The transcriptional response during human cutaneous leishmaniasis

DIYtranscriptomics class (modified by Sébastien Wieckowski

2020-06-02

Contents

Introduction

During the Spring 2020 offering of **DIYtranscriptomics**, we analyzed a subset of patients and healthy controls from Amorim et al., 2019. This reproducible and dynamic report was created using Rmarkdown and the Knitr package, and summarizes the basic code and outputs (plots, tables, etc) produced during the course. This version was adapted for exporting to a PDF, i.e. removing interactive plots and tables.

R packages used

A variety of R packages was used for this analysis. All graphics and data wrangling were handled using the tidyverse suite of packages. All packages used are available from the Comprehensive R Archive Network (CRAN), Bioconductor.org, or Github.

Read mapping

Aligning raw reads with Kallisto

Raw reads were mapped to the human reference transcriptome using Kallisto, version 0.46.2. The quality of raw reads, as well as the results of Kallisto mapping are summarized in this summary report generated using fastqc and multiqc.

Importing count data into R

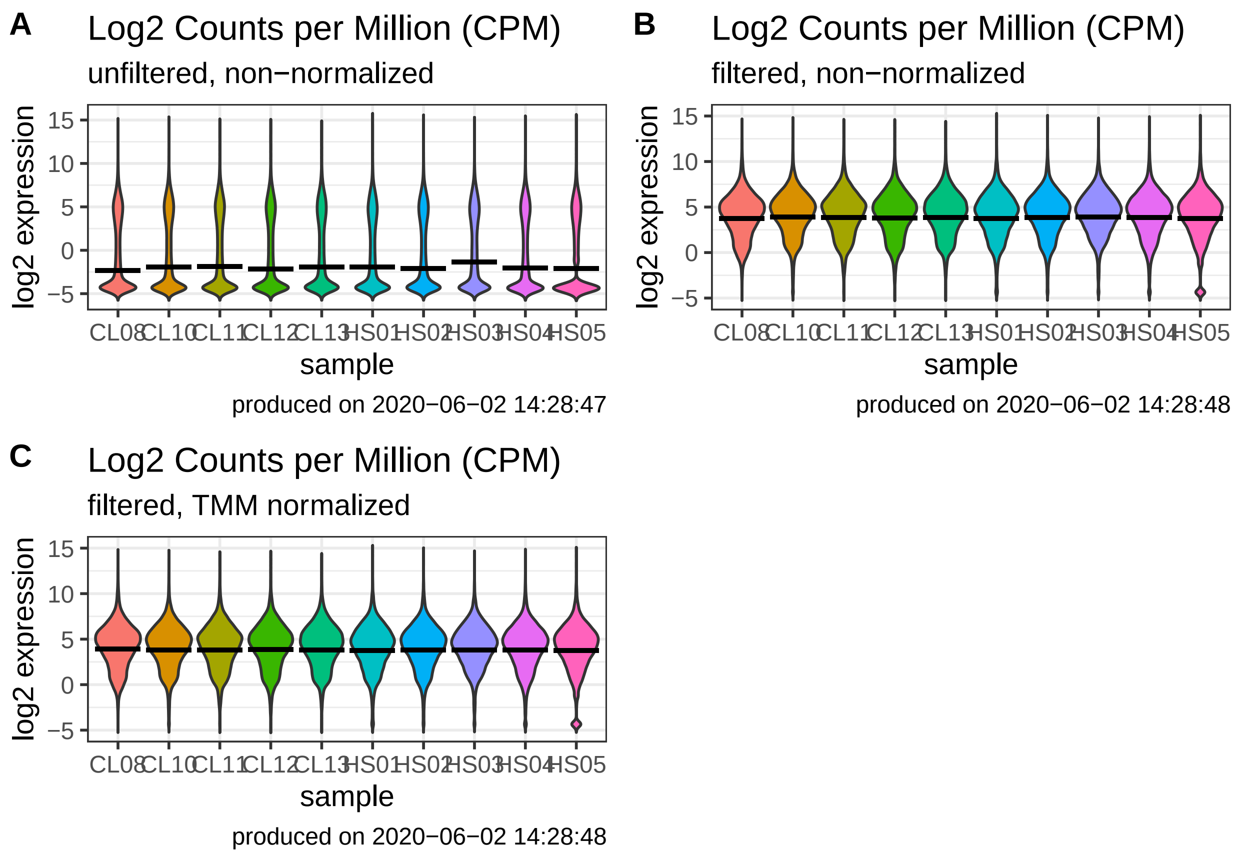
After read mapping with Kallisto, TxImport was used to read kallisto outputs into the R environment. Annotation data from Biomart was used to summarize data from transcript-level to gene-level.

```r
library(tidyverse) # provides access to Hadley Wickham's collection of R packages for data science
library(tximport) # package for getting Kallisto results into R
library(ensembldb) # helps deal with ensembl
library(EnsDb.Hsapiens.v86) # replace with your organism-specific database package
targets <- read_tsv("../3-read_mapping_Kallisto/test/studydesign.txt") # read in your study design
path <- file.path("../3-read_mapping_Kallisto/test", targets$sample, "abundance.tsv") # set file paths
Tx <- transcripts(EnsDb.Hsapiens.v86, columns=c("tx_id", "gene_name"))
Tx <- as_tibble(Tx)
Tx <- dplyr::rename(Tx, target_id = tx_id)
Tx <- dplyr::select(Tx, "target_id", "gene_name")
Txi_gene <- tximport(path,
                     type = "kallisto",
                     tx2gene = Tx,
                     txOut = FALSE, # determines whether your data represented at transcript or gene le
                     countsFromAbundance = "lengthScaledTPM",
                     ignoreTxVersion = TRUE)
```

