene sets. We can simplify this list by removing redundant gene sets:

Code starts here:

```
enr_go <- clusterProfiler::simplify(tum_vs_norm_go)

View(enr_go@result)
```

Code ends here

	ID	Description	GeneRatio	BgRatio	pvalue	p.adjust	qvalue
GO:0007059	GO:0007059	chromosome segregation	105/809	424/18870	0	0	0
GO:0098813	GO:0098813	nuclear chromosome segregation	79/809	312/18870	0	0	0
GO:0000070	GO:0000070	mitotic sister chromatid segregation	62/809	184/18870	0	0	0
GO:0000280	GO:0000280	nuclear division	88/809	441/18870	0	0	0
GO:0044772	GO:0044772	mitotic cell cycle phase transition	77/809	470/18870	0	0	0
GO:0051983	GO:0051983	regulation of chromosome segregation	40/809	131/18870	0	0	0

We can quite easily generate a plot called an enrichment map with the `enrichplot` package:

Code starts here:

```
enrichplot::emapplot(enrichplot::pairwise_termsim(enr_go),
                     showCategory = 30)
```

Code ends here

Instead of testing for Gene Ontology terms, we can also test for other gene set collections. For example the Hallmark collection from MSigDB:

Code starts here:

```
library(msigdbr)

Warning: package 'msigdbr' was built under R version 4.3.3

library(msigdbdf)
gmt <- msigdbr::msigdbr(species = "human", category = "H")

Warning: The `category` argument of `msigdbr()` is deprecated as of msigdbr 1
0.0.0.
ℹ Please use the `collection` argument instead.
```

Code ends here

We can use the function `enricher` to test for over-representation of any set of genes of the Hallmark collection. We have to include the "universe", i.e. the full list of background, non significant genes, against which to test for differences in proportions:

Code starts here:

```
tum_vs_norm_enrich <- clusterProfiler::enricher(gene = tum_down_genes,
                                                universe = rownames(proB),
                                                pAdjustMethod = "BH",
                                                pvalueCutoff  = 0.05,
                                                qvalueCutoff  = 0.05,
                                                TERM2GENE = gmt[,c("gs_name",
"gene_symbol")])
```

Code ends here

When using the genes down-regulated in tumor, among the over-represented Hallmark gene sets, we have `HALLMARK_G2M_CHECKPOINT`, which includes genes involved in the G2/M checkpoint in the progression through the cell division cycle.

Code starts here:

```
View(tum_vs_norm_enrich@result[tum_vs_norm_enrich@result$p.adjust < 0.05,])
```

Code ends here

	ID	Description	GeneRatio	BgRatio	pvalue	p.adjust	qvalue
HALLMARK_E2F_TARGETS	HALLMARK_E2F_TARGETS	HALLMARK_E2F_TARGETS	80/344	195/3858	0.000000000	0.000000000	0.000000000
HALLMARK_G2M_CHECKPOINT	HALLMARK_G2M_CHECKPOINT	HALLMARK_G2M_CHECKPOINT	67/344	188/3858	0.000000000	0.000000000	0.000000000
HALLMARK_MITOTIC_SPINDLE	HALLMARK_MITOTIC_SPINDLE	HALLMARK_MITOTIC_SPINDLE	48/344	197/3858	0.000000000	0.000000000	0.000000000
HALLMARK_MYC_TARGETS_V1	HALLMARK_MYC_TARGETS_V1	HALLMARK_MYC_TARGETS_V1	33/344	193/3858	0.001469	0.016889	0.015459
HALLMARK_ESTROGEN_RESPONSE_LATE	HALLMARK_ESTROGEN_RESPONSE_LATE	HALLMARK_ESTROGEN_RESPONSE_LATE	28/344	164/3858	0.004850	0.044623	0.040845

Clear environment

Clear your environment:

Code starts here:

```
rm(list = ls())
gc()
.rs.restartR()
```

Code ends here