DAY2 GSE198256

David Gomez-Cabrero

2024-January

Experimental design

Lets review experimental design from a practical perspective

```
# Read data
urld <- "https://www.ncbi.nlm.nih.gov/geo/download/?format=file&type=rnaseq_counts"
path <- paste(urld, "acc=GSE198256", "file=GSE198256_raw_counts_GRCh38.p13_NCBI.tsv.gz", sep="&");
GSE198256_count <- as.matrix(data.table::fread(path, header=T, colClasses="integer"), rownames=1)

# Read Meta data
library(GEOquery)
```

```
## Loading required package: Biobase
```

```
## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
## Setting options('download.file.method.GEOquery'='auto')
```

```
## Setting options('GEOquery.inmemory.gpl'=FALSE)
```

```
gds <- getGEO("GSE198256")
```

```
## Found 1 file(s)
```

```
## GSE198256_series_matrix.txt.gz
```

```
Meta_GSE198256 <- pData(gds$GSE198256_series_matrix.txt.gz@phenoData)
Group <- Meta_GSE198256[,c("disease state:ch1")]

dim(GSE198256_count)
```

```
## [1] 39376    34
```

```
Group
```

```
##  [1] "Healthy"                  "Healthy"
##  [3] "Healthy"                  "Healthy"
##  [5] "Healthy"                  "Healthy"
##  [7] "Healthy"                  "Healthy"
##  [9] "Healthy"                  "Healthy"
## [11] "Healthy"                  "Covid19: Acute infection"
## [13] "Covid19: Acute infection" "Covid19: Acute infection"
## [15] "Covid19: Acute infection" "Covid19: Acute infection"
## [17] "Covid19: Acute infection" "Covid19: Acute infection"
## [19] "Covid19: Recovery 3Mo"    "Covid19: Recovery 3Mo"
## [21] "Covid19: Recovery 3Mo"    "Covid19: Recovery 3Mo"
## [23] "Covid19: Recovery 3Mo"    "Covid19: Recovery 3Mo"
## [25] "Covid19: Recovery 6Mo"    "Covid19: Recovery 6Mo"
## [27] "Covid19: Recovery 6Mo"    "Covid19: Recovery 6Mo"
## [29] "Covid19: Recovery 6Mo"    "Covid19: Recovery 6Mo"
## [31] "Covid19: Recovery 6Mo"    "Covid19: Recovery 6Mo"
## [33] "Covid19: Recovery 6Mo"    "Covid19: Recovery 6Mo"
```

Limma: Normalize and set design

```
# set DGE class
require(limma)
```

```
## Loading required package: limma
```

```
##
## Attaching package: 'limma'
```

```
## The following object is masked from 'package:BiocGenerics':
##
##     plotMA
```

```
require(edgeR)
```

```
## Loading required package: edgeR
```

```
dge <- DGEList(counts=GSE198256_count)

# Make sure on the metadata
rownames(Meta_GSE198256)==colnames(GSE198256_count)
```

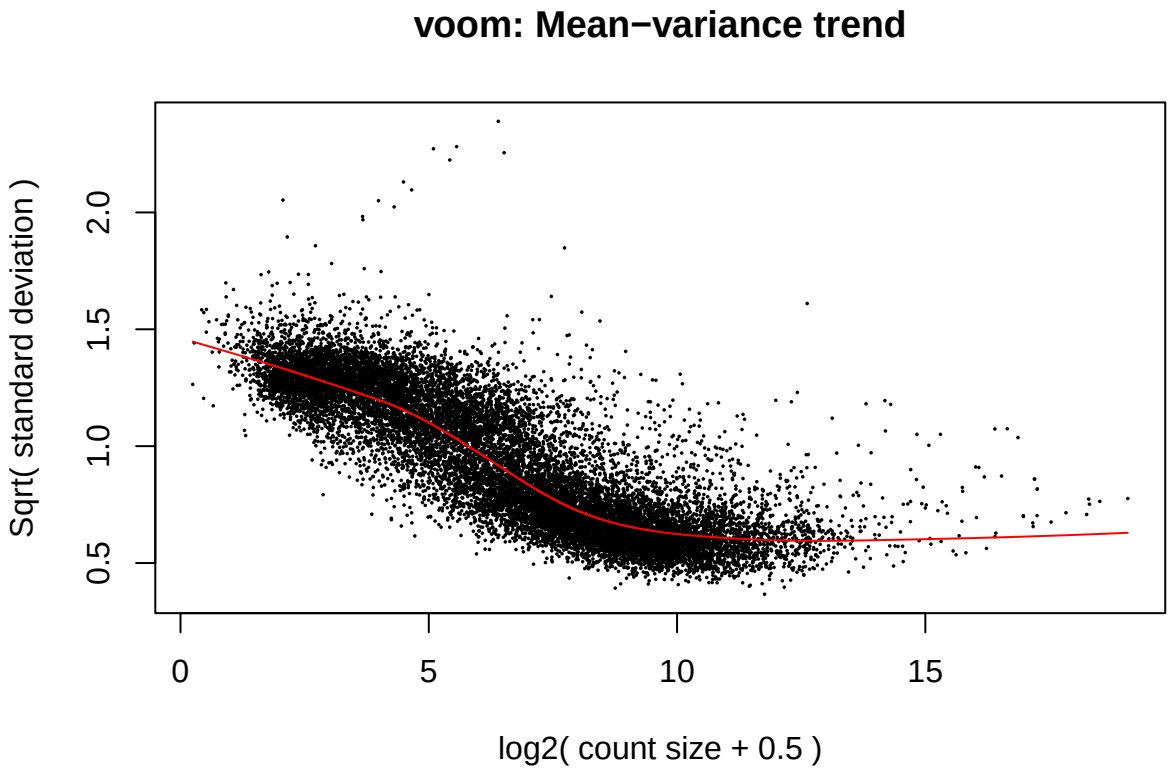
```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [31] TRUE TRUE TRUE TRUE
```

```
Group[Group=="Covid19: Acute infection"] <- "Covid19AI"
Group[Group=="Covid19: Recovery 3Mo"] <- "Covid193Mo"
```

```
Group[Group=="Covid19: Recovery 6Mo"] <- "Covid196Mo"
design <- model.matrix(~ Group )

# Filter
keep <- filterByExpr(dge, design=design)
dge <- dge[keep,,keep.lib.sizes=FALSE]

# Normalization
dge <- calcNormFactors(dge)
```

Limma: Voom or Trend?

```
## Trend

# If the sequencing depth is reasonably consistent across the RNA samples, then the simplest and most r
logCPM <- cpm(dge, log=TRUE, prior.count=3)
 # The prior count is used here to damp down the variances of logarithms of low counts.
fit <- lmFit(logCPM, design)

fit <- eBayes(fit, trend=TRUE)
 # logical, should an intensity-dependent trend be allowed for the prior variance? If FALSE then the pr
 # The use of eBayes or treat with trend=TRUE is known as the limma-trend method (Law et al, 2014; Phip

res_t = topTable(fit, coef=ncol(design), number = nrow(dge))


## Voom

# When the library sizes are quite variable between samples, then the voom approach is theoretically mo
v <- voom(dge, design, plot=TRUE)
```



or Trend-1.pdf

```
# The voom method is similar in purpose to the limma-trend method, which uses eBayes or treat with tren
fit_v <- lmFit(v, design)
fit_v <- eBayes(fit_v)
res_v = topTable(fit_v, coef=ncol(design), number = nrow(dge))
```

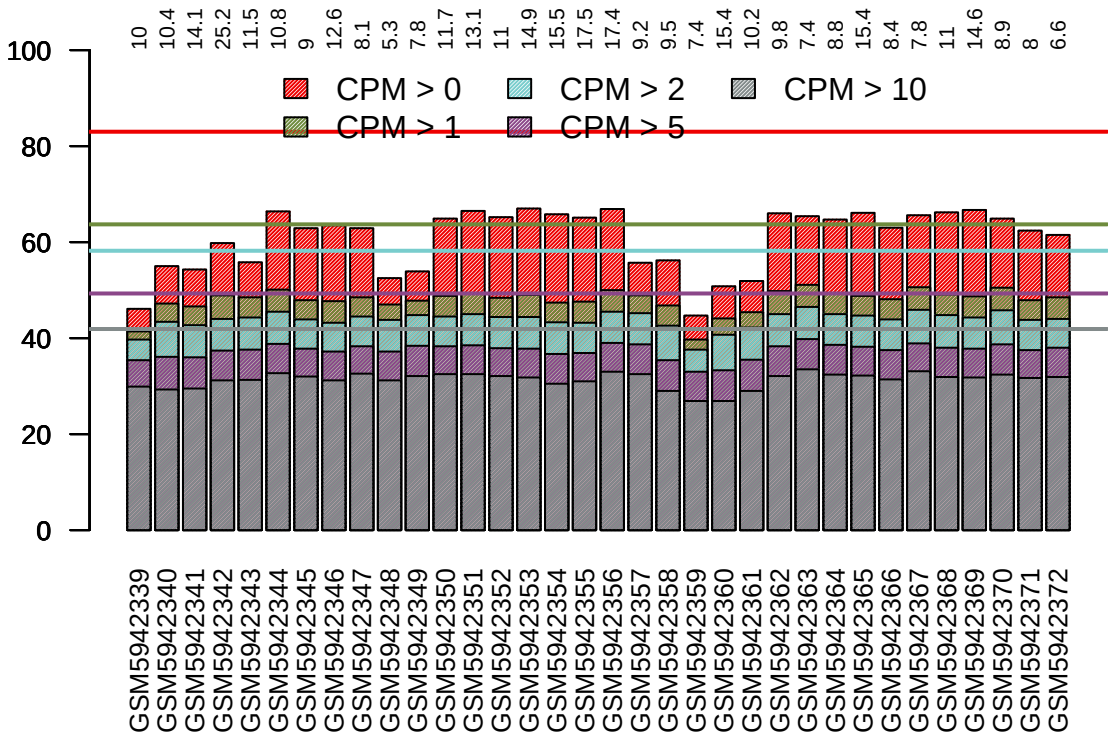
```
require(NOISeq)
```

```
## Loading required package: NOISeq
## Loading required package: splines
## Loading required package: Matrix
##
## Attaching package: 'NOISeq'
## The following object is masked from 'package:edgeR':
##
##     rpkm
```

```
load("GSE198256_step1.Rda")

mycountsbio = dat(data_NOISEQ, factor = NULL, type = "countsbio")
```

```
## [1] "Warning: 4402 features with 0 counts in all samples are to be removed for this analysis."
## [1] "Count distributions are to be computed for:"
##  [1] "GSM5942339" "GSM5942340" "GSM5942341" "GSM5942342" "GSM5942343"
##  [6] "GSM5942344" "GSM5942345" "GSM5942346" "GSM5942347" "GSM5942348"
## [11] "GSM5942349" "GSM5942350" "GSM5942351" "GSM5942352" "GSM5942353"
## [16] "GSM5942354" "GSM5942355" "GSM5942356" "GSM5942357" "GSM5942358"
## [21] "GSM5942359" "GSM5942360" "GSM5942361" "GSM5942362" "GSM5942363"
## [26] "GSM5942364" "GSM5942365" "GSM5942366" "GSM5942367" "GSM5942368"
## [31] "GSM5942369" "GSM5942370" "GSM5942371" "GSM5942372"
```

```
explo.plot(mycountsbio, toplot = 1, samples = NULL, plottype = "barplot")
```



ACTIVITY 1:

- How would you compare the results between voom and trend?

We can either compare the LFC or the Pvalues to get a sense of how DE can be enhanced under one method vs the other. In our case, voom method seems to enhance LFC values compared to trend-method. Voom seems to also exagerate PValues compared to trend.

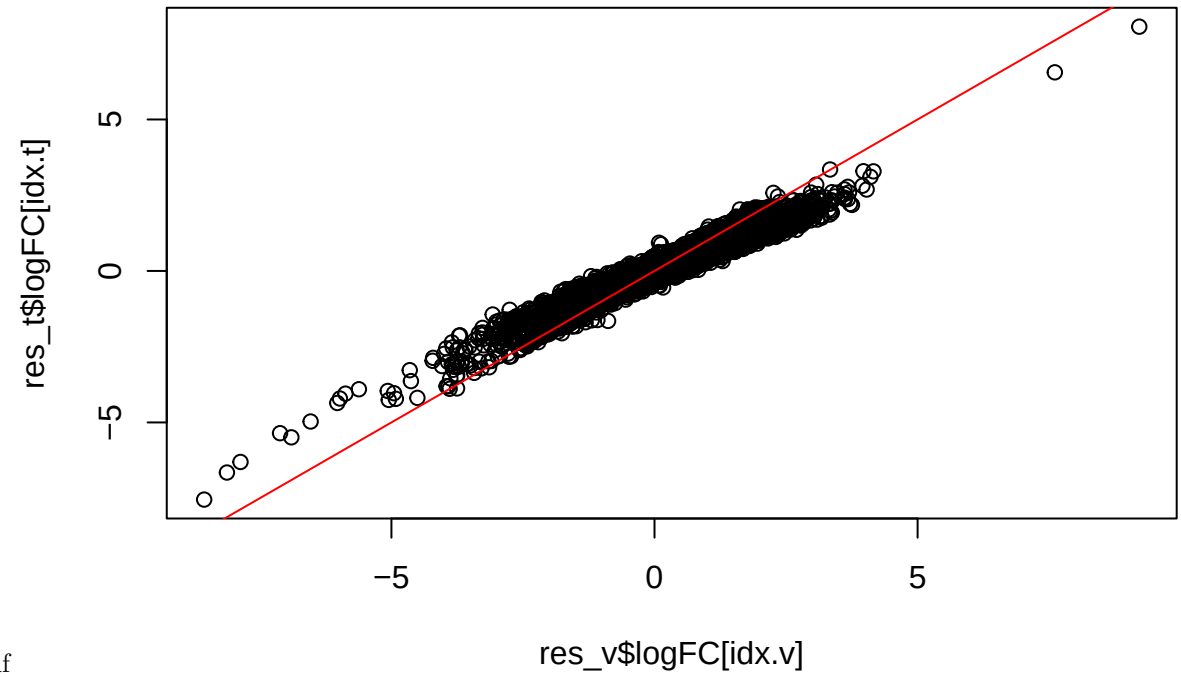
- Is it required to run more than analysis?

Depends on the question we want to ask. In this case, we are demonstrating the difference between both, so observing one case should be sufficient.

- What exactly are we asking with this differential expression?

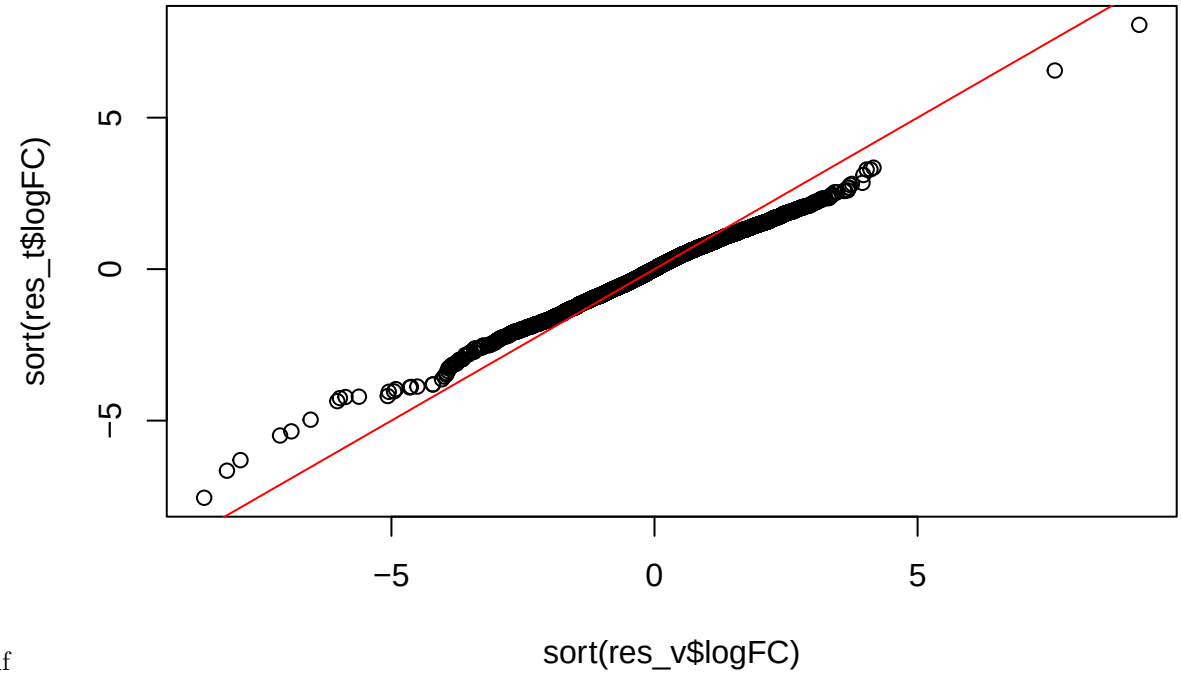
when we set a coeffecient, we ask for the differential expression for the contrast coeffecient. If the coeff is not defined, then we are asking if the feature is differentially expressed across the groups as a whole, similar to an ANOVA.

```
# Comparison 1: Are the LFCs generated different for the same genes?
idx.t = order(rownames(res_t))
idx.v = order(rownames(res_v))
plot(res_v$logFC[idx.v], res_t$logFC[idx.t])
abline(a=0, b=1, col = "red")
```