Preprocessing

Impact of filtering and normalization

```r
library(tidyverse)
library(edgeR)
library(matrixStats)
library(cowplot)

sampleLabels <- targets$sample
myDGEList <- DGEList(Txi_gene$counts)
log2.cpm <- cpm(myDGEList, log=TRUE)

log2.cpm.df <- as_tibble(log2.cpm, rownames = "geneID")
colnames(log2.cpm.df) <- c("geneID", sampleLabels)
log2.cpm.df.pivot <- pivot_longer(log2.cpm.df, # dataframe to be pivoted
                                  cols = HS01:CL13, # column names to be stored as a SINGLE variable
                                  names_to = "samples", # name of that new variable (column)
                                  values_to = "expression") # name of new variable (column) storing all

p1 <- ggplot(log2.cpm.df.pivot) +
  aes(x=samples, y=expression, fill=samples) +
  geom_violin(trim = FALSE, show.legend = FALSE) +
  stat_summary(fun = "median",
               geom = "point",
               shape = 95,
               size = 10,
               color = "black",
               show.legend = FALSE) +
  labs(y="log2 expression", x = "sample",
       title="Log2 Counts per Million (CPM)",
       subtitle="unfiltered, non-normalized",
       caption=paste0("produced on ", Sys.time())) +
  theme_bw()

cpm <- cpm(myDGEList)
keepers <- rowSums(cpm>1)>=5 #user defined
myDGEList.filtered <- myDGEList[keepers,]

log2.cpm.filtered <- cpm(myDGEList.filtered, log=TRUE)
log2.cpm.filtered.df <- as_tibble(log2.cpm.filtered, rownames = "geneID")
colnames(log2.cpm.filtered.df) <- c("geneID", sampleLabels)
log2.cpm.filtered.df.pivot <- pivot_longer(log2.cpm.filtered.df, # dataframe to be pivoted
                                           cols = HS01:CL13, # column names to be stored as a SINGLE va
                                           names_to = "samples", # name of that new variable (column)
                                           values_to = "expression") # name of new variable (column) st

p2 <- ggplot(log2.cpm.filtered.df.pivot) +
  aes(x=samples, y=expression, fill=samples) +
  geom_violin(trim = FALSE, show.legend = FALSE) +
  stat_summary(fun = "median",
               geom = "point",
               shape = 95,
               size = 10,
               color = "black",
               show.legend = FALSE) +
  labs(y="log2 expression", x = "sample",
       title="Log2 Counts per Million (CPM)",
       subtitle="filtered, non-normalized",
       caption=paste0("produced on ", Sys.time())) +
  theme_bw()

myDGEList.filtered.norm <- calcNormFactors(myDGEList.filtered, method = "TMM")
log2.cpm.filtered.norm <- cpm(myDGEList.filtered.norm, log=TRUE)
log2.cpm.filtered.norm.df <- as_tibble(log2.cpm.filtered.norm, rownames = "geneID")
colnames(log2.cpm.filtered.norm.df) <- c("geneID", sampleLabels)
log2.cpm.filtered.norm.df.pivot <- pivot_longer(log2.cpm.filtered.norm.df, # dataframe to be pivoted
                                                cols = HS01:CL13, # column names to be stored as a SING
                                                names_to = "samples", # name of that new variable (colu
                                                values_to = "expression") # name of new variable (colum

p3 <- ggplot(log2.cpm.filtered.norm.df.pivot) +
  aes(x=samples, y=expression, fill=samples) +
  geom_violin(trim = FALSE, show.legend = FALSE) +
  stat_summary(fun = "median",
               geom = "point",
               shape = 95,
               size = 10,
               color = "black",
               show.legend = FALSE) +
  labs(y="log2 expression", x = "sample",
       title="Log2 Counts per Million (CPM)",
       subtitle="filtered, TMM normalized",
       caption=paste0("produced on ", Sys.time())) +
  theme_bw()

plot_grid(p1, p2, p3, labels = c('A', 'B', 'C'), label_size = 12)
```



Filtering was carried out to remove lowly expressed genes. Genes with less than 1 count per million (CPM) in at least 5 or more samples filtered out. This reduced the number of genes from 35643 to 15944.

---

table of filtered and normalized data

```
library(tidyverse)
library(gt)

mydata.df <- mutate(log2.cpm.filtered.norm.df,
                    healthy.AVG = (HS01 + HS02 + HS03 + HS04 + HS05)/5,
                    disease.AVG = (CL08 + CL10 + CL11 + CL12 + CL13)/5,
                    #now make columns comparing each of the averages above that you're interested in
                    LogFC = (disease.AVG - healthy.AVG)) %>% #note that this is the first time you've s
  mutate_if(is.numeric, round, 2)

# Produce publication-quality tables using the gt package

mydata.filter <- mydata.df %>%
  dplyr::filter(geneID=="MMP1" | geneID=="GZMB" | geneID=="IL1B" | geneID=="GNLY" | geneID=="IFNG"
                | geneID=="CCL4" | geneID=="KIR2DL4" | geneID=="PRF1" | geneID=="APOBEC3A" | geneID=="U
```

```r
  dplyr::select(geneID, healthy.AVG, disease.AVG, LogFC) %>%
  dplyr::arrange(desc(LogFC))

# gt table with a few more options
mydata.filter %>%
  gt() %>%
  fmt_number(columns=2:4, decimals = 1) %>%
  tab_header(title = md("**Regulators of skin pathogenesis**"),
             subtitle = md("*during cutaneous leishmaniasis*")) %>%
  tab_footnote(
    footnote = "Deletion or blockaid ameliorates disease in mice",
    locations = cells_body(
      columns = vars(geneID),
      rows = c(6, 7))) %>%
  tab_footnote(
    footnote = "Associated with treatment failure in multiple studies",
    locations = cells_body(
      columns = vars(geneID),
      rows = c(2:10))) %>%
  tab_footnote(
    footnote = "Implicated in parasite control",
    locations = cells_body(
      columns = vars(geneID),
      rows = c(2))) %>%
  tab_source_note(
    source_note = md("Reference: Amorim *et al*., (2019). DOI: 10.1126/scitranslmed.aar3619"))
```

Regulators of skin pathogenesis

during cutaneous leishmaniasis

geneID	healthy.AVG	disease.AVG	LogFC
MMP1	1.4	11.7	10.4
IFNG[1,2]	−3.8	4.3	8.1
CCL4[1]	−1.7	6.1	7.9
GZMB[1]	−0.6	6.3	6.9
GNLY[1]	0.9	7.0	6.1
IL1B[3,1]	0.7	6.3	5.7
PRF1[3,1]	1.0	6.5	5.5
APOBEC3A[1]	0.1	5.6	5.5
KIR2DL4[1]	−4.0	1.2	5.2
UNC13A[1]	−0.7	1.8	2.5

[1]Associated with treatment failure in multiple studies
[2]Implicated in parasite control
[3]Deletion or blockaid ameliorates disease in mice

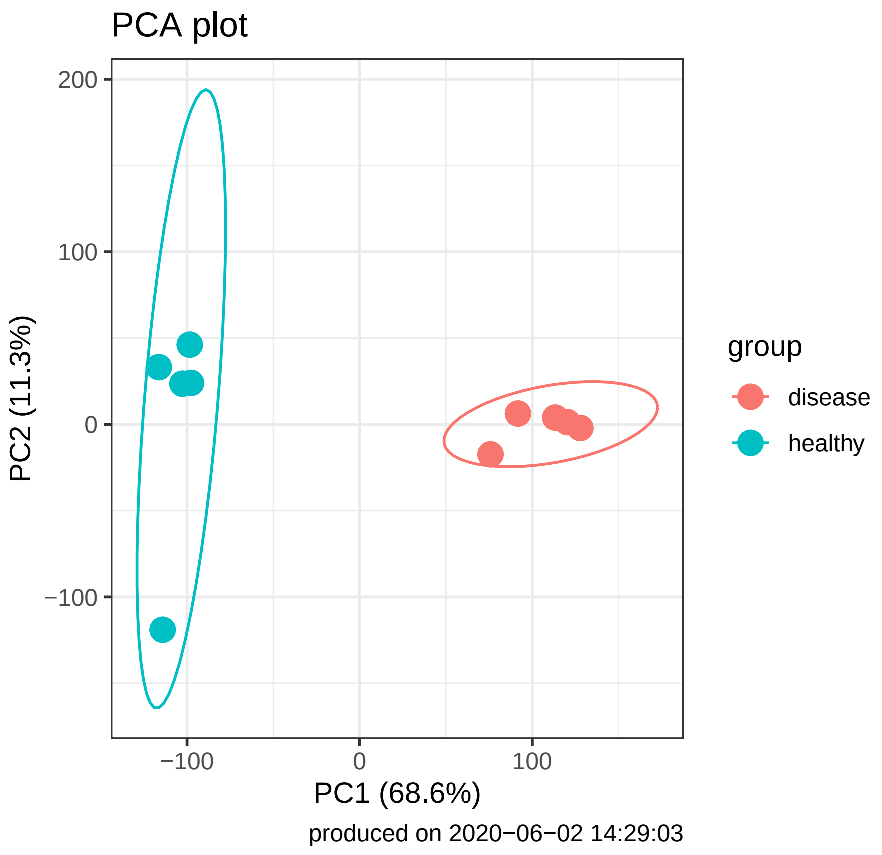
Reference: Amorim *et al*., (2019). DOI: 10.1126/scitranslmed.aar3619

The table shown below includes expression data for a selection of genes. ***

PCA plot

```r
group <- targets$group
group <- factor(group)
```

```
pca.res <- prcomp(t(log2.cpm.filtered.norm), scale.=F, retx=T)
pc.var<-pca.res$sdev^2 # sdev^2 captures these eigenvalues from the PCA result
pc.per<-round(pc.var/sum(pc.var)*100, 1)
pca.res.df <- as_tibble(pca.res$x)
ggplot(pca.res.df) +
  aes(x=PC1, y=PC2, label=sampleLabels, color = group) +
  geom_point(size=4) +
  stat_ellipse() +
  xlab(paste0("PC1 (",pc.per[1],"%",")")) +
  ylab(paste0("PC2 (",pc.per[2],"%",")")) +
  labs(title="PCA plot",
       caption=paste0("produced on ", Sys.time())) +
  coord_fixed() +
  theme_bw()
```