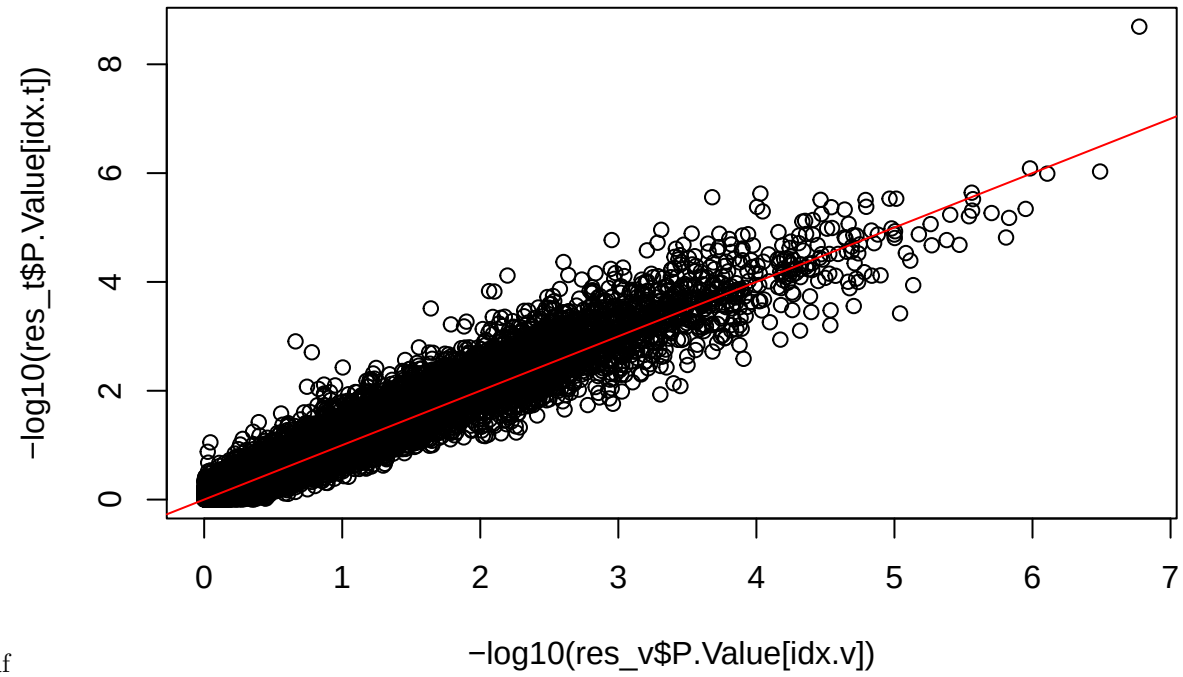
1-1.pdf

```
plot(sort(res_v$logFC), sort(res_t$logFC))
abline(a=0, b=1, col = "red")
```



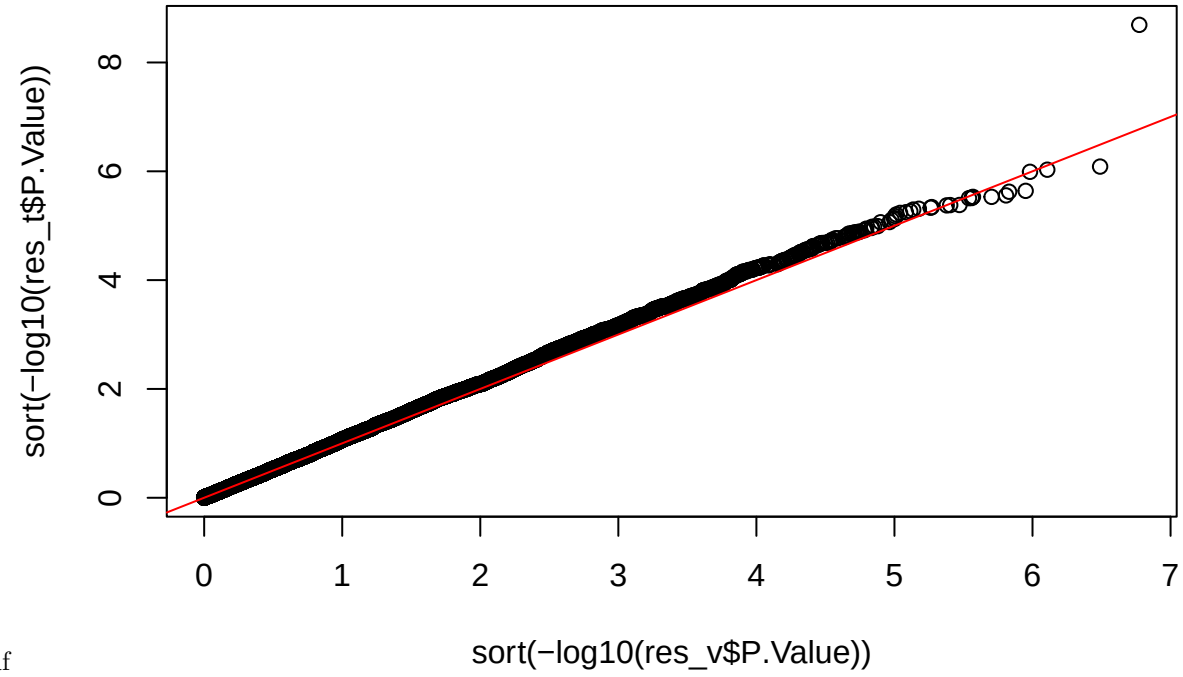
1-2.pdf

```
# Comparison 2: Are the Pvalues generated different for the same genes?
idx.t = order(rownames(res_t))
idx.v = order(rownames(res_v))
plot(-log10(res_v$P.Value[idx.v]), -log10(res_t$P.Value[idx.t]))
abline(a=0, b=1, col = "red")
```



1-3.pdf

```
plot(sort(-log10(res_v$P.Value)), sort(-log10(res_t$P.Value)))
abline(a=0, b=1, col = "red")
```



1-4.pdf

```
# plot(rank(as.numeric(rownames(res_t))), rank(as.numeric(rownames(res_v))))
# abline(a=0, b=1)
#
# plot(rank(as.numeric(rownames(res_t))), rank(as.numeric(rownames(res_v))))
#
# cor(as.numeric(rownames(res_t)),as.numeric(rownames(res_v)))
#
# wilcox.test(as.numeric(rownames(res_t)), as.numeric(rownames(res_v)))
```

```
#
# par(mfrow = c(2, 1))
# plot(res_t$logFC, -log10(res_t$P.Value))
# plot(res_v$logFC, -log10(res_v$P.Value))
```

ACTIVITY 2:

- Plan the next analysis: questions, steps,...

In the following test, we are interested in the change of monocyte expression over the process of recovery (healthy to acute to 3 month to 6 month to healthy).

```
# PACKAGES
#BiocManager::install("clusterProfiler", update = FALSE)
#BiocManager::install("ggupset", update = FALSE)
#BiocManager::install("msigdbr", update = FALSE)
#BiocManager::install("org.Hs.eg.db", update = FALSE)

library(clusterProfiler)
```

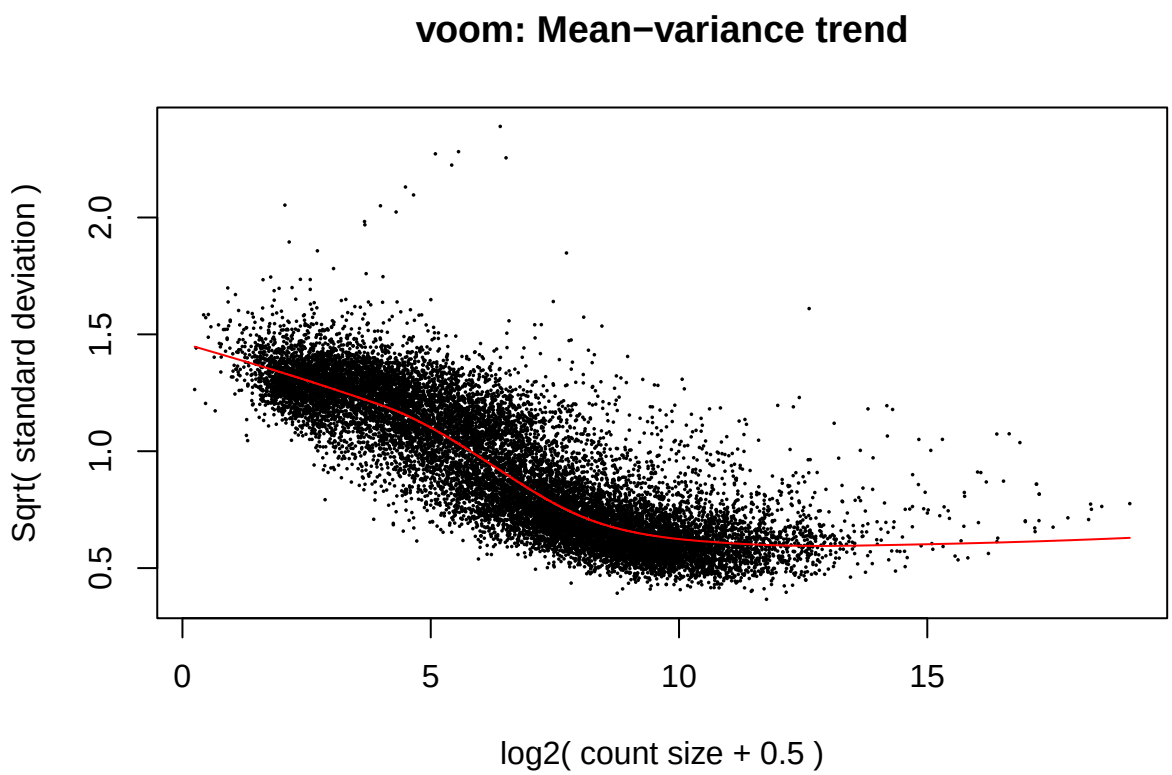
```
##
## clusterProfiler v4.10.0  For help: https://yulab-smu.top/biomedical-knowledge-mining-book/
##
## If you use clusterProfiler in published research, please cite:
## T Wu, E Hu, S Xu, M Chen, P Guo, Z Dai, T Feng, L Zhou, W Tang, L Zhan, X Fu, S Liu, X Bo, and G Yu.
##
## Attaching package: 'clusterProfiler'
## The following object is masked from 'package:stats':
##
##     filter
```

```
library(msigdbr)
library(org.Hs.eg.db)
```

```
## Loading required package: AnnotationDbi
## Loading required package: stats4
## Loading required package: IRanges
## Loading required package: S4Vectors
##
## Attaching package: 'S4Vectors'
## The following object is masked from 'package:clusterProfiler':
##
##     rename
## The following objects are masked from 'package:Matrix':
##
##     expand, unname
## The following object is masked from 'package:utils':
##
##     findMatches
```

```
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
##
## Attaching package: 'IRanges'
## The following object is masked from 'package:clusterProfiler':
##
##     slice
##
## Attaching package: 'AnnotationDbi'
## The following object is masked from 'package:clusterProfiler':
##
##     select
##
```

```r
library(magrittr)

# Add more contrasts

v <- voom(dge, design, plot=TRUE)
```



2-1.pdf

```r
colnames(design) <- c("Intercept", "Covid196Mo", "Covid19AI", "Healthy")
fit <- lmFit(v, design)

contrast.matrix <- makeContrasts(Covid19AI-Healthy,
                                 Intercept-Covid19AI,
                                 Covid196Mo-Intercept,
                                 Healthy-Covid196Mo,
```

```
                                    levels=design)
fit2 <- contrasts.fit(fit, contrast.matrix)
fit2 <- eBayes(fit2)
res_v_all = topTable(fit2, number = nrow(dge))
res_v_hta = topTable(fit2,coef=1, number = nrow(dge))
res_v_at3 = topTable(fit2,coef=2, number = nrow(dge))
res_v_3t6 = topTable(fit2,coef=3, number = nrow(dge))
res_v_6th = topTable(fit2,coef=4, number = nrow(dge))
# topTable(fit2,coef=5) #try this!

# Store all of them
# save(res_v_all, res_v_hta, res_v_at3, res_v_3t6, res_v_6th, file = "results_DEG_Limma.Rda")
load("results_DEG_Limma.Rda")
```

ORA and Gene Set Enrichment analysis.

- What do we need to do the analysis? GO references for our system of interest, in our case, that would be human.
- What are the tools required?

```
keytypes(org.Hs.eg.db)
```

```
##  [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
##  [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
## [11] "GENETYPE"     "GO"           "GOALL"        "IPI"          "MAP"
## [16] "OMIM"         "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
## [21] "PMID"         "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"
## [26] "UNIPROT"
```

```
# If we want to shift annotations:
ENSEMBL_vector <- mapIds(
  # Replace with annotation package for the organism relevant to your data
  org.Hs.eg.db,
  # The vector of gene identifiers we want to map
  keys = rownames(GSE198256_count),
  # Replace with the type of gene identifiers in your data
  keytype = "ENTREZID",
  # Replace with the type of gene identifiers you would like to map to
  column = "ENSEMBL",
  # In the case of 1:many mappings, return the
  # first one. This is default behavior!
  multiVals = "first"
)
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
table(is.na(ENSEMBL_vector))
```

```
##
## FALSE  TRUE
## 27291 12085
```

```
table(duplicated(ENSEMBL_vector))
```

```
##
```

```
## FALSE  TRUE
## 27155 12221
```

```
length(ENSEMBL_vector)
```

```
## [1] 39376
```

```
# We would like a data frame we can join to the differential expression stats
gene_key_df <- data.frame(
  ensembl_id = ENSEMBL_vector,
  entrez_id = names(ENSEMBL_vector),
  stringsAsFactors = FALSE
) %>%
  # If an Ensembl gene identifier doesn't map to a gene symbol, drop that
  # from the data frame
  dplyr::filter(!is.na(ensembl_id))
```

Lets conduct ORA.

- What do we need to do the analysis?
- What are the tools required?

```
# Step 1: determine genes of interest.
diff_table <- topTable(fit2,coef=1,p.value=0.01,number=10000)
genes_dif<- rownames(diff_table )

# Step 2: determine background.

background_set <- unique(rownames(logCPM))

# Step 3: Determine gene sets.

msigdbr_species()
```

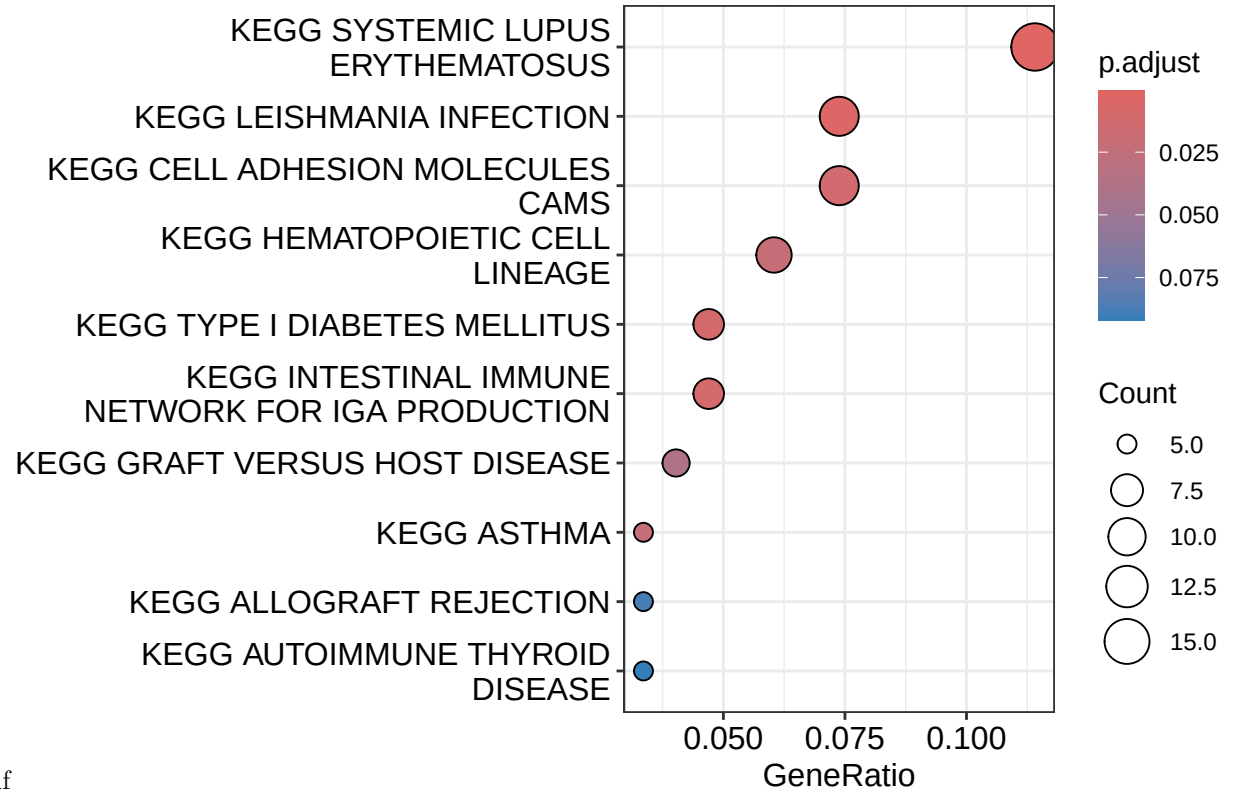
```
## # A tibble: 20 x 2
##    species_name                  species_common_name
##    <chr>                         <chr>
##  1 Anolis carolinensis           Carolina anole, green anole
##  2 Bos taurus                    bovine, cattle, cow, dairy cow, domestic cat~
##  3 Caenorhabditis elegans        <NA>
##  4 Canis lupus familiaris        dog, dogs
##  5 Danio rerio                   leopard danio, zebra danio, zebra fish, zebr~
##  6 Drosophila melanogaster       fruit fly
##  7 Equus caballus                domestic horse, equine, horse
##  8 Felis catus                   cat, cats, domestic cat
##  9 Gallus gallus                 bantam, chicken, chickens, Gallus domesticus
## 10 Homo sapiens                  human
## 11 Macaca mulatta                rhesus macaque, rhesus macaques, Rhesus monk~
## 12 Monodelphis domestica         gray short-tailed opossum
## 13 Mus musculus                  house mouse, mouse
## 14 Ornithorhynchus anatinus      duck-billed platypus, duckbill platypus, pla~
## 15 Pan troglodytes               chimpanzee
## 16 Rattus norvegicus             brown rat, Norway rat, rat, rats
## 17 Saccharomyces cerevisiae      baker's yeast, brewer's yeast, S. cerevisiae
## 18 Schizosaccharomyces pombe 972h- <NA>
```

```
## 19 Sus scrofa                      pig, pigs, swine, wild boar
## 20 Xenopus tropicalis              tropical clawed frog, western clawed frog
```

```
hs_msigdb_df <- msigdbr(species = "Homo sapiens")
head(hs_msigdb_df)
```