Volcano plot

```
library(tidyverse)
library(limma)
library(edgeR)
library(gt)

group <- factor(targets$group)
```

```r
design <- model.matrix(~0 + group)
colnames(design) <- levels(group)

v.DEGList.filtered.norm <- voom(myDGEList.filtered.norm, design, plot = FALSE)
fit <- lmFit(v.DEGList.filtered.norm, design)
contrast.matrix <- makeContrasts(infection = disease - healthy,
                                 levels=design)

fits <- contrasts.fit(fit, contrast.matrix)
ebFit <- eBayes(fits)
myTopHits <- topTable(ebFit, adjust ="BH", coef=1, number=40000, sort.by="logFC")

myTopHits.df <- myTopHits %>%
  as_tibble(rownames = "geneID")

ggplot(myTopHits.df) +
  aes(y=-log10(adj.P.Val), x=logFC, text = paste("Symbol:", geneID)) +
  geom_point(size=2) +
  geom_hline(yintercept = -log10(0.01), linetype="longdash", colour="grey", size=1) +
  geom_vline(xintercept = 1, linetype="longdash", colour="#BE684D", size=1) +
  geom_vline(xintercept = -1, linetype="longdash", colour="#2C467A", size=1) +
  #annotate("rect", xmin = 1, xmax = 12, ymin = -log10(0.01), ymax = 7.5, alpha=.2, fill="#BE684D") +
  #annotate("rect", xmin = -1, xmax = -12, ymin = -log10(0.01), ymax = 7.5, alpha=.2, fill="#2C467A") +
  labs(title="Volcano plot",
       subtitle = "Cutaneous leishmaniasis",
       caption=paste0("produced on ", Sys.time())) +
  theme_bw()
```

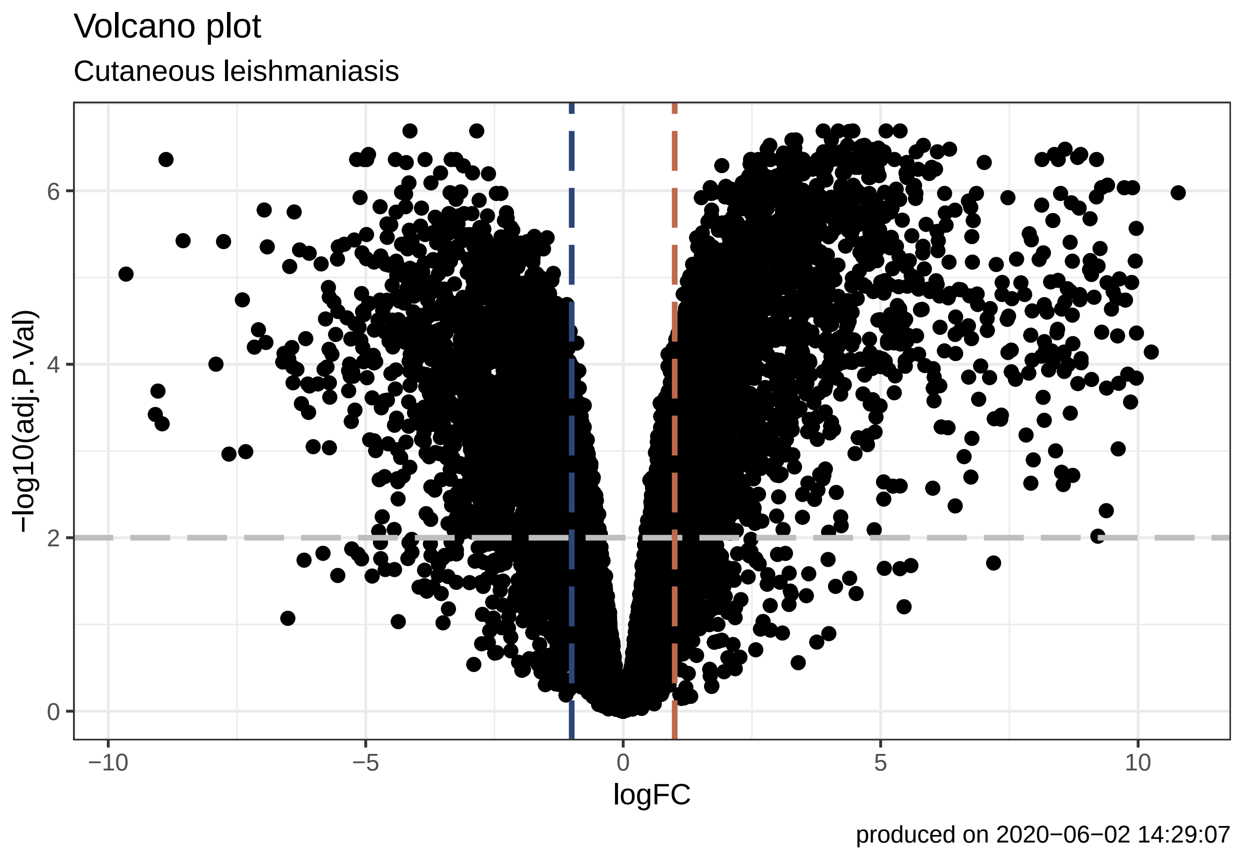


Table of DEGs

To identify differentially expressed genes, precision weights were first applied to each gene based on its mean-variance relationship using VOOM, then data was normalized using the TMM method in EdgeR. Linear modeling and bayesian stats were employed via Limma to find genes that were up- or down-regulated in leishmania patients by 4-fold or more, with a false-discovery rate (FDR) of 0.01.

```
results <- decideTests(ebFit, method="global", adjust.method="BH", p.value=0.01, lfc=2)
colnames(v.DEGList.filtered.norm$E) <- sampleLabels
diffGenes <- v.DEGList.filtered.norm$E[results[,1] !=0,]
diffGenes.df <- as_tibble(diffGenes, rownames = "geneID")
diffGenes.df %>%
  sample_n(20) %>%
  gt() %>%
  fmt_number(columns = 2:11, decimals = 2)
```