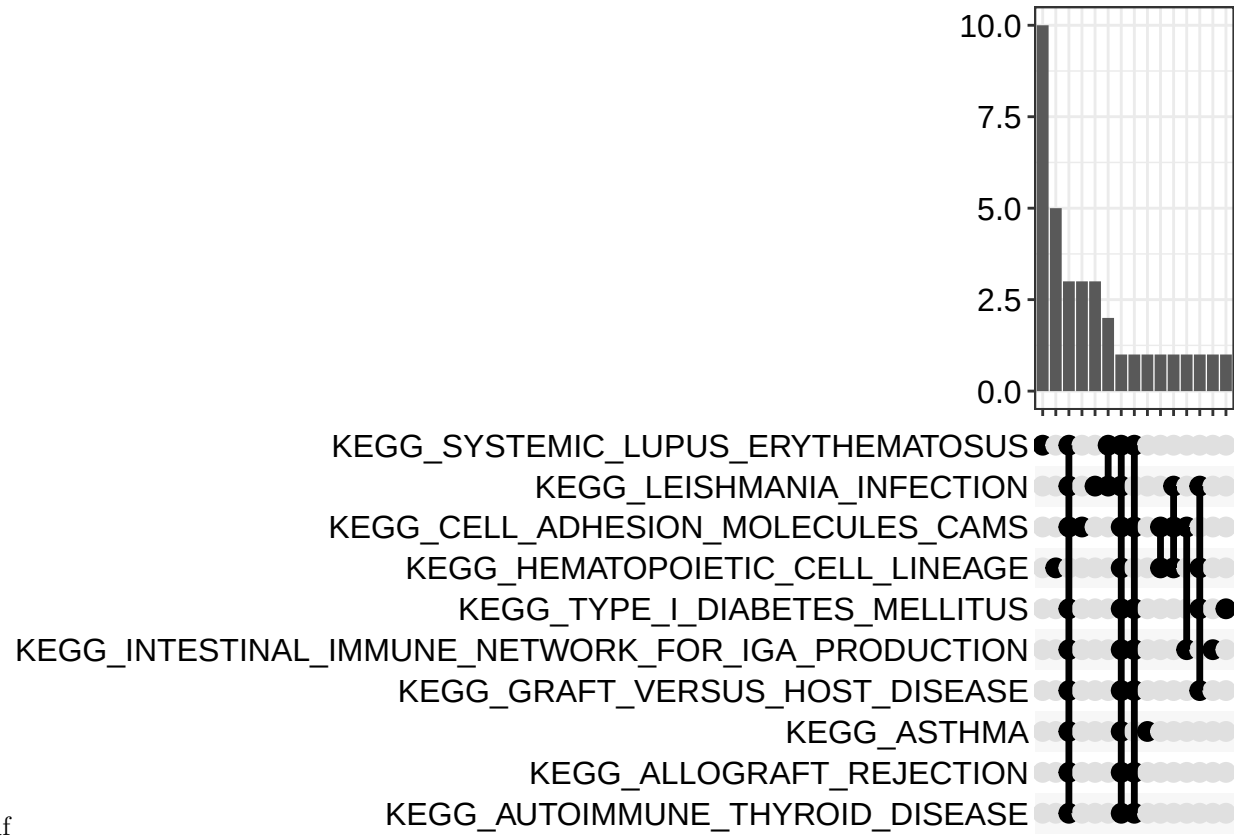
```
## # A tibble: 6 x 15
##   gs_cat gs_subcat      gs_name        gene_symbol entrez_gene ensembl_gene
##   <chr>  <chr>          <chr>          <chr>              <int> <chr>
## 1 C3     MIR:MIR_Legacy AAACCAC_MIR140 ABCC4              10257 ENSG00000125257
## 2 C3     MIR:MIR_Legacy AAACCAC_MIR140 ABRAXAS2           23172 ENSG00000165660
## 3 C3     MIR:MIR_Legacy AAACCAC_MIR140 ACTN4                 81 ENSG00000130402
## 4 C3     MIR:MIR_Legacy AAACCAC_MIR140 ACTN4                 81 ENSG00000282844
## 5 C3     MIR:MIR_Legacy AAACCAC_MIR140 ACVR1                 90 ENSG00000115170
## 6 C3     MIR:MIR_Legacy AAACCAC_MIR140 ADAM9               8754 ENSG00000168615
## # i 9 more variables: human_gene_symbol <chr>, human_entrez_gene <int>,
## #   human_ensembl_gene <chr>, gs_id <chr>, gs_pmid <chr>, gs_geoid <chr>,
## #   gs_exact_source <chr>, gs_url <chr>, gs_description <chr>
```

```
hs_kegg_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C2", # This is to filter only to the C2 curated gene sets
    gs_subcat == "CP:KEGG" # This is because we only want KEGG pathways
  )

# Step 4: conduct ORA.

kegg_ora_results <- enricher(
  gene = genes_dif, # A vector of your genes of interest
  pvalueCutoff = 0.1, # Can choose a FDR cutoff
  pAdjustMethod = "BH", # Method to be used for multiple testing correction
  universe = background_set, # A vector containing your background set genes
  # The pathway information should be a data frame with a term name or
  # identifier and the gene identifiers
  TERM2GENE = dplyr::select(
    hs_kegg_df,
    gs_name,
    human_entrez_gene
  )
)


# Step 5: Visualize / explore

enrich_plot <- enrichplot::dotplot(kegg_ora_results)
enrich_plot
```



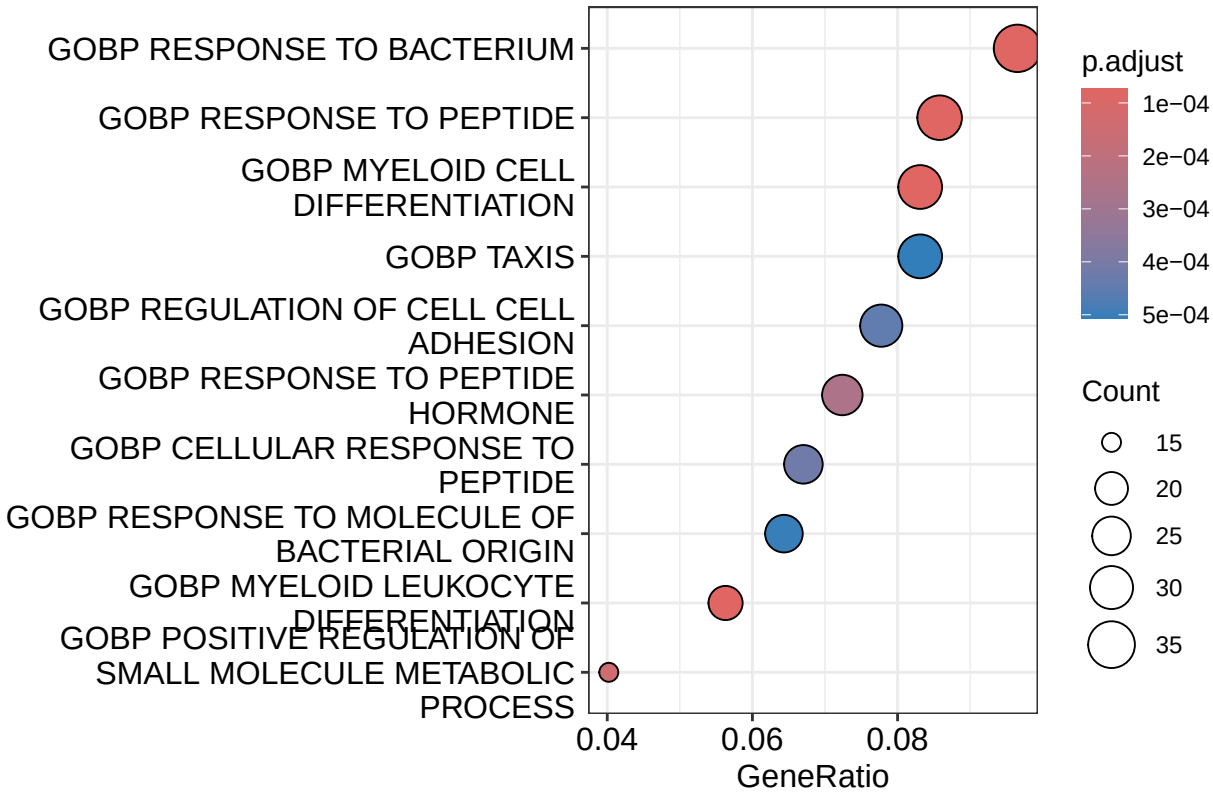
ORA-1.pdf

```
upset_plot <- enrichplot::upsetplot(kegg_ora_results)
upset_plot
```



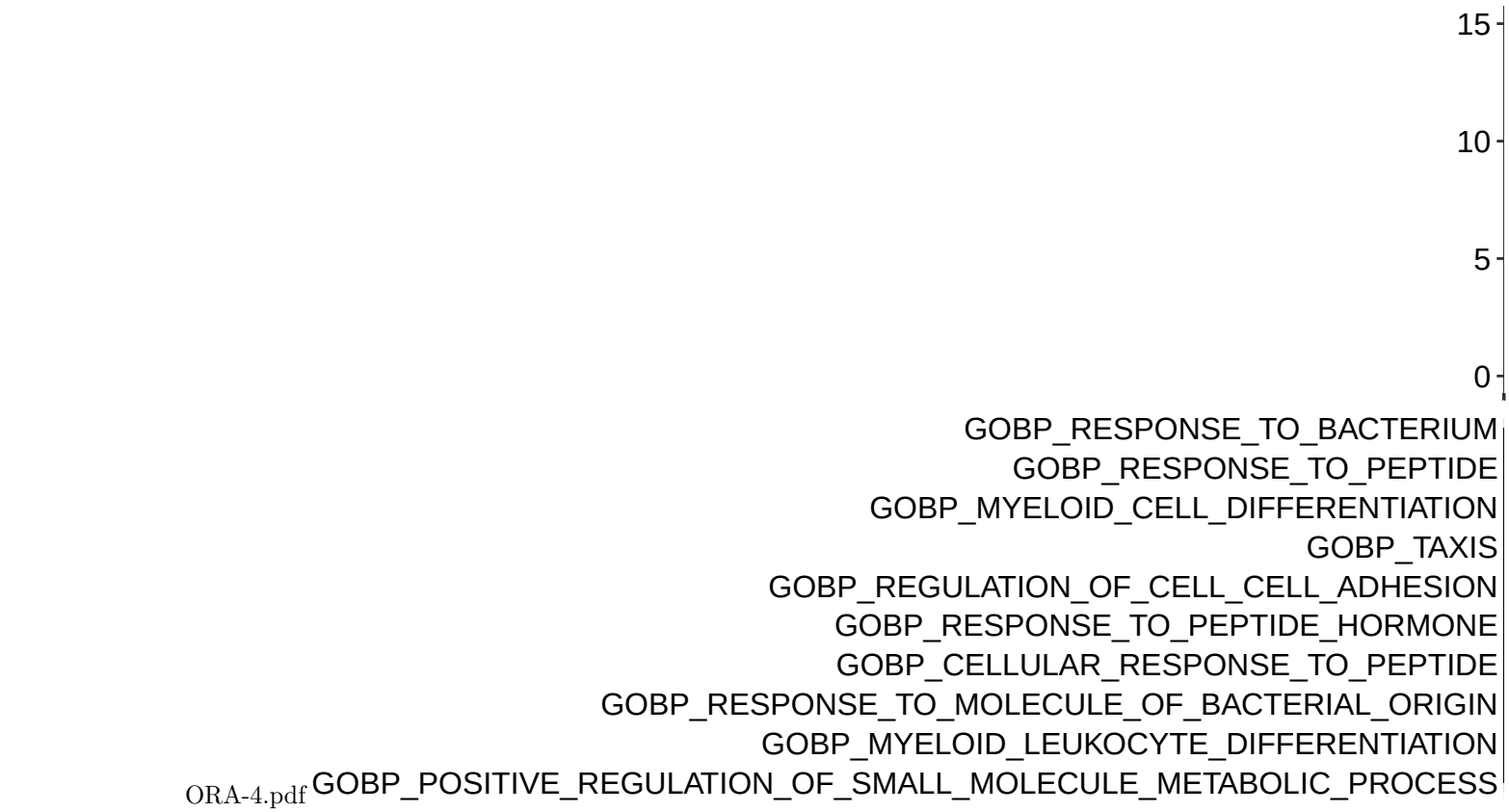
ORA-2.pdf

```r
# Step 6: EXERCISE: alternatives to KEGG?
hs_BP_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C5", # This is to filter only to the C2 curated gene sets
    gs_subcat == "GO:BP" # This is because we only want KEGG pathways
  )

GOBP_ora_results <- enricher(
  gene = genes_dif, # A vector of your genes of interest
  pvalueCutoff = 0.1, # Can choose a FDR cutoff
  pAdjustMethod = "BH", # Method to be used for multiple testing correction
  universe = background_set, # A vector containing your background set genes
  # The pathway information should be a data frame with a term name or
  # identifier and the gene identifiers
  TERM2GENE = dplyr::select(
    hs_BP_df,
    gs_name,
    human_entrez_gene
  )
)

enrich_plot <- enrichplot::dotplot(GOBP_ora_results)
enrich_plot
```



ORA-3.pdf

```r
upset_plot <- enrichplot::upsetplot(GOBP_ora_results)
upset_plot
```



ORA-4.pdf

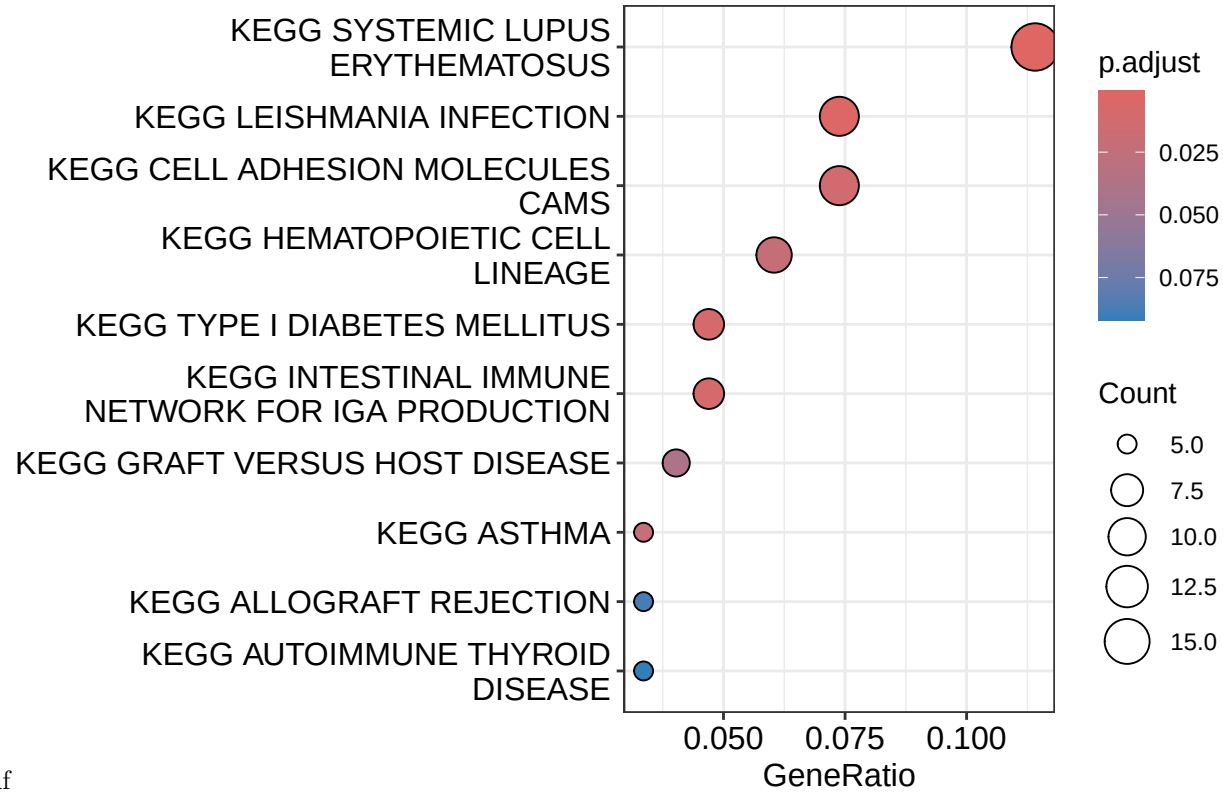
```
# note: see mor examples: https://alexslemonade.github.io/refinebio-examples/03-rnaseq/pathway-analysis_
```

```
res_v_hta = topTable(fit2,coef=1, p.value=0.01, number = nrow(dge))
res_v_at3 = topTable(fit2,coef=2, p.value=0.01, number = nrow(dge))
res_v_3t6 = topTable(fit2,coef=3, p.value=0.01, number = nrow(dge))
res_v_6th = topTable(fit2,coef=4, p.value=0.01, number = nrow(dge))

res_list = list(
                res_v_hta,
                res_v_at3,
                res_v_3t6)
```

```
## Check KEGG pathways for all transitions

GET_KEGG_ORA_RESULTS = function(diff_table){

  genes_dif<- rownames(diff_table)

  kegg_ora_results <- enricher(
  gene = genes_dif, # A vector of your genes of interest
  pvalueCutoff = 0.1, # Can choose a FDR cutoff
  pAdjustMethod = "BH", # Method to be used for multiple testing correction
  universe = background_set, # A vector containing your background set genes
  # The pathway information should be a data frame with a term name or
  # identifier and the gene identifiers
  TERM2GENE = dplyr::select(
    hs_kegg_df,
    gs_name,
```

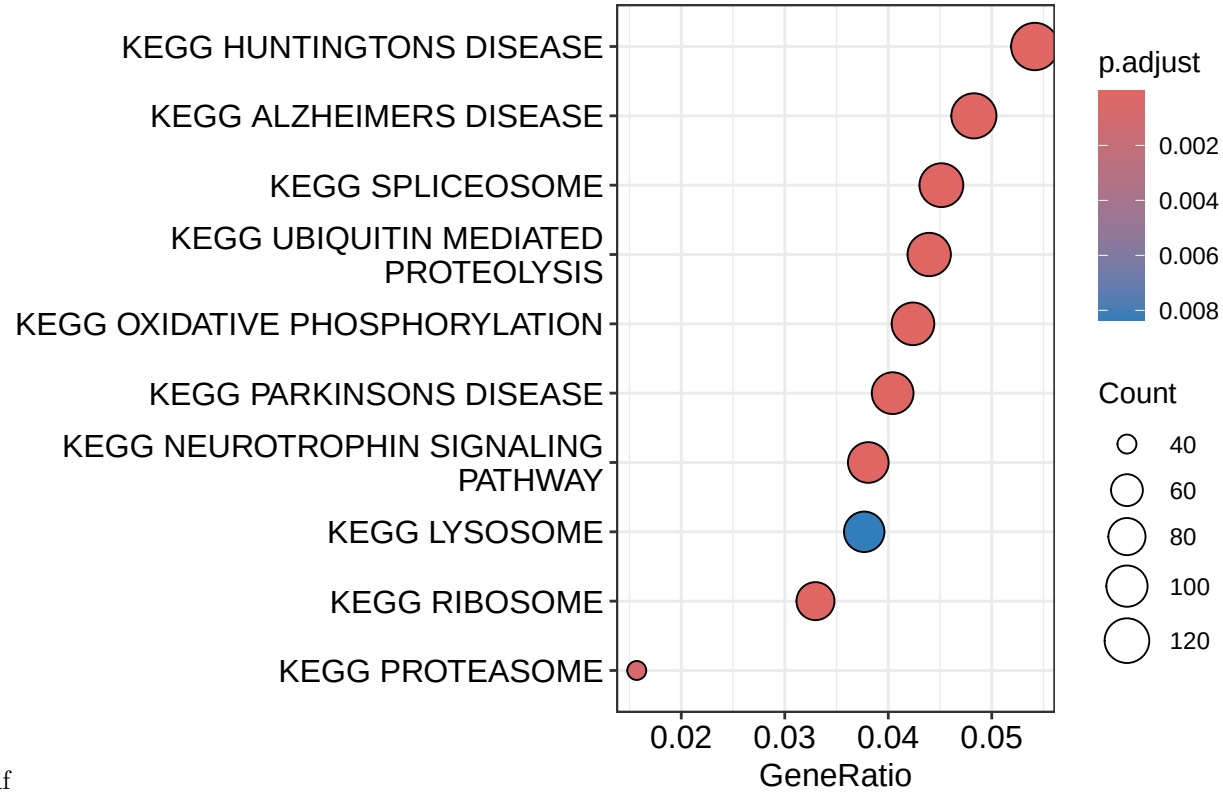
```
    human_entrez_gene
  )
)
  return(kegg_ora_results)
}
GET_GO_BP_ORA_RESULTS = function(diff_table){

  genes_dif<- rownames(diff_table)

  bp_ora_results <- enricher(
  gene = genes_dif, # A vector of your genes of interest
  pvalueCutoff = 0.1, # Can choose a FDR cutoff
  pAdjustMethod = "BH", # Method to be used for multiple testing correction
  universe = background_set, # A vector containing your background set genes
  # The pathway information should be a data frame with a term name or
  # identifier and the gene identifiers
  TERM2GENE = dplyr::select(
    hs_BP_df,
    gs_name,
    human_entrez_gene
  )
)
  return(bp_ora_results)
}

# Step 5: Visualize / explore
kegg_ora_results.ls = lapply(res_list, GET_KEGG_ORA_RESULTS)
enrich_plot.ls = lapply(kegg_ora_results.ls, function(x) enrichplot::dotplot(x))
enrich_plot.ls
```

```
## [[1]]
```



GSEA 1-1.pdf

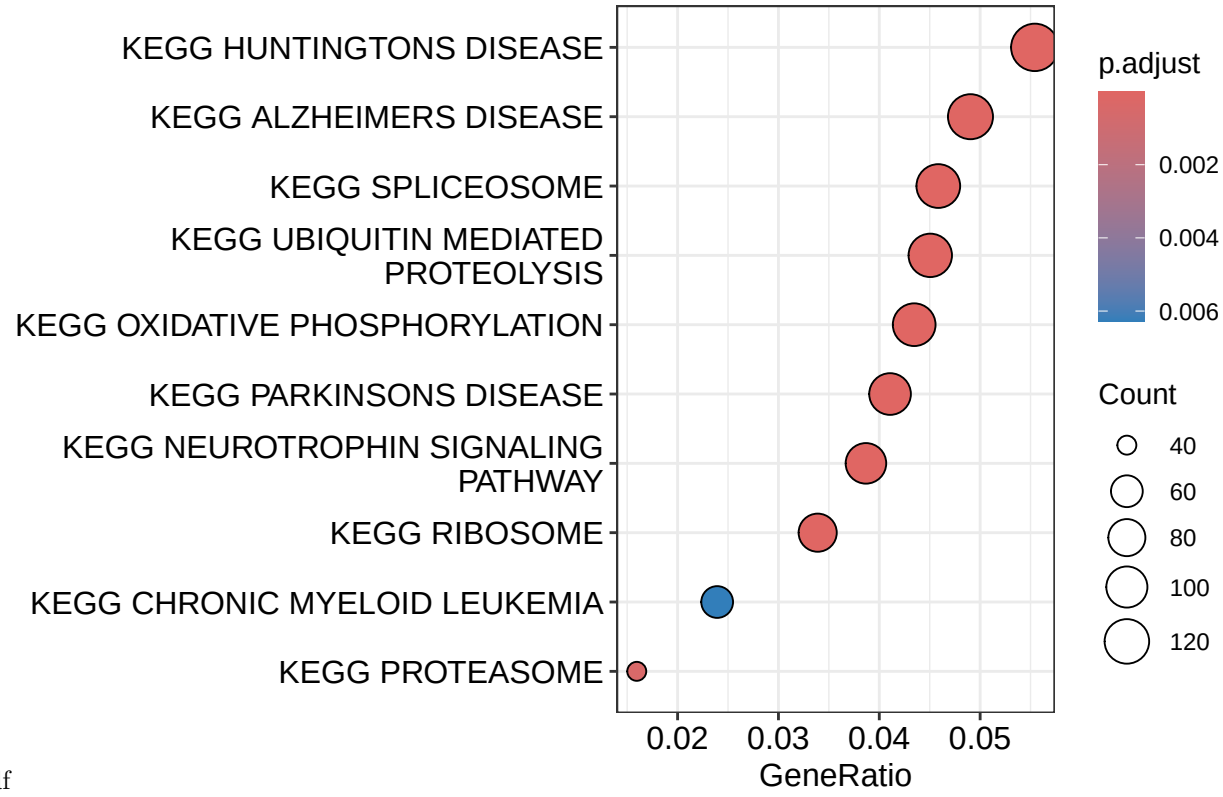
##
[[2]]



GSEA 1-2.pdf

##

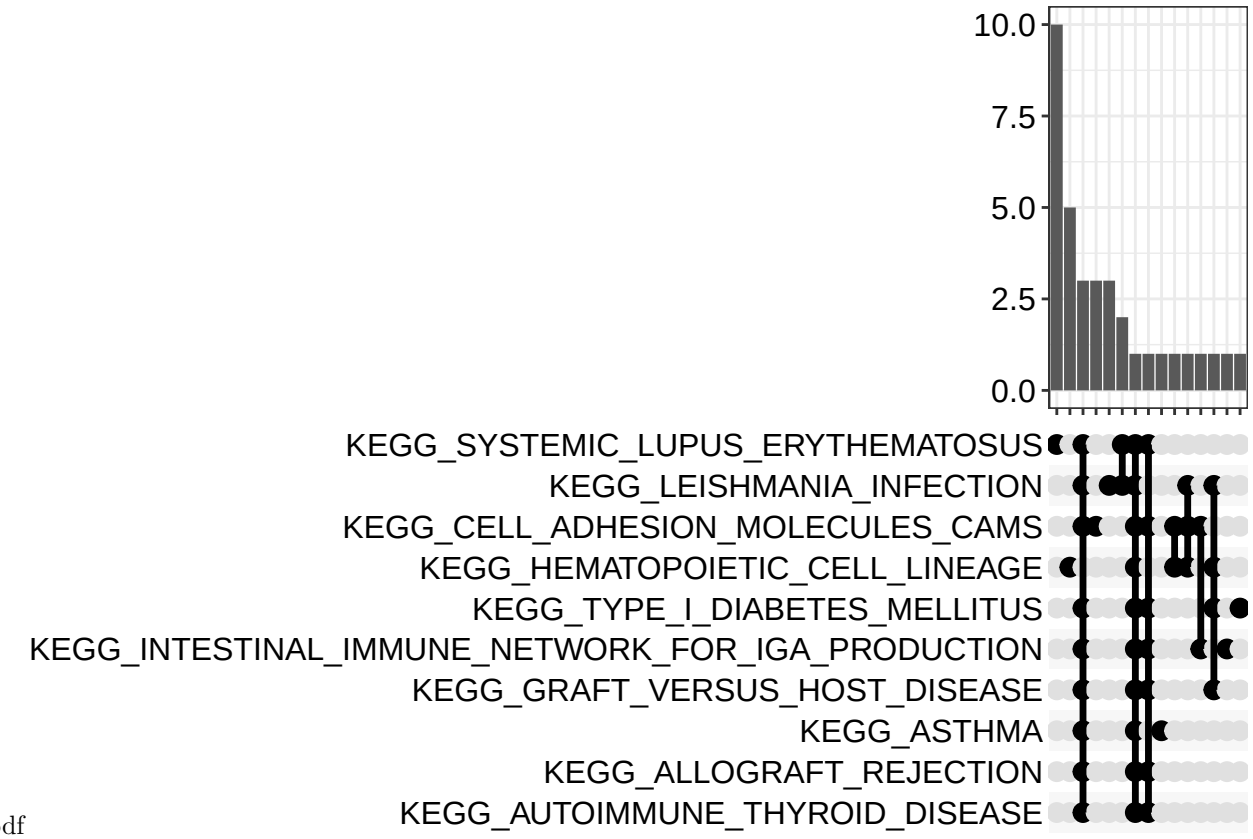
```
## [[3]]
```



GSEA 1-3.pdf

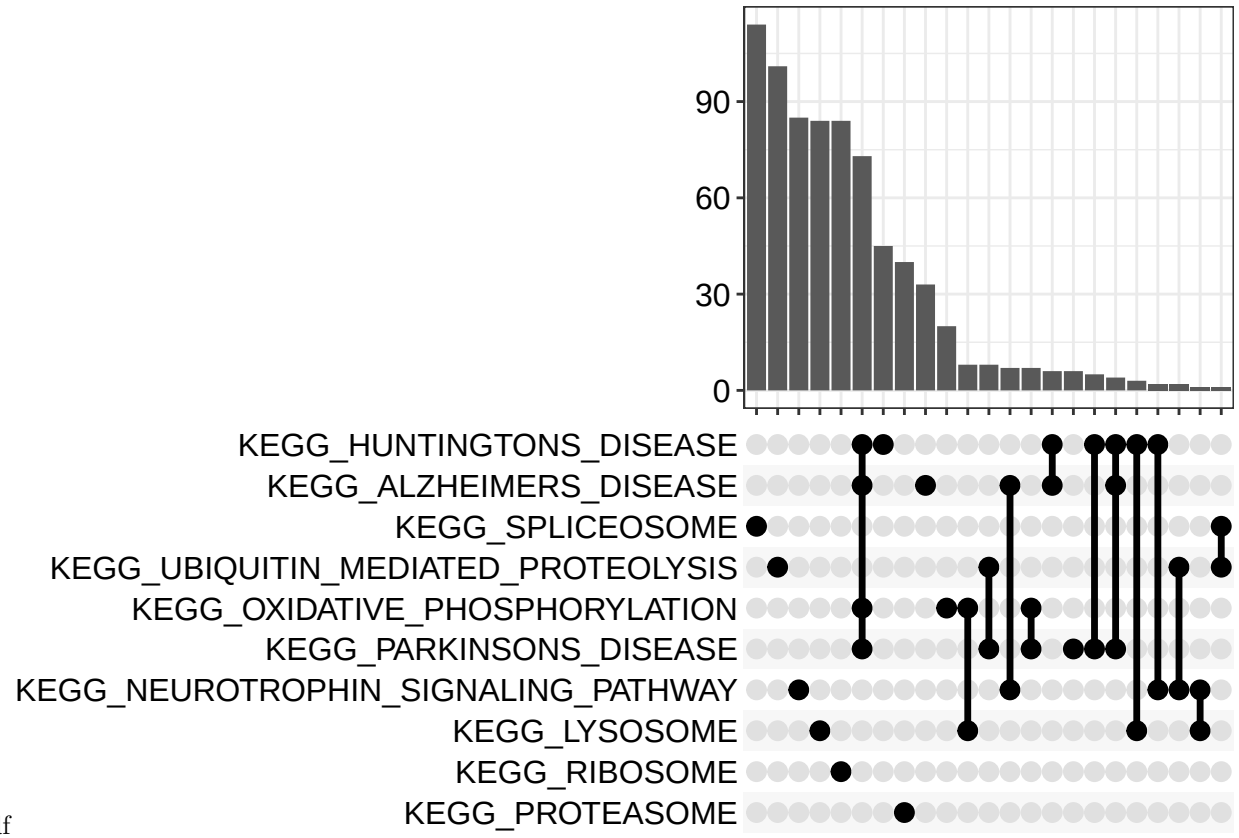
```
upset_plot.ls = lapply(kegg_ora_results.ls, function(x) enrichplot::upsetplot(x))
upset_plot.ls
```

```
## [[1]]
```


10.0
7.5
5.0
2.5
0.0
KEGG_SYSTEMIC_LUPUS_ERYTHEMATOSUS
KEGG_LEISHMANIA_INFECTION
KEGG_CELL_ADHESION_MOLECULES_CAMS
KEGG_HEMATOPOIETIC_CELL_LINEAGE
KEGG_TYPE_I_DIABETES_MELLITUS
KEGG_INTESTINAL_IMMUNE_NETWORK_FOR_IGA_PRODUCTION
KEGG_GRAFT_VERSUS_HOST_DISEASE
KEGG_ASTHMA
KEGG_ALLOGRAFT_REJECTION
KEGG_AUTOIMMUNE_THYROID_DISEASE

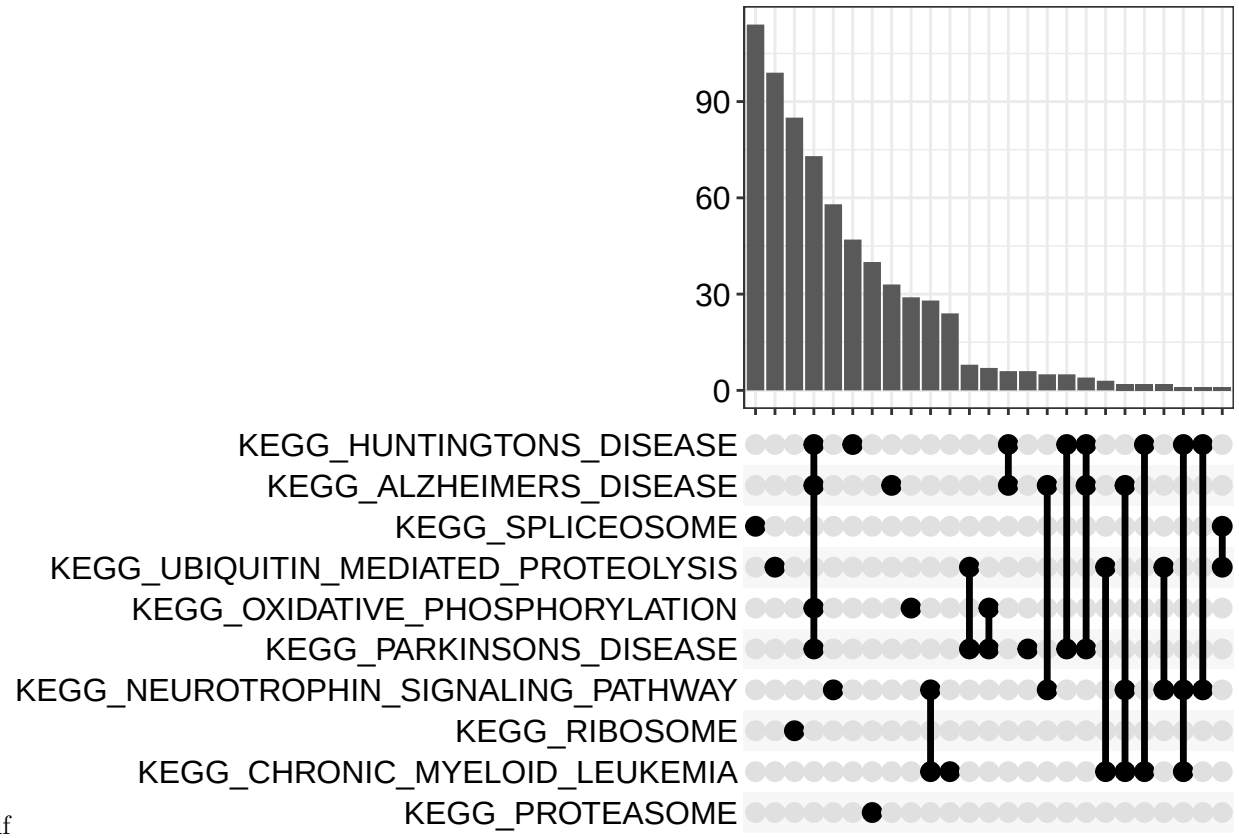
GSEA 1-4.pdf

```
##
## [[2]]
```


90
60
30
0
KEGG_HUNTINGTONS_DISEASE
KEGG_ALZHEIMERS_DISEASE
KEGG_SPLICEOSOME
KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS
KEGG_OXIDATIVE_PHOSPHORYLATION
KEGG_PARKINSONS_DISEASE
KEGG_NEUROTROPHIN_SIGNALING_PATHWAY
KEGG_LYSOSOME
KEGG_RIBOSOME
KEGG_PROTEASOME