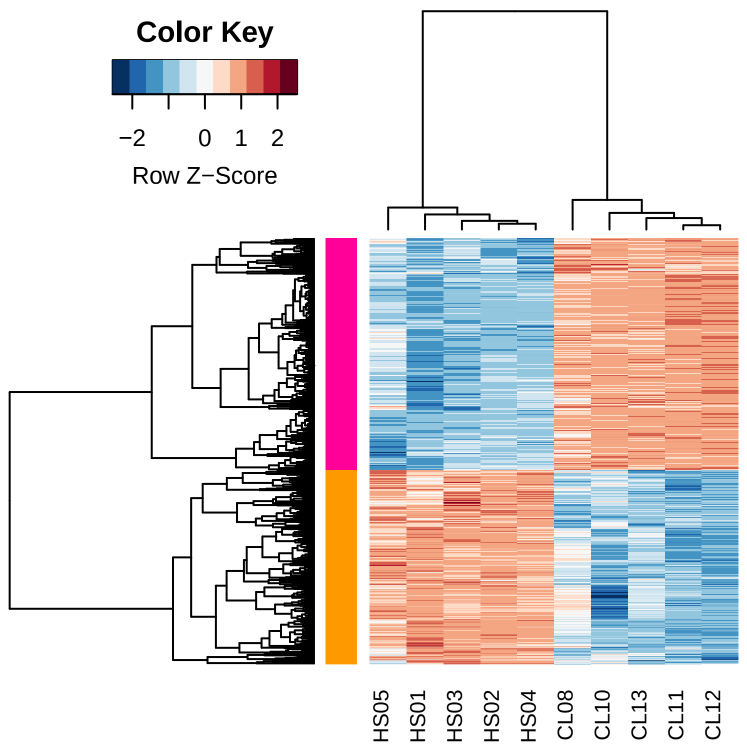
geneID	HS01	HS02	HS03	HS04	HS05	CL08	CL10	CL11	CL12	CL13
TRAC	1.99	2.78	2.39	3.22	2.20	6.31	6.14	6.78	6.80	6.03
GPR35	−0.02	0.02	0.23	−0.33	0.49	1.63	2.59	2.55	1.57	2.47
FMO1	1.74	3.19	3.52	3.37	1.44	5.93	5.81	5.77	6.29	4.76
MT1H	−1.37	0.33	−1.69	−0.99	−1.40	1.23	1.81	4.83	4.60	3.93
HSPB6	0.63	2.12	2.71	2.75	3.57	−1.11	0.39	−0.85	−1.29	−2.91
OCLN	6.98	6.65	6.56	6.35	6.24	4.46	3.48	2.53	2.53	4.06
LMOD1	4.56	6.00	6.26	6.00	5.24	3.80	4.73	1.45	2.56	2.95

CDKN2A	−2.06	−1.76	−2.12	−3.09	1.49	0.94	0.79	0.76	1.43	0.25
PCYOX1	7.63	7.92	8.07	8.15	8.10	5.95	6.18	5.02	5.13	6.06
IMPDH1P10	−1.78	−1.30	−0.63	−0.85	−2.25	1.68	1.82	2.81	2.37	1.41
ADGRL3	5.41	5.60	5.32	5.35	5.19	1.98	2.92	1.31	1.06	2.66
TRBV29-1	−4.90	−3.95	−4.96	−2.83	−2.25	0.80	0.18	0.78	0.36	0.36
IL10RA	2.96	4.37	4.61	4.27	4.46	8.47	8.31	8.53	8.93	8.56
S1PR4	−1.28	0.41	0.19	−0.36	−0.16	4.21	4.32	5.25	5.04	4.01
PCLO	4.42	4.54	5.94	4.61	2.77	1.22	0.79	−0.26	−0.50	1.08
RP11-371A22.1	5.03	4.45	3.61	4.41	3.64	4.01	0.18	0.40	0.39	3.11
RGS7BP	0.26	1.50	2.92	1.36	1.40	0.29	−0.96	−2.24	−1.85	−1.42
SP110	3.64	4.11	3.98	3.52	4.00	5.26	5.99	5.86	6.19	6.24
HIST1H2AL	3.07	2.84	1.82	2.49	2.61	3.99	4.31	5.41	5.48	4.04
PMEL	7.36	7.25	5.83	6.16	8.21	5.82	0.97	4.00	3.06	3.85