GSEA 1-5.pdf

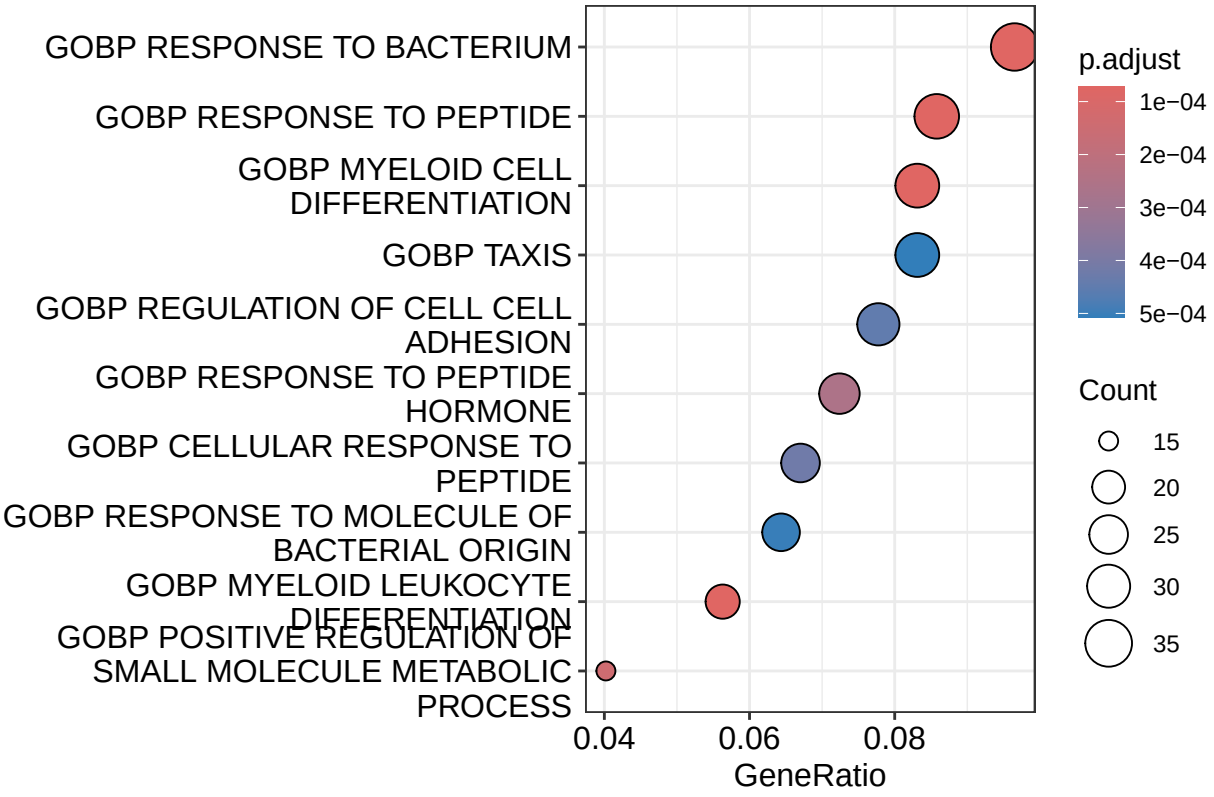
```
##
## [[3]]
```



GSEA 1-6.pdf

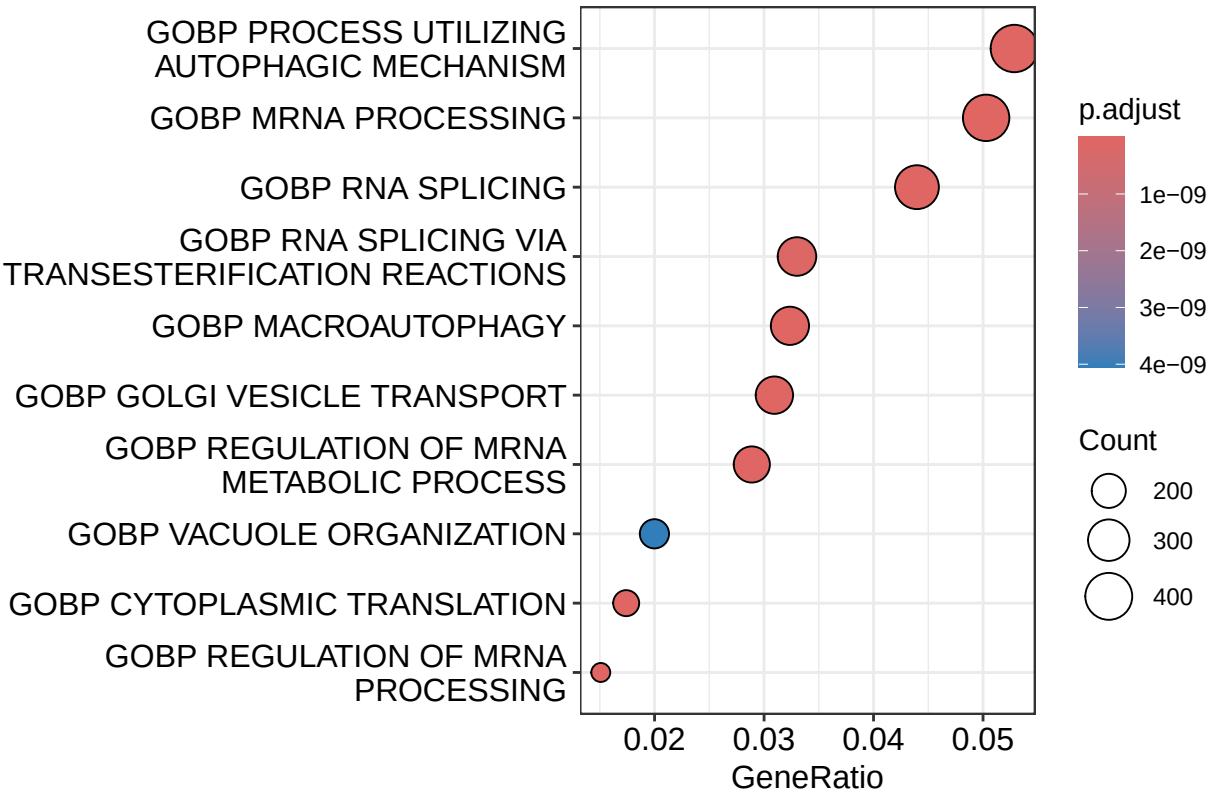
```
# Step 5: Visualize / explore
go_bp_ora_results.ls = lapply(res_list, GET_GO_BP_ORA_RESULTS)
enrich_plot.ls = lapply(go_bp_ora_results.ls, function(x) enrichplot::dotplot(x))
enrich_plot.ls
```

```
## [[1]]
```



GSEA 1-7.pdf

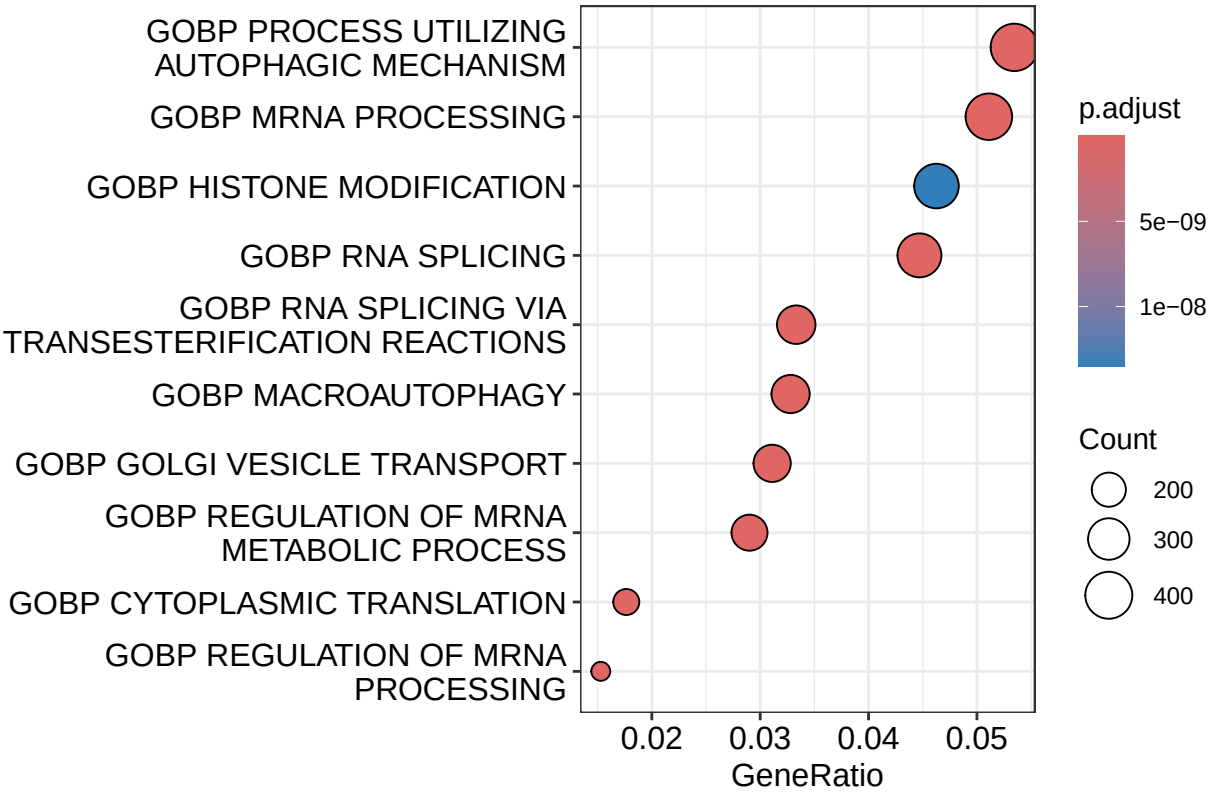
```
##
## [[2]]
```



GSEA 1-8.pdf

```
##
```

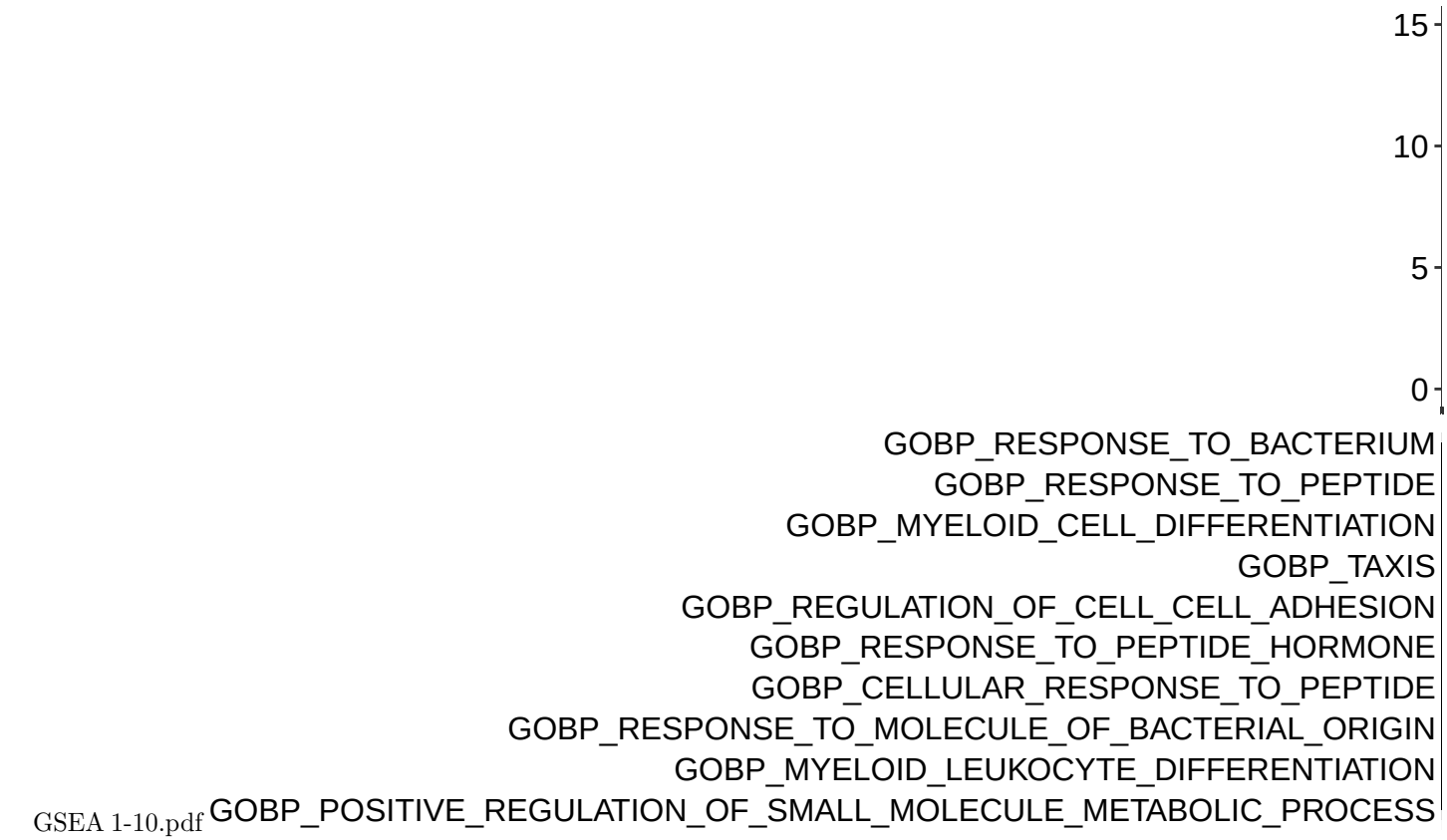
```
## [[3]]
```



GSEA 1-9.pdf

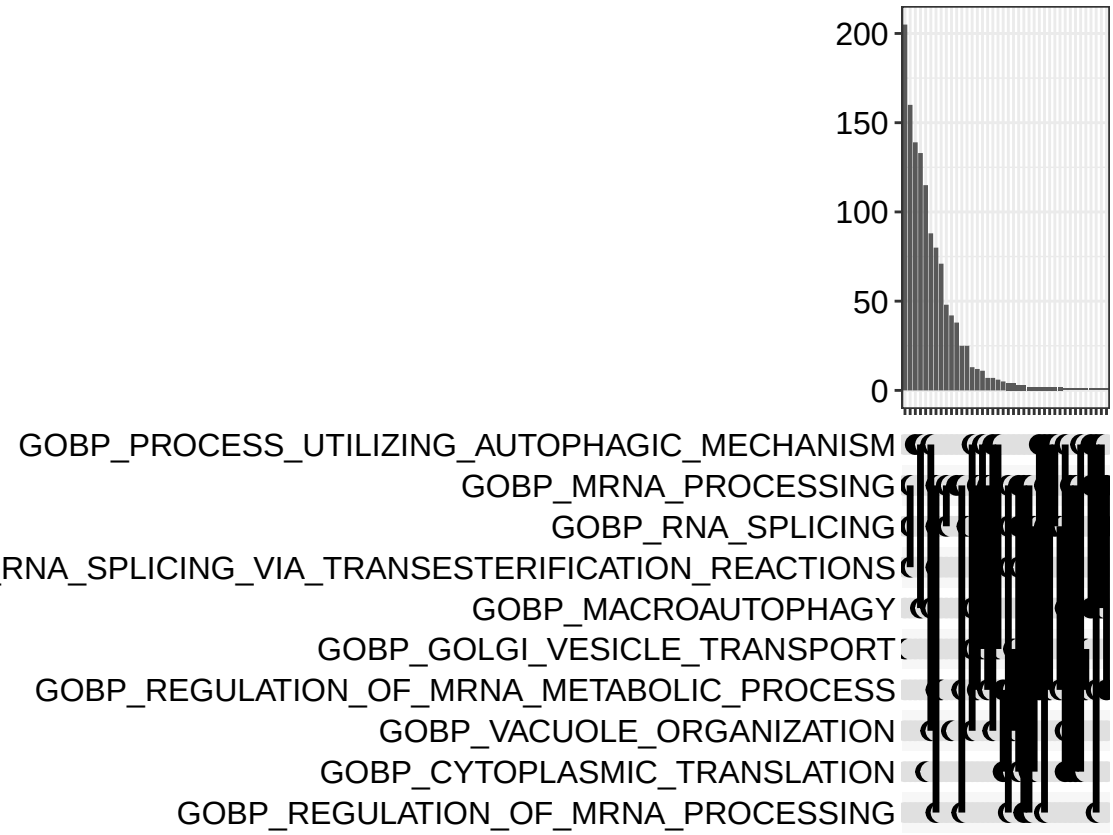
```
upset_plot.ls = lapply(go_bp_ora_results.ls, function(x) enrichplot::upsetplot(x))
upset_plot.ls
```

```
## [[1]]
```



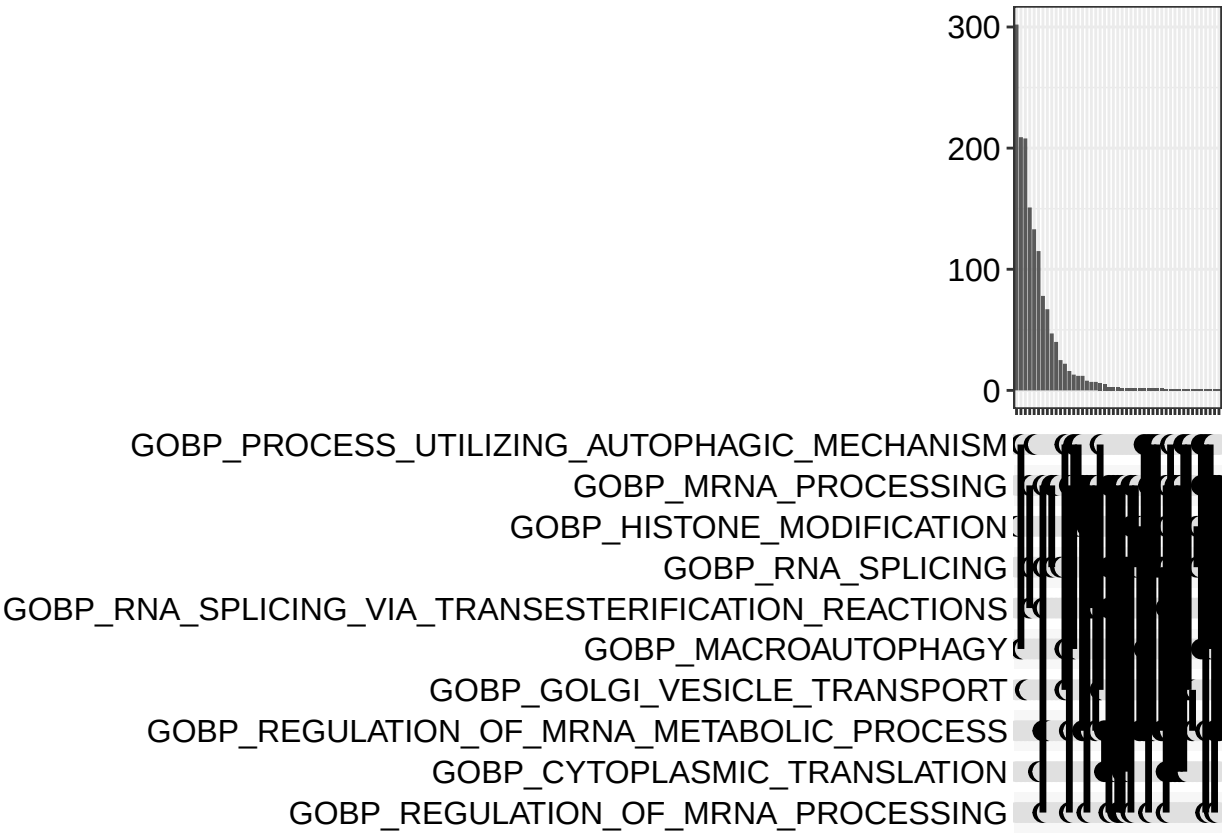
GSEA 1-10.pdf

```
##
## [[2]]
```



GSEA 1-11.pdf

```
##
## [[3]]
```



GSEA 1-12.pdf

```
# healthy to acute, acute to 3 months, 3 months to 6 months
```

Lets conduct GSEA.

```
res_v_hta = topTable(fit2,coef=1, p.value=1, number = nrow(dge))
res_v_at3 = topTable(fit2,coef=2, p.value=1, number = nrow(dge))
res_v_3t6 = topTable(fit2,coef=3, p.value=1, number = nrow(dge))
res_v_6th = topTable(fit2,coef=4, p.value=1, number = nrow(dge))

res_2_list = list(
                res_v_hta,
                res_v_at3,
                res_v_3t6,
                res_v_6th)
```

```
# Step 1: determine genes of interest.
diff_table_all <- topTable(fit2,coef=1,p.value=1,number=nrow(logCPM))

# we want all the genes for our analysis

# Step 2: determine background.

# we don't have to, because all the genes are evaluated.

# Step 3: Determine gene sets.
```

```
msigdbr_species()
```

```
## # A tibble: 20 x 2
##    species_name                     species_common_name
##    <chr>                            <chr>
##  1 Anolis carolinensis              Carolina anole, green anole
##  2 Bos taurus                       bovine, cattle, cow, dairy cow, domestic cat~
##  3 Caenorhabditis elegans           <NA>
##  4 Canis lupus familiaris           dog, dogs
##  5 Danio rerio                      leopard danio, zebra danio, zebra fish, zebr~
##  6 Drosophila melanogaster          fruit fly
##  7 Equus caballus                   domestic horse, equine, horse
##  8 Felis catus                      cat, cats, domestic cat
##  9 Gallus gallus                    bantam, chicken, chickens, Gallus domesticus
## 10 Homo sapiens                     human
## 11 Macaca mulatta                   rhesus macaque, rhesus macaques, Rhesus monk~
## 12 Monodelphis domestica            gray short-tailed opossum
## 13 Mus musculus                     house mouse, mouse
## 14 Ornithorhynchus anatinus         duck-billed platypus, duckbill platypus, pla~
## 15 Pan troglodytes                  chimpanzee
## 16 Rattus norvegicus                brown rat, Norway rat, rat, rats
## 17 Saccharomyces cerevisiae         baker's yeast, brewer's yeast, S. cerevisiae
## 18 Schizosaccharomyces pombe 972h-  <NA>
## 19 Sus scrofa                       pig, pigs, swine, wild boar
## 20 Xenopus tropicalis               tropical clawed frog, western clawed frog
```

```
hs_msigdb_df <- msigdbr(species = "Homo sapiens")
head(hs_msigdb_df)
```

```
## # A tibble: 6 x 15
##   gs_cat gs_subcat      gs_name        gene_symbol entrez_gene ensembl_gene
##   <chr>  <chr>          <chr>          <chr>             <int> <chr>
## 1 C3     MIR:MIR_Legacy AAACCAC_MIR140 ABCC4             10257 ENSG00000125257
## 2 C3     MIR:MIR_Legacy AAACCAC_MIR140 ABRAXAS2          23172 ENSG00000165660
## 3 C3     MIR:MIR_Legacy AAACCAC_MIR140 ACTN4                81 ENSG00000130402
## 4 C3     MIR:MIR_Legacy AAACCAC_MIR140 ACTN4                81 ENSG00000282844
## 5 C3     MIR:MIR_Legacy AAACCAC_MIR140 ACVR1                90 ENSG00000115170
## 6 C3     MIR:MIR_Legacy AAACCAC_MIR140 ADAM9              8754 ENSG00000168615
## # i 9 more variables: human_gene_symbol <chr>, human_entrez_gene <int>,
## #   human_ensembl_gene <chr>, gs_id <chr>, gs_pmid <chr>, gs_geoid <chr>,
## #   gs_exact_source <chr>, gs_url <chr>, gs_description <chr>
```

```
hs_kegg_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C2", # This is to filter only to the C2 curated gene sets
    gs_subcat == "CP:KEGG" # This is because we only want KEGG pathways
  )

# Step 4: conduct GSEA
# function(diff_table_all){
  list_ordered <- diff_table_all[,"B"]
  names(list_ordered) <- rownames(diff_table_all)


  gsea_results <- GSEA(
```

```
    geneList = list_ordered, # Ordered ranked gene list
    minGSSize = 25, # Minimum gene set size
    maxGSSize = 500, # Maximum gene set set
    pvalueCutoff = 0.05, # p-value cutoff
    eps = 0, # Boundary for calculating the p value
    seed = FALSE, # Set seed to make results reproducible
    pAdjustMethod = "BH", # Benjamini-Hochberg correction
    TERM2GENE = dplyr::select(
      hs_kegg_df,
      gs_name,
      human_entrez_gene
    )
  )
```

```
## preparing geneSet collections...
```

```
## GSEA analysis...
```

```
## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize, gseaParam, : There are ties in
## The order of those tied genes will be arbitrary, which may produce unexpected results.
```

```
## Warning in fgseaMultilevel(pathways = pathways, stats = stats, minSize =
## minSize, : There were 3 pathways for which P-values were not calculated
## properly due to unbalanced (positive and negative) gene-level statistic values.
## For such pathways pval, padj, NES, log2err are set to NA. You can try to
## increase the value of the argument nPermSimple (for example set it nPermSimple
## = 10000)
```

```
## leading edge analysis...
```

```
## done...
```

```
#   return(gsea_results)
# }

# Step 5: Visualize / explore


head(gsea_results@result)
```

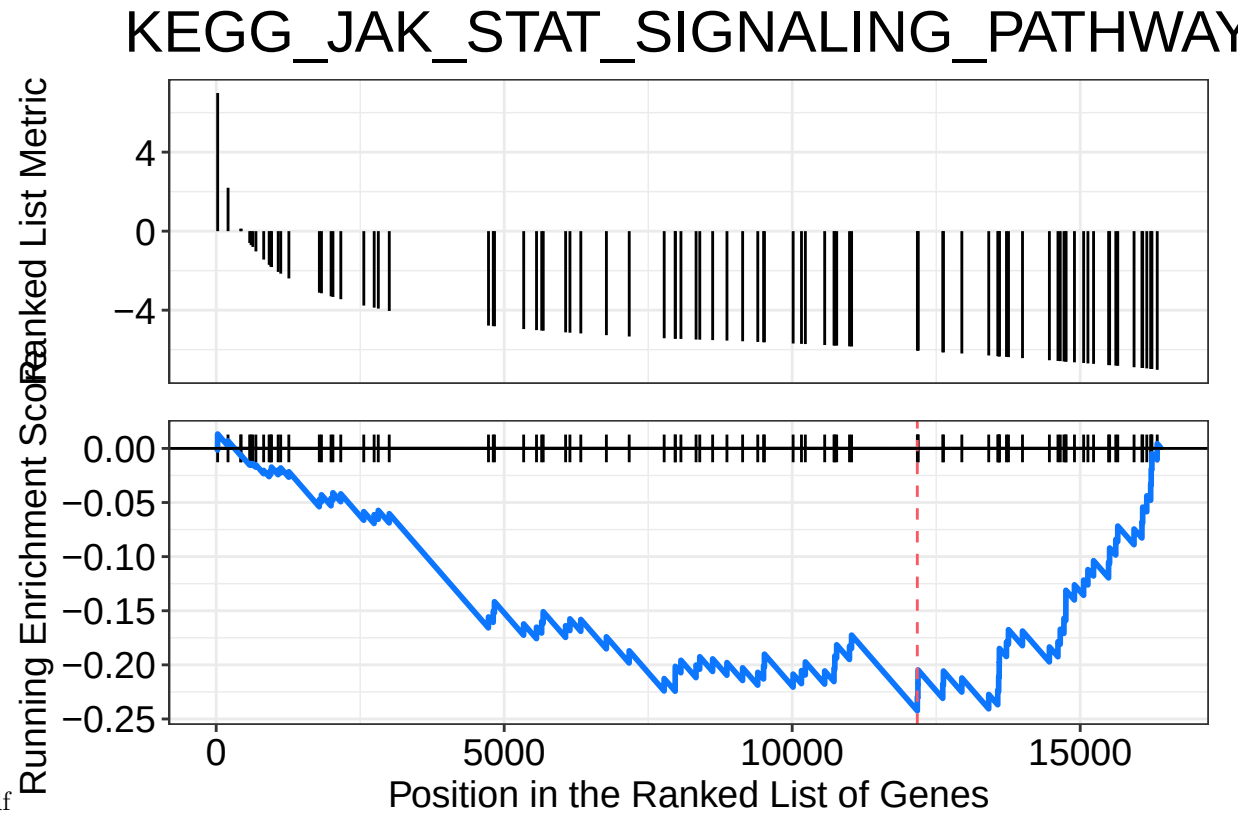
```
##                                                                      ID
## KEGG_SPLICEOSOME                                       KEGG_SPLICEOSOME
## KEGG_INSULIN_SIGNALING_PATHWAY               KEGG_INSULIN_SIGNALING_PATHWAY
## KEGG_REGULATION_OF_ACTIN_CYTOSKELETON KEGG_REGULATION_OF_ACTIN_CYTOSKELETON
## KEGG_ENDOCYTOSIS                                       KEGG_ENDOCYTOSIS
## KEGG_RNA_DEGRADATION                                 KEGG_RNA_DEGRADATION
## KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS     KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS
##                                                             Description
## KEGG_SPLICEOSOME                                       KEGG_SPLICEOSOME
## KEGG_INSULIN_SIGNALING_PATHWAY               KEGG_INSULIN_SIGNALING_PATHWAY
## KEGG_REGULATION_OF_ACTIN_CYTOSKELETON KEGG_REGULATION_OF_ACTIN_CYTOSKELETON
## KEGG_ENDOCYTOSIS                                       KEGG_ENDOCYTOSIS
## KEGG_RNA_DEGRADATION                                 KEGG_RNA_DEGRADATION
## KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS     KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS
##                                       setSize enrichmentScore       NES
## KEGG_SPLICEOSOME                          126      -0.4612522 -3.288915
## KEGG_INSULIN_SIGNALING_PATHWAY            111      -0.3688939 -2.574908
```

```
## KEGG_REGULATION_OF_ACTIN_CYTOSKELETON     147      -0.3387370 -2.482257
## KEGG_ENDOCYTOSIS                          153      -0.3091265 -2.277984
## KEGG_RNA_DEGRADATION                       55      -0.4263806 -2.574446
## KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS       124      -0.3228639 -2.298169
##                                             pvalue     p.adjust       qvalue
## KEGG_SPLICEOSOME                        2.375452e-17 2.921806e-15 1.300247e-15
## KEGG_INSULIN_SIGNALING_PATHWAY          5.511027e-09 2.259521e-07 1.005521e-07
## KEGG_REGULATION_OF_ACTIN_CYTOSKELETON 5.173198e-09 2.259521e-07 1.005521e-07
## KEGG_ENDOCYTOSIS                        1.633472e-07 5.022927e-06 2.235278e-06
## KEGG_RNA_DEGRADATION                    7.250611e-07 1.486375e-05 6.614592e-06
## KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS   6.867726e-07 1.486375e-05 6.614592e-06
##                                       rank                  leading_edge
## KEGG_SPLICEOSOME                      5073 tags=71%, list=31%, signal=49%
## KEGG_INSULIN_SIGNALING_PATHWAY        4340 tags=52%, list=26%, signal=39%
## KEGG_REGULATION_OF_ACTIN_CYTOSKELETON 4481 tags=47%, list=27%, signal=34%
## KEGG_ENDOCYTOSIS                      4258 tags=46%, list=26%, signal=35%
## KEGG_RNA_DEGRADATION                  4297 tags=60%, list=26%, signal=44%
## KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS   4868 tags=52%, list=30%, signal=37%
##
## KEGG_SPLICEOSOME                      84321/5093/10929/22916/6625/3310/84991/10286/51691/8559/9410/33
## KEGG_INSULIN_SIGNALING_PATHWAY
## KEGG_REGULATION_OF_ACTIN_CYTOSKELETON
## KEGG_ENDOCYTOSIS
## KEGG_RNA_DEGRADATION
## KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS
```

```
gsea_result_df <- data.frame(gsea_results@result)
gsea_result_df %>%
  # This returns the 3 rows with the largest NES values
  dplyr::slice_max(NES, n = 3)
```

```
##                                           ID             Description setSize
## KEGG_PURINE_METABOLISM   KEGG_PURINE_METABOLISM  KEGG_PURINE_METABOLISM     124
## KEGG_ALZHEIMERS_DISEASE KEGG_ALZHEIMERS_DISEASE KEGG_ALZHEIMERS_DISEASE     139
## KEGG_TIGHT_JUNCTION         KEGG_TIGHT_JUNCTION     KEGG_TIGHT_JUNCTION      82
##                          enrichmentScore       NES     pvalue   p.adjust
## KEGG_PURINE_METABOLISM       -0.2069111 -1.472808 0.02602603 0.04924925
## KEGG_ALZHEIMERS_DISEASE      -0.2143559 -1.555340 0.01115692 0.02450538
## KEGG_TIGHT_JUNCTION          -0.2391036 -1.569025 0.02297803 0.04486187
##                             qvalue rank                  leading_edge
## KEGG_PURINE_METABOLISM  0.02191665 7403 tags=55%, list=45%, signal=30%
## KEGG_ALZHEIMERS_DISEASE 0.01090526 5071 tags=41%, list=31%, signal=29%
## KEGG_TIGHT_JUNCTION     0.01996421 5835 tags=49%, list=36%, signal=32%
##
## KEGG_PURINE_METABOLISM  203/4833/5435/122622/56985/8833/10201/4832/51082/5438/3704/5138/29922/5558/8
## KEGG_ALZHEIMERS_DISEASE                                                                              7
## KEGG_TIGHT_JUNCTION
```

```
most_positive_nes_plot <- enrichplot::gseaplot(
  gsea_results,
  geneSetID = "KEGG_JAK_STAT_SIGNALING_PATHWAY",
  title = "KEGG_JAK_STAT_SIGNALING_PATHWAY",
  color.line = "#0d76ff"
)
most_positive_nes_plot
```