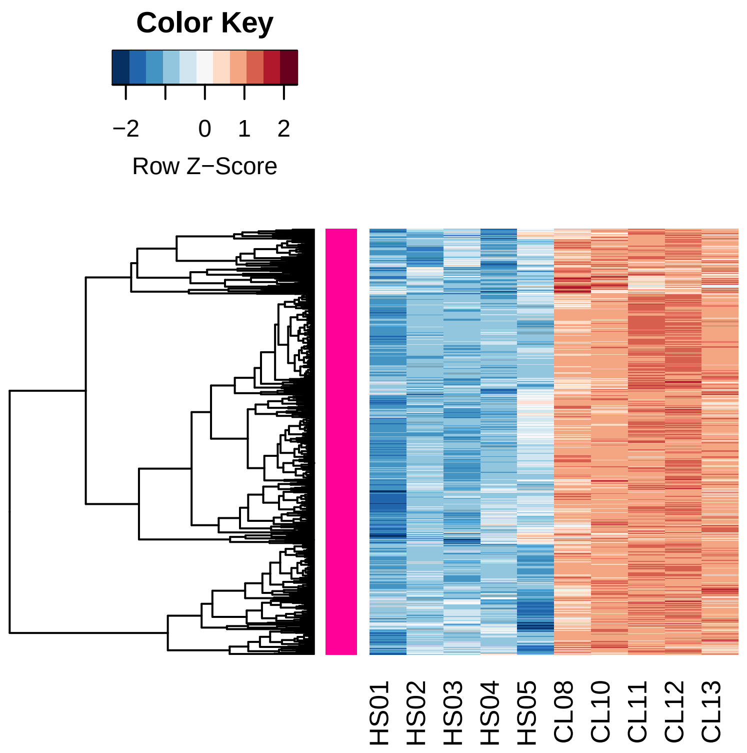
Heatmaps and modules

Pearson correlation was used to cluster **2314** differentially expressed genes, which were then represented as heatmap with the data scaled by Zscore for each row.

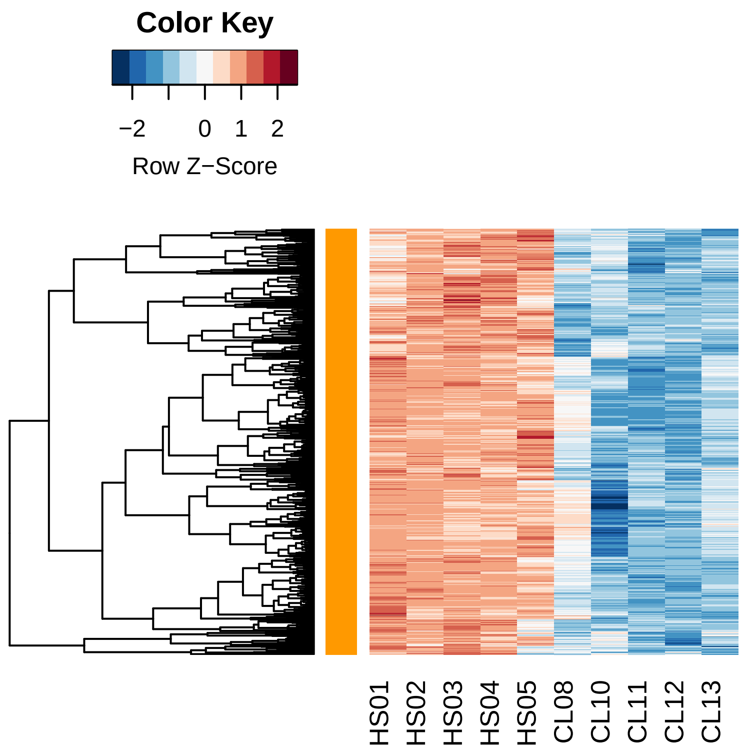
```r
library(tidyverse)
library(gplots)
library(RColorBrewer)
myheatcolors <- rev(brewer.pal(name="RdBu", n=11))
clustRows <- hclust(as.dist(1-cor(t(diffGenes), method="pearson")), method="complete") #cluster rows by
clustColumns <- hclust(as.dist(1-cor(diffGenes, method="spearman")), method="complete")
module.assign <- cutree(clustRows, k=2)
module.color <- rainbow(length(unique(module.assign)), start=0.1, end=0.9)
module.color <- module.color[as.vector(module.assign)]
heatmap.2(diffGenes,
          Rowv=as.dendrogram(clustRows),
          Colv=as.dendrogram(clustColumns),
          RowSideColors=module.color,
          col=myheatcolors, scale='row', labRow=NA,
          density.info="none", trace="none",
          cexRow=1, cexCol=1, margins=c(8,20))
```



```r
modulePick <- 2
myModule_up <- diffGenes[names(module.assign[module.assign %in% modulePick]),]
hrsub_up <- hclust(as.dist(1-cor(t(myModule_up), method="pearson")), method="complete")

heatmap.2(myModule_up,
          Rowv=as.dendrogram(hrsub_up),
          Colv=NA,
          labRow = NA,
          col=myheatcolors, scale="row",
          density.info="none", trace="none",
          RowSideColors=module.color[module.assign%in%modulePick], margins=c(8,20))
```