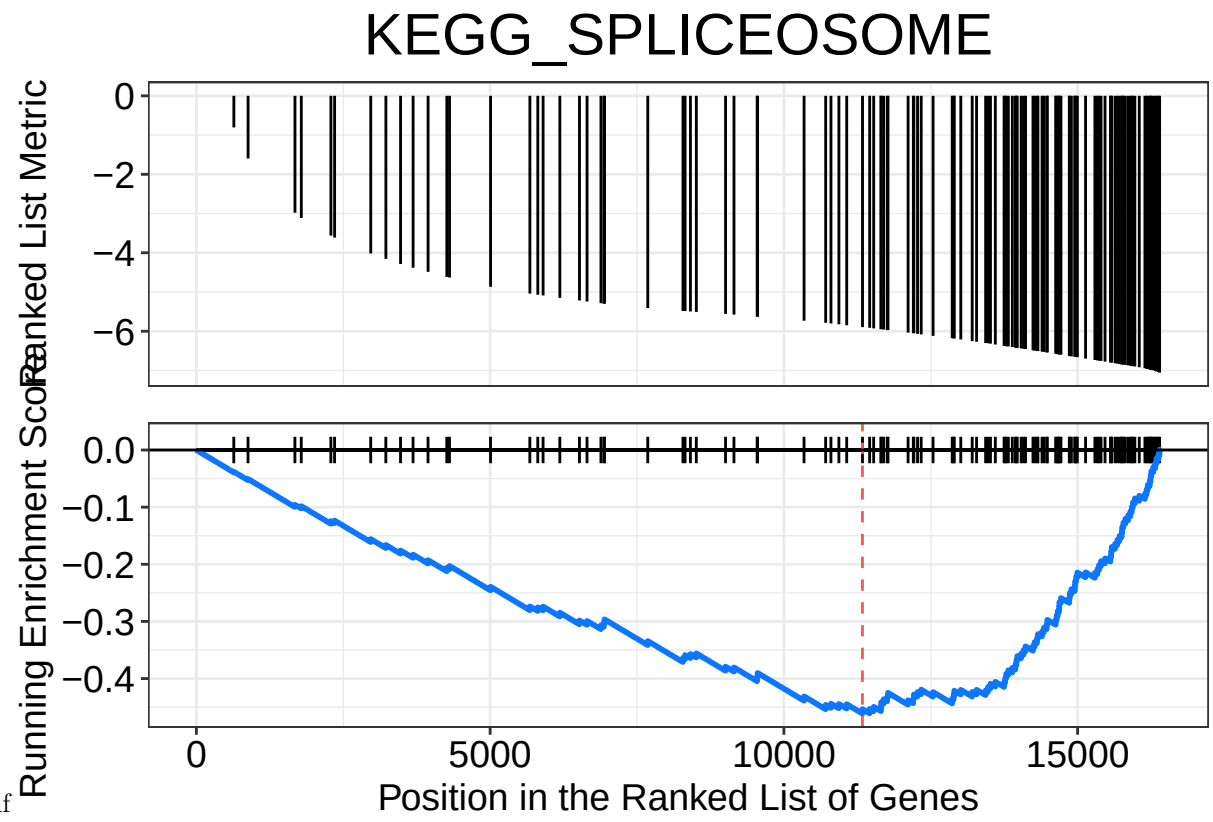


GSEA 2-1.pdf

```
gsea_result_df %>%
  # Return the 3 rows with the smallest (most negative) NES values
  dplyr::slice_min(NES, n = 3)
```

```
##                                                          ID
## KEGG_SPLICEOSOME                           KEGG_SPLICEOSOME
## KEGG_INSULIN_SIGNALING_PATHWAY KEGG_INSULIN_SIGNALING_PATHWAY
## KEGG_RNA_DEGRADATION                   KEGG_RNA_DEGRADATION
##                                                  Description setSize
## KEGG_SPLICEOSOME                           KEGG_SPLICEOSOME     126
## KEGG_INSULIN_SIGNALING_PATHWAY KEGG_INSULIN_SIGNALING_PATHWAY     111
## KEGG_RNA_DEGRADATION                   KEGG_RNA_DEGRADATION      55
##                                enrichmentScore       NES       pvalue
## KEGG_SPLICEOSOME                     -0.4612522 -3.288915 2.375452e-17
## KEGG_INSULIN_SIGNALING_PATHWAY       -0.3688939 -2.574908 5.511027e-09
## KEGG_RNA_DEGRADATION                 -0.4263806 -2.574446 7.250611e-07
##                                    p.adjust       qvalue rank
## KEGG_SPLICEOSOME               2.921806e-15 1.300247e-15 5073
## KEGG_INSULIN_SIGNALING_PATHWAY 2.259521e-07 1.005521e-07 4340
## KEGG_RNA_DEGRADATION           1.486375e-05 6.614592e-06 4297
##                                                leading_edge
## KEGG_SPLICEOSOME               tags=71%, list=31%, signal=49%
## KEGG_INSULIN_SIGNALING_PATHWAY tags=52%, list=26%, signal=39%
## KEGG_RNA_DEGRADATION           tags=60%, list=26%, signal=44%
##
## KEGG_SPLICEOSOME               84321/5093/10929/22916/6625/3310/84991/10286/51691/8559/9410/3304/515
## KEGG_INSULIN_SIGNALING_PATHWAY
## KEGG_RNA_DEGRADATION
```

```
most_negative_nes_plot <- enrichplot::gseaplot(
  gsea_results,
  geneSetID = "KEGG_SPLICEOSOME",
  title = "KEGG_SPLICEOSOME",
  color.line = "#0d76ff"
)
most_negative_nes_plot
```



GSEA 2-2.pdf

```
# Step 6: EXERCISE: alternatives to KEGG?

  gsea_bp_results <- GSEA(
    geneList = list_ordered, # Ordered ranked gene list
    minGSSize = 25, # Minimum gene set size
    maxGSSize = 500, # Maximum gene set set
    pvalueCutoff = 0.05, # p-value cutoff
    eps = 0, # Boundary for calculating the p value
    seed = FALSE, # Set seed to make results reproducible
    pAdjustMethod = "BH", # Benjamini-Hochberg correction
    TERM2GENE = dplyr::select(
      hs_BP_df,
      gs_name,
      human_entrez_gene
    )
  )
```

```
## preparing geneSet collections...
```

```
## GSEA analysis...
```

```
## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize, gseaParam, : There are ties in
## The order of those tied genes will be arbitrary, which may produce unexpected results.
```

```
## Warning in fgseaMultilevel(pathways = pathways, stats = stats, minSize =
## minSize, : There were 1 pathways for which P-values were not calculated
## properly due to unbalanced (positive and negative) gene-level statistic values.
## For such pathways pval, padj, NES, log2err are set to NA. You can try to
## increase the value of the argument nPermSimple (for example set it nPermSimple
## = 10000)
```

```
## Warning in fgseaMultilevel(pathways = pathways, stats = stats, minSize =
## minSize, : For some of the pathways the P-values were likely overestimated. For
## such pathways log2err is set to NA.
```

```
## leading edge analysis...
```

```
## done...
```

```
# Step 5: Visualize / explore


head(gsea_bp_results@result)
```

```
##
## GOBP_MRNA_PROCESSING                                                                GOBP_MRNA_PROCESSI
## GOBP_RNA_SPLICING                                                                     GOBP_RNA_SPLICI
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIO
## GOBP_MACROAUTOPHAGY                                                                 GOBP_MACROAUTOPHA
## GOBP_PROCESS_UTILIZING_AUTOPHAGIC_MECHANISM                GOBP_PROCESS_UTILIZING_AUTOPHAGIC_MECHANI
## GOBP_RNA_CATABOLIC_PROCESS                                                    GOBP_RNA_CATABOLIC_PROCE
##                                                                                                 Descripti
## GOBP_MRNA_PROCESSING                                                                GOBP_MRNA_PROCESSI
## GOBP_RNA_SPLICING                                                                     GOBP_RNA_SPLICI
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIO
## GOBP_MACROAUTOPHAGY                                                                 GOBP_MACROAUTOPHA
## GOBP_PROCESS_UTILIZING_AUTOPHAGIC_MECHANISM                GOBP_PROCESS_UTILIZING_AUTOPHAGIC_MECHANI
## GOBP_RNA_CATABOLIC_PROCESS                                                    GOBP_RNA_CATABOLIC_PROCE
##                                                    setSize enrichmentScore
## GOBP_MRNA_PROCESSING                                   447      -0.3606072
## GOBP_RNA_SPLICING                                      394      -0.3639290
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS    292      -0.3754122
## GOBP_MACROAUTOPHAGY                                    283      -0.3730996
## GOBP_PROCESS_UTILIZING_AUTOPHAGIC_MECHANISM            481      -0.3068724
## GOBP_RNA_CATABOLIC_PROCESS                             252      -0.3638191
##                                                         NES       pvalue
## GOBP_MRNA_PROCESSING                                -3.026140 7.874547e-31
## GOBP_RNA_SPLICING                                   -3.009089 9.600104e-28
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS -3.010029 3.968022e-23
## GOBP_MACROAUTOPHAGY                                 -2.987996 1.229793e-21
## GOBP_PROCESS_UTILIZING_AUTOPHAGIC_MECHANISM         -2.592134 3.767558e-21
## GOBP_RNA_CATABOLIC_PROCESS                          -2.865106 3.175123e-18
##                                                        p.adjust       qvalue
## GOBP_MRNA_PROCESSING                                1.921390e-27 1.076741e-27
## GOBP_RNA_SPLICING                                   1.171213e-24 6.563439e-25
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS 3.227324e-20 1.808583e-20
## GOBP_MACROAUTOPHAGY                                 7.501739e-19 4.203951e-19
## GOBP_PROCESS_UTILIZING_AUTOPHAGIC_MECHANISM         1.838568e-18 1.030328e-18
```

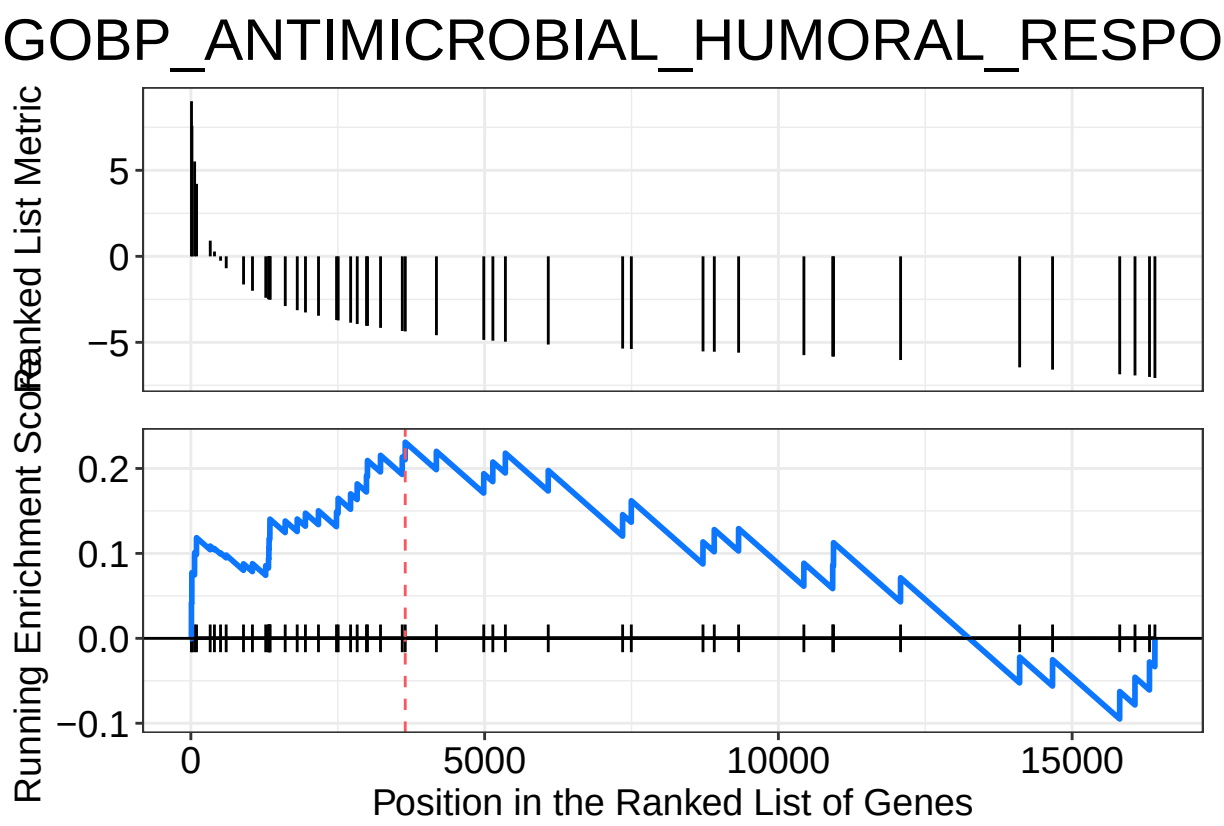
```
## GOBP_RNA_CATABOLIC_PROCESS                             1.291217e-15 7.235938e-16
##                                                        rank
## GOBP_MRNA_PROCESSING                                   4758
## GOBP_RNA_SPLICING                                      4883
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS    4758
## GOBP_MACROAUTOPHAGY                                    4647
## GOBP_PROCESS_UTILIZING_AUTOPHAGIC_MECHANISM            4756
## GOBP_RNA_CATABOLIC_PROCESS                             5763
##                                                                      leading_edge
## GOBP_MRNA_PROCESSING                                   tags=56%, list=29%, signal=41%
## GOBP_RNA_SPLICING                                      tags=57%, list=30%, signal=41%
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS    tags=58%, list=29%, signal=42%
## GOBP_MACROAUTOPHAGY                                    tags=54%, list=28%, signal=39%
## GOBP_PROCESS_UTILIZING_AUTOPHAGIC_MECHANISM            tags=48%, list=29%, signal=35%
## GOBP_RNA_CATABOLIC_PROCESS                             tags=61%, list=35%, signal=40%
##
## GOBP_MRNA_PROCESSING                                   22916/6625/54496/84991/50628/10286/54960/79760/9
## GOBP_RNA_SPLICING
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS
## GOBP_MACROAUTOPHAGY
## GOBP_PROCESS_UTILIZING_AUTOPHAGIC_MECHANISM
## GOBP_RNA_CATABOLIC_PROCESS
```

```
gsea_result_df <- data.frame(gsea_bp_results@result)
gsea_result_df %>%
  # This returns the 3 rows with the largest NES values
  dplyr::slice_max(NES, n = 3)
```

```
##
## GOBP_ANTIMICROBIAL_HUMORAL_RESPONSE                                    GOBP_ANTIMICROBIAL_HUMORAL_RE
## GOBP_DNA_REPLICATION_DEPENDENT_CHROMATIN_ORGANIZATION GOBP_DNA_REPLICATION_DEPENDENT_CHROMATIN_ORGANI
## GOBP_MONOCYTE_DIFFERENTIATION                                             GOBP_MONOCYTE_DIFFEREN
##                                                                                                  Desc
## GOBP_ANTIMICROBIAL_HUMORAL_RESPONSE                                    GOBP_ANTIMICROBIAL_HUMORAL_RE
## GOBP_DNA_REPLICATION_DEPENDENT_CHROMATIN_ORGANIZATION GOBP_DNA_REPLICATION_DEPENDENT_CHROMATIN_ORGANI
## GOBP_MONOCYTE_DIFFERENTIATION                                             GOBP_MONOCYTE_DIFFEREN
##                                                      setSize enrichmentScore
## GOBP_ANTIMICROBIAL_HUMORAL_RESPONSE                       49       0.2309357
## GOBP_DNA_REPLICATION_DEPENDENT_CHROMATIN_ORGANIZATION     32       0.2468140
## GOBP_MONOCYTE_DIFFERENTIATION                             28       0.2631942
##                                                          NES      pvalue
## GOBP_ANTIMICROBIAL_HUMORAL_RESPONSE                   2.081346 0.002336308
## GOBP_DNA_REPLICATION_DEPENDENT_CHROMATIN_ORGANIZATION 1.704187 0.013819997
## GOBP_MONOCYTE_DIFFERENTIATION                         1.659422 0.015482139
##                                                        p.adjust      qvalue
## GOBP_ANTIMICROBIAL_HUMORAL_RESPONSE                   0.01104766 0.006191073
## GOBP_DNA_REPLICATION_DEPENDENT_CHROMATIN_ORGANIZATION 0.04236499 0.023741211
## GOBP_MONOCYTE_DIFFERENTIATION                         0.04612505 0.025848336
##                                                       rank
## GOBP_ANTIMICROBIAL_HUMORAL_RESPONSE                   3649
## GOBP_DNA_REPLICATION_DEPENDENT_CHROMATIN_ORGANIZATION 6466
## GOBP_MONOCYTE_DIFFERENTIATION                         2177
##                                                                      leading_edge
## GOBP_ANTIMICROBIAL_HUMORAL_RESPONSE                   tags=59%, list=22%, signal=46%
## GOBP_DNA_REPLICATION_DEPENDENT_CHROMATIN_ORGANIZATION tags=75%, list=39%, signal=46%
```

```
## GOBP_MONOCYTE_DIFFERENTIATION                                tags=54%, list=13%, signal=47%
##
## GOBP_ANTIMICROBIAL_HUMORAL_RESPONSE                     6280/6036/6283/2919/2597/8349/8970/85236/8344/8
## GOBP_DNA_REPLICATION_DEPENDENT_CHROMATIN_ORGANIZATION                                8355/8358/8356/8352/8368
## GOBP_MONOCYTE_DIFFERENTIATION
```

```
most_positive_nes_plot <- enrichplot::gseaplot(
  gsea_bp_results,
  geneSetID = "GOBP_ANTIMICROBIAL_HUMORAL_RESPONSE",
  title = "GOBP_ANTIMICROBIAL_HUMORAL_RESPONSE",
  color.line = "#0d76ff"
)
most_positive_nes_plot
```

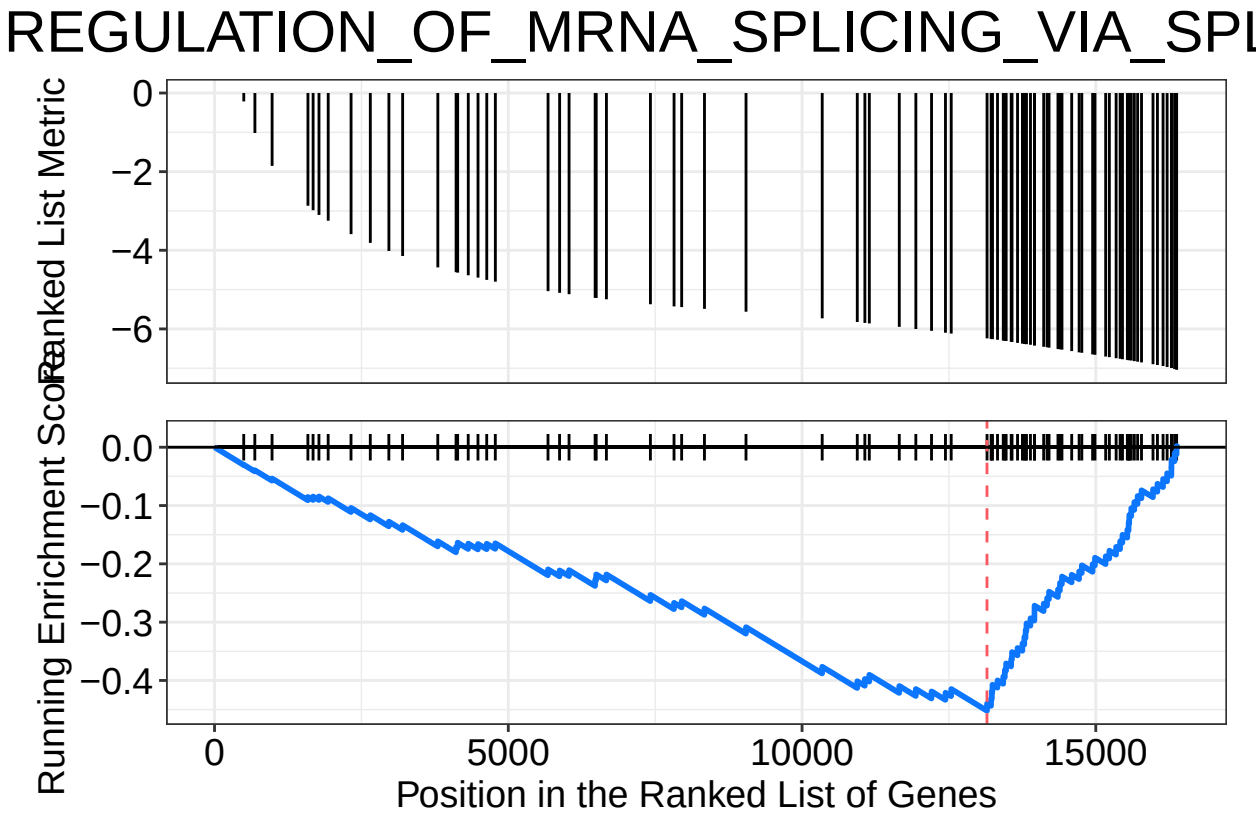



```
gsea_result_df %>%
  # Return the 3 rows with the smallest (most negative) NES values
  dplyr::slice_min(NES, n = 3)
```

```
##
## GOBP_MRNA_PROCESSING                                                                GOBP_MRNA_PROCESSI
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTI
## GOBP_RNA_SPLICING                                                                     GOBP_RNA_SPLICI
##                                                                                             Descripti
## GOBP_MRNA_PROCESSING                                                                GOBP_MRNA_PROCESSI
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTI
## GOBP_RNA_SPLICING                                                                     GOBP_RNA_SPLICI
##                                                    setSize enrichmentScore
## GOBP_MRNA_PROCESSING                                   447      -0.3606072
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS    292      -0.3754122
```

```
## GOBP_RNA_SPLICING                                          394      -0.3639290
##                                                              NES       pvalue
## GOBP_MRNA_PROCESSING                                   -3.026140 7.874547e-31
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS    -3.010029 3.968022e-23
## GOBP_RNA_SPLICING                                      -3.009089 9.600104e-28
##                                                            p.adjust       qvalue
## GOBP_MRNA_PROCESSING                                   1.921390e-27 1.076741e-27
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS    3.227324e-20 1.808583e-20
## GOBP_RNA_SPLICING                                      1.171213e-24 6.563439e-25
##                                                        rank
## GOBP_MRNA_PROCESSING                                   4758
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS    4758
## GOBP_RNA_SPLICING                                      4883
##                                                                          leading_edge
## GOBP_MRNA_PROCESSING                                   tags=56%, list=29%, signal=41%
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS    tags=58%, list=29%, signal=42%
## GOBP_RNA_SPLICING                                      tags=57%, list=30%, signal=41%
##
## GOBP_MRNA_PROCESSING                                   22916/6625/54496/84991/50628/10286/54960/79760/9
## GOBP_RNA_SPLICING_VIA_TRANSESTERIFICATION_REACTIONS
## GOBP_RNA_SPLICING
```

```r
most_negative_nes_plot <- enrichplot::gseaplot(
  gsea_bp_results,
  geneSetID = "GOBP_REGULATION_OF_MRNA_SPLICING_VIA_SPLICEOSOME",
  title = "GOBP_REGULATION_OF_MRNA_SPLICING_VIA_SPLICEOSOME",
  color.line = "#0d76ff"
)
most_negative_nes_plot
```



Step 7: EXERCISE: compare GSEA vs ORA?

GSEA uses a sorted list of genes regardless of their significance, and rather relies on the order to determine on whether a particular gene set is enriched or not. ORA on the other hand uses a predefined filtered gene list, and searches for the gene sets with largest overlap with the provided list.

In our analysis, GSEA shows GOBP_ANTIMICROBIAL_HUMORAL_RESPONSE as the set with the highest enrichment score (NES). Whereas for ORA, GOBP_RESPONSE_TO_BACTERIUM and GOBP_PROCESS_UTILIZING_AUTOPHAGIC_MECHANISMS are more prominent in the DEG lists.

Lets conduct GeneSetCluster.

```
# Healthy vs Group Covid19
# We prepare a function from the previous analysis
load("../GSE198256_RNAseq/results_DEG_Limma_2.Rda")

# Healthy vs Covid19AI
# Diff_HvsAI <- topTable(fit2,coef=1,p.value=1,number=nrow(logCPM))
# Diff_Hvs6Mo <- topTable(fit2,coef=2,p.value=1,number=nrow(logCPM))
# # Healthy vs Covid196Mo
# Diff_Hvs6Mo <- topTable(fit2,coef=3,p.value=1,number=nrow(logCPM))
require(limma)
Diff_HvAI = topTable(fit2,coef=1, p.value=1, number = nrow(fit2))
Diff_AIv3Mo = topTable(fit2,coef=2, p.value=1, number = nrow(fit2))
Diff_3Mov6Mo = topTable(fit2,coef=3, p.value=1, number = nrow(fit2))
```

```r
Diff_6MovH = topTable(fit2,coef=4, p.value=1, number = nrow(fit2))


hs_msigdb_df <- msigdbr(species = "Homo sapiens")
hs_kegg_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C5", # This is to filter only to the C2 curated gene sets
    gs_subcat == "GO:BP" # This is because we only want KEGG pathways
  )


doGSEA <- function(diff_table) {
  list_ordered <- diff_table[,"B"]
  names(list_ordered) <- rownames(diff_table)

  return(GSEA(
    geneList = list_ordered, # Ordered ranked gene list
    minGSSize = 25, # Minimum gene set size
    maxGSSize = 500, # Maximum gene set set
    pvalueCutoff = 0.05, # p-value cutoff
    eps = 0, # Boundary for calculating the p value
    seed = FALSE, # Set seed to make results reproducible
    pAdjustMethod = "BH", # Benjamini-Hochberg correction
    TERM2GENE = dplyr::select(
      hs_kegg_df,
      gs_name,
      human_entrez_gene
    )
  ))
}

GSEA_HvAI <- doGSEA(Diff_HvAI)
```

```
## preparing geneSet collections...
```

```
## GSEA analysis...
```

```
## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize, gseaParam, : There are ties in
## The order of those tied genes will be arbitrary, which may produce unexpected results.
```

```
## Warning in fgseaMultilevel(pathways = pathways, stats = stats, minSize =
## minSize, : There were 4 pathways for which P-values were not calculated
## properly due to unbalanced (positive and negative) gene-level statistic values.
## For such pathways pval, padj, NES, log2err are set to NA. You can try to
## increase the value of the argument nPermSimple (for example set it nPermSimple
## = 10000)
```

```
## Warning in fgseaMultilevel(pathways = pathways, stats = stats, minSize =
## minSize, : For some of the pathways the P-values were likely overestimated. For
## such pathways log2err is set to NA.
```

```
## leading edge analysis...
```

```
## done...
```

```r
GSEA_AIv3Mo <- doGSEA(Diff_AIv3Mo)
```

```
## preparing geneSet collections...

## GSEA analysis...

## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize, gseaParam, : There are ties in
## The order of those tied genes will be arbitrary, which may produce unexpected results.

## Warning in fgseaMultilevel(pathways = pathways, stats = stats, minSize =
## minSize, : There were 1 pathways for which P-values were not calculated
## properly due to unbalanced (positive and negative) gene-level statistic values.
## For such pathways pval, padj, NES, log2err are set to NA. You can try to
## increase the value of the argument nPermSimple (for example set it nPermSimple
## = 10000)

## leading edge analysis...

## done...
```

```
GSEA_3Mov6Mo <- doGSEA(Diff_3Mov6Mo)
```

```
## preparing geneSet collections...

## GSEA analysis...

## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize, gseaParam, : There are ties in
## The order of those tied genes will be arbitrary, which may produce unexpected results.

## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize, gseaParam, : There were 1 pathu

## leading edge analysis...

## done...
```

```
GSEA_6MovH <- doGSEA(Diff_6MovH)
```

```
## preparing geneSet collections...

## GSEA analysis...

## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize, gseaParam, : There are ties in
## The order of those tied genes will be arbitrary, which may produce unexpected results.

## leading edge analysis...

## done...
```

```
path <- "../gene_set_clusters"

write.csv(GSEA_HvAI, file = paste0(path, "/GSEA_HvsAI.csv"), row.names = FALSE)
write.csv(GSEA_AIv3Mo, file = paste0(path, "/GSEA_AIvs3Mo.csv"), row.names = FALSE)
write.csv(GSEA_3Mov6Mo, file = paste0(path, "/GSEA_3Movs6Mo.csv"), row.names = FALSE)
write.csv(GSEA_6MovH, file = paste0(path, "/GSEA_6MovsH.csv"), row.names = FALSE)
```

```
######## Check if the required packages are installed ########
# packages <- c("limma","stats","methods","RColorBrewer","clusterProfiler","GGally",
#               "network","clustree","readxl","org.Hs.eg.db",
#               "org.Mm.eg.db","cluster","factoextra","STRINGdb","WebGestaltR","stringr",
#               "AnnotationDbi","ComplexHeatmap","GO.db","GetoptLong","bigstatsr","colorRamp2",
#               "cowplot","doParallel","dplyr","foreach","ggdendro","ggnewscale","ggplot2",
#               "ggtree","ggwordcloud","grid","httr","jsonlite","parallel","patchwork","pbapply",
#               "reshape2","rgl","seriation","simplifyEnrichment","slam","tidyverse","umap",
#                "utils","grDevices")
#
```

```
# new.packages <- packages[!(packages %in% installed.packages()[,"Package"])]
#
# install.packages(new.packages)

# path <- "gene_set_clusters"
# install.packages(path, repos=NULL, type='source')
```

```
require(GeneSetCluster)
```

```
## Loading required package: GeneSetCluster
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
## Warning: replacing previous import 'AnnotationDbi::select' by
## 'clusterProfiler::select' when loading 'GeneSetCluster'
```

```
## Warning: replacing previous import 'ComplexHeatmap::%v%' by 'network::%v%' when
## loading 'GeneSetCluster'
```

```
##
```

```
## Warning: replacing previous import 'cowplot::align_plots' by
## 'patchwork::align_plots' when loading 'GeneSetCluster'
```

```
## Warning: replacing previous import 'dplyr::lag' by 'stats::lag' when loading
## 'GeneSetCluster'
```

```
## Warning: replacing previous import 'dplyr::filter' by 'stats::filter' when
## loading 'GeneSetCluster'
```

```
## Warning: replacing previous import 'AnnotationDbi::tail' by 'utils::tail' when
## loading 'GeneSetCluster'
```

```
## Warning: replacing previous import 'AnnotationDbi::head' by 'utils::head' when
## loading 'GeneSetCluster'
```

```
# path <- "../gene_set_clusters/"
# GSEA.files <- paste0(path, list.files(path, pattern = ".csv"))
#
# keep.files = gsub(".csv","",basename(GSEA.files)) %in% c("GSEA_HvsAI", "GSEA_AIvs3Mo")
# GSEA.files = GSEA.files[keep.files]
#
#
# # Load the data and create Pathway object
# # Automatically for GSEA, GREAT or IPA
# GSEA.Object1 <- LoadGeneSets(file_location = GSEA.files,
#                              # groupnames= c("GSEA_HvsAI", "GSEA_AIvs3Mo", "GSEA_3Movs6Mo", "GSEA_6M
#                              groupnames= c("GSEA_HvsAI", "GSEA_AIvs3Mo"), # names of the groups
#                              P.cutoff = 0.05, # cut off the p.adjust
#                              Mol.cutoff = 15, # minimum number of genes per pathway
#                              Source = "GSEA", # the analysis (GSEA, GREAT or IPA)
#                              structure = "ENTREZID", # Gene type (SYMBOL, ENTREZID, ENSEMBLID)
#                              Organism = "org.Hs.eg.db", # database: Homo Sapiens or Mus musculus
#                              seperator = "/") # the separator used for listing genes
#
# # IMPORTANT when created manually, it is assumed that the pathways have been filtered by p-value and
# # Make sure you have filtered your data
```

```
# # GSEA.Object1Manual <- ObjectCreator(Pathways = c(GSEA_HvsAI@result$ID,
# #                                                  GSEA_Hvs6Mo@result$ID),
# #                                     Molecules = c(GSEA_HvsAI@result$core_enrichment,
# #                                                   GSEA_Hvs6Mo@result$core_enrichment),
# #                                     Groups = c(rep("GSEA_HvsAI", times=nrow(GSEA_HvsAI@result)),
# #                                                rep("GSEA_Hvs6Mo", times=nrow(GSEA_Hvs6Mo@result))),
# #                                     Pvalues = c(GSEA_HvsAI@result$p.adjust,  # optional
# #                                                 GSEA_Hvs6Mo@result$p.adjust),
# #                                     enrichmentScore = c(GSEA_HvsAI@result$NES, # optional
# #                                                         GSEA_Hvs6Mo@result$NES),
# #                                     structure = "ENTREZID", Type = "", sep = "/",
# #                                     Source = "GSEA", organism = "org.Hs.eg.db")
#
# GSEA.Object2 <- CombineGeneSets(Object = GSEA.Object1,
#                                 combineMethod = "Standard", threads = 4)
# ## over consume resources
#
# OptimalGeneSets(Object = GSEA.Object2,
#                 uniquePathway = FALSE, # consider all the pathways (also repeated) or the unique path
#                 method = "silhouette", max_cluster= 24, cluster_method = "kmeans", main= "Kmeans for
#
# OptimalGeneSets(Object = GSEA.Object2,
#                 uniquePathway = TRUE, # consider all the pathways (also repeated) or the unique pathw
#                 method = "silhouette", max_cluster= 24, cluster_method = "kmeans", main= "Kmeans for
#
# # in both cases the optimal cluster is 2
#
# GSEA.Object3 <- ClusterGeneSets(Object = GSEA.Object2,
#                                 clusters = 2, # consider all the pathways (also repeated) or the uniq
#                                 method = "Hierarchical", # Hierarchical clustering or kmeans
#                                 order = "cluster",
#                                 molecular.signature = "All")
#
# # plot results for both all pathways and unique pathways
# plotnounique <- PlotGeneSets(GSEA.Object3,
#                              uniquePathways = FALSE,
#                              wordcloud = FALSE, # wordcloud only supported for GO terms
#                              doORA = T) # do ora per cluster
#
# plotunique <- PlotGeneSets(GSEA.Object3,
#                            uniquePathways = TRUE,
#                            wordcloud = FALSE, # wordcloud only supported for GO terms
#                            doORA = T) # do ora per cluster
#
# plotunique
```

```
# let's say we are interested in exploring cluster 2 in plotunique. Lets break up this cluste for furth

# plotoptimalcluster2 <- OptimalGeneSets(Object = GSEA.Object3,
#                 uniquePathway = TRUE, # consider all the pathways (also repeated) or the unique pathw
#                 cluster = 2, # which cluster
#                 method = "silhouette", max_cluster= 24, cluster_method = "kmeans", main= "Kmeans for
#
# plotoptimalcluster2 # optimal 2 break up cluster 2 in 2 clusters
```

```
#
# GSEA.Object3breakup <- BreakUpCluster(GSEA.Object3,
#                                       breakup.cluster = 2, # which cluster
#                                       sub.cluster = 2, # in how many cluster split up
#                                       uniquePathways = TRUE) # conside unique pathways
#
# plotuniquebreakup <- PlotGeneSets(GSEA.Object3breakup,
#                                   uniquePathways = TRUE,
#                                   wordcloud = FALSE, # wordcloud only supported for GO terms
#                                   doORA = T) # do ora per cluster
#
# plotuniquebreakup
```

```
# Now break up the cluster 1
# plotoptimalcluster1 <- OptimalGeneSets(Object = GSEA.Object3,
#                 uniquePathway = TRUE, # consider all the pathways (also repeated) or the unique pathw
#                 cluster = 1, # which cluster
#                 method = "silhouette", max_cluster= 24, cluster_method = "kmeans", main= "Kmeans for
#
# plotoptimalcluster1 # optimal 1 break up cluster 1 in 8 clusters
#
# GSEA.Object3breakup2 <- BreakUpCluster(GSEA.Object3breakup,
#                                       breakup.cluster = 1, # which cluster
#                                       sub.cluster = 8, # in how many cluster split up
#                                       uniquePathways = TRUE) # conside unique pathways
#
# plotuniquebreakup2 <- PlotGeneSets(GSEA.Object3breakup2,
#                                   uniquePathways = TRUE,
#                                   wordcloud = TRUE, # wordcloud only supported for GO terms
#                                   doORA = T) # do ora per cluster
#
# plotuniquebreakup2
```

It would seem the most dominant GSEA sets are not particular to one stage (Healthy, Acute, etc.) vs another, but rather they are abundant in all stages.