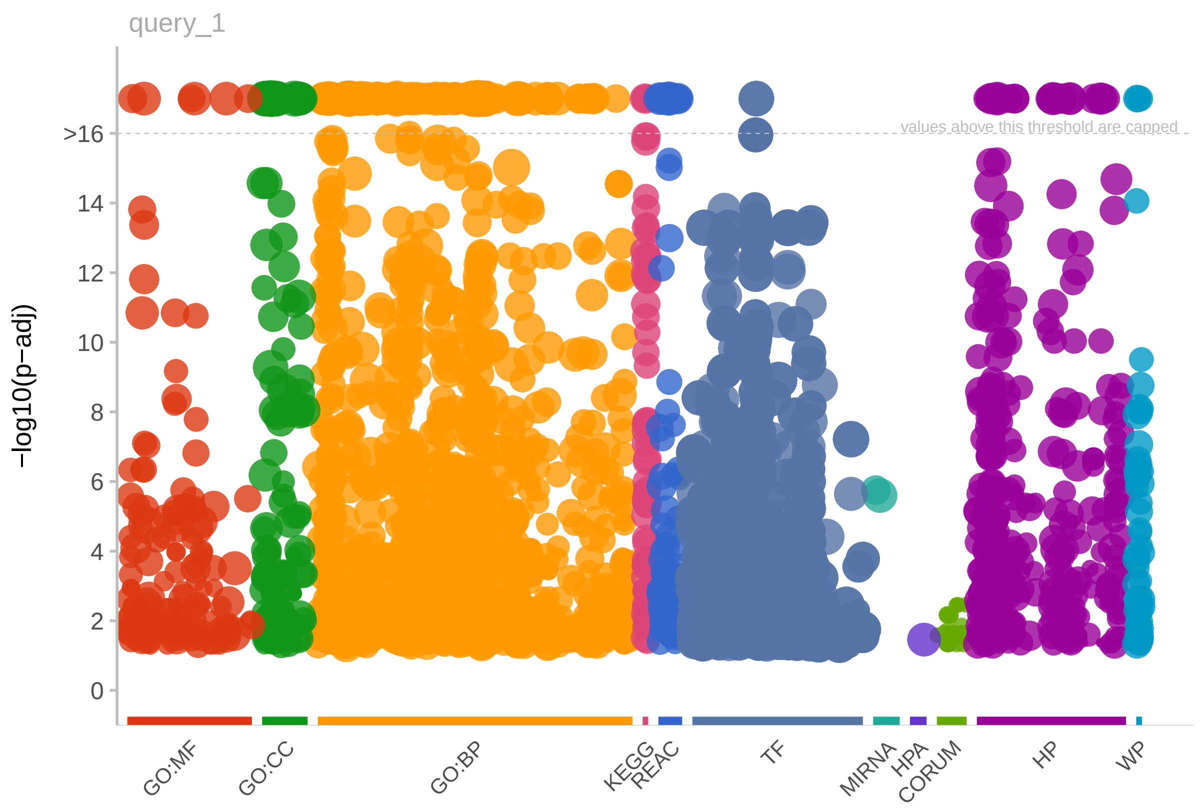
```r
modulePick <- 1
myModule_down <- diffGenes[names(module.assign[module.assign %in% modulePick]),]
hrsub_down <- hclust(as.dist(1-cor(t(myModule_down), method="pearson")), method="complete")

heatmap.2(myModule_down,
          Rowv=as.dendrogram(hrsub_down),
          Colv=NA,
          labRow = NA,
          col=myheatcolors, scale="row",
          density.info="none", trace="none",
          RowSideColors=module.color[module.assign%in%modulePick], margins=c(8,20))
```



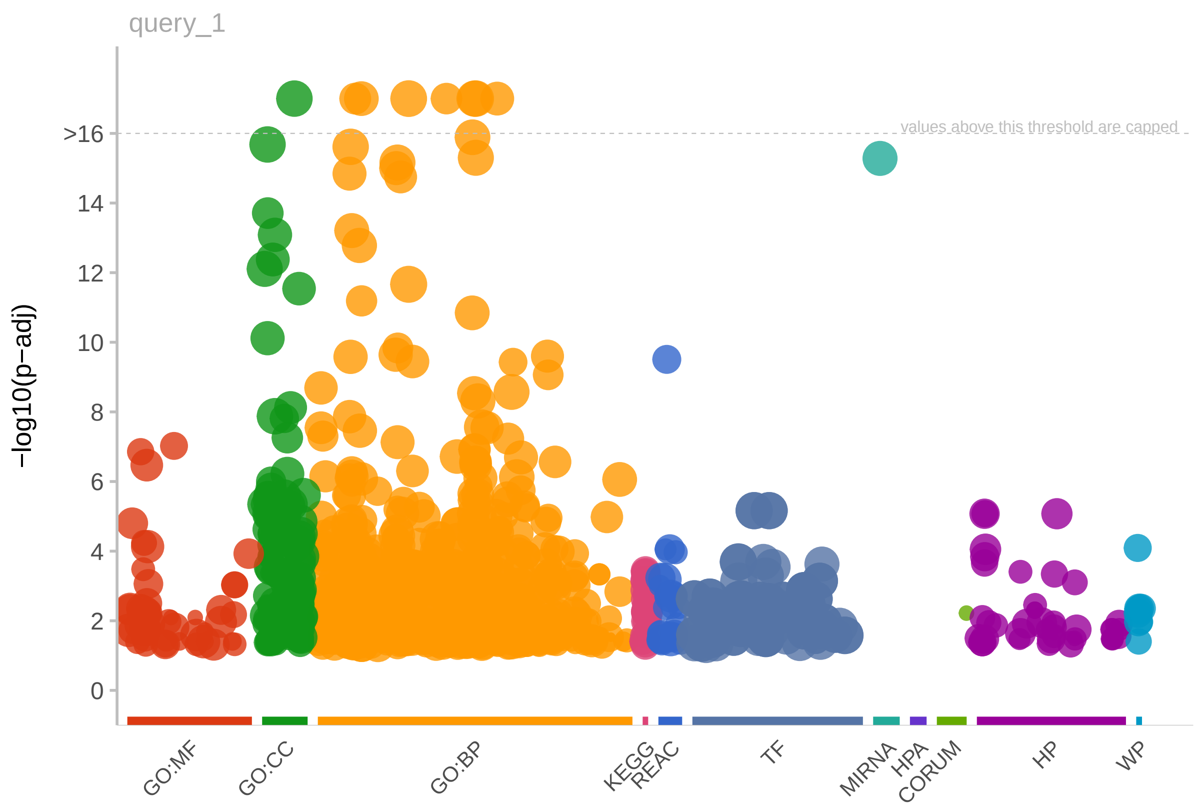
GO enrichment

GO enrichment for the 15944 genes induced by infection

```
library(tidyverse)
library(limma)
library(gplots) #f or heatmaps
library(GSEABase) # functions and methods for Gene Set Enrichment Analysis
library(Biobase) # base functions for bioconductor; required by GSEABase
library(GSVA) # GSVA, a non-parametric and unsupervised method for estimating variation of gene set enr
library(gprofiler2) # tools for accessing the GO enrichment results using g:Profiler web resources
library(clusterProfiler) # provides a suite of tools for functional enrichment analysis
library(msigdbr) # access to msigdb collections directly within R
library(enrichplot) # great for making the standard GSEA enrichment plots
gost.res_up <- gost(rownames(myModule_up), organism = "hsapiens", correction_method = "fdr")
gostplot(gost.res_up, interactive = F, capped = T) # set interactive=FALSE to get plot for publications
```



```
gost.res_down <- gost(rownames(myModule_down), organism = "hsapiens", correction_method = "fdr")
gostplot(gost.res_down, interactive = F, capped = T) # set interactive=FALSE to get plot for publicatio
```



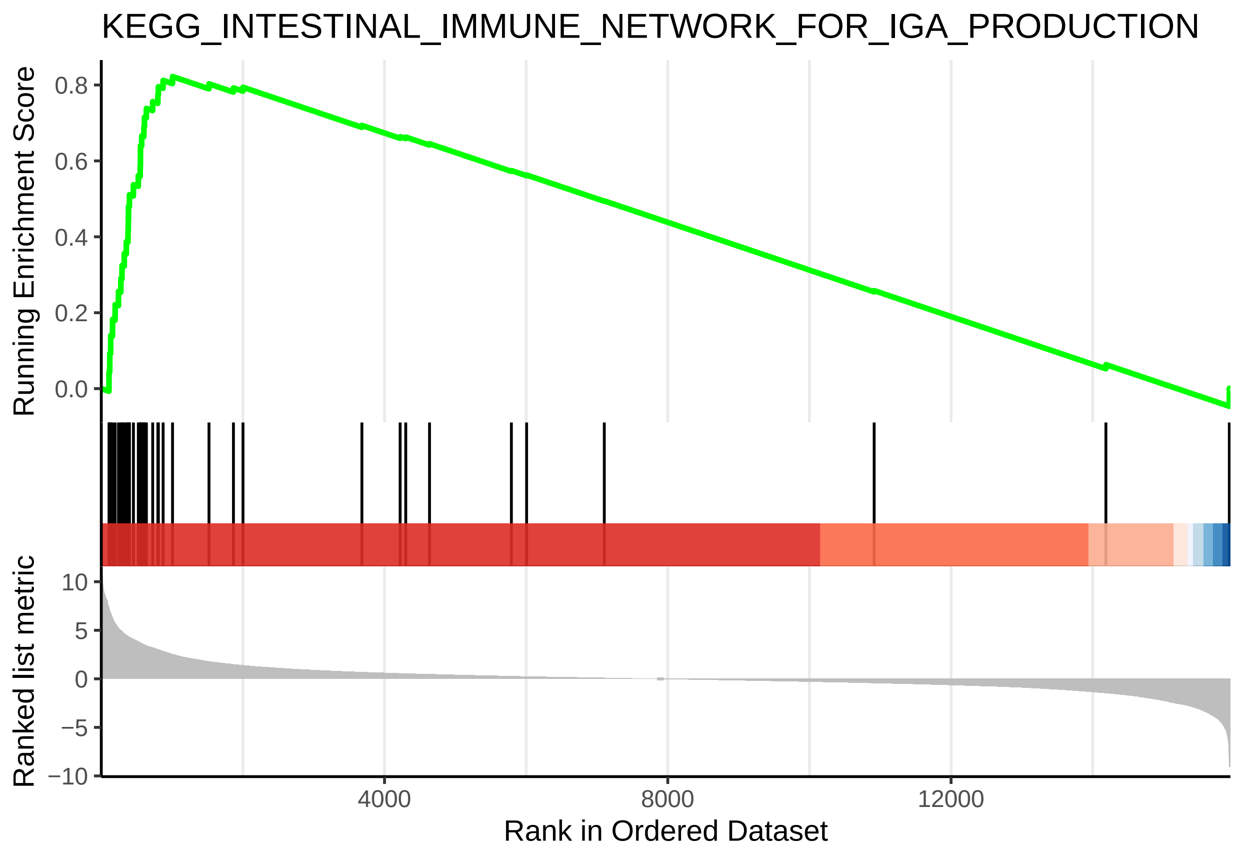
GSEA

```r
hs_gsea_c2 <- msigdbr(species = "Homo sapiens", # change depending on species your data came from
                      category = "C2") %>% # choose your msigdb collection of interest
  dplyr::select(gs_name, gene_symbol) # just get the columns corresponding to signature name and gene s

# Now that you have your msigdb collections ready, prepare your data
# grab the dataframe you made in step3 script
# Pull out just the columns corresponding to gene symbols and LogFC for at least one pairwise compariso
mydata.df.sub <- dplyr::select(mydata.df, geneID, LogFC)
mydata.gsea <- mydata.df.sub$LogFC
names(mydata.gsea) <- as.character(mydata.df.sub$geneID)
mydata.gsea <- sort(mydata.gsea, decreasing = TRUE)

# run GSEA using the 'GSEA' function from clusterProfiler
myGSEA.res <- GSEA(mydata.gsea, TERM2GENE=hs_gsea_c2, verbose=FALSE)
myGSEA.df <- as_tibble(myGSEA.res@result)

# view results as an interactive table
gt(head(myGSEA.df))
```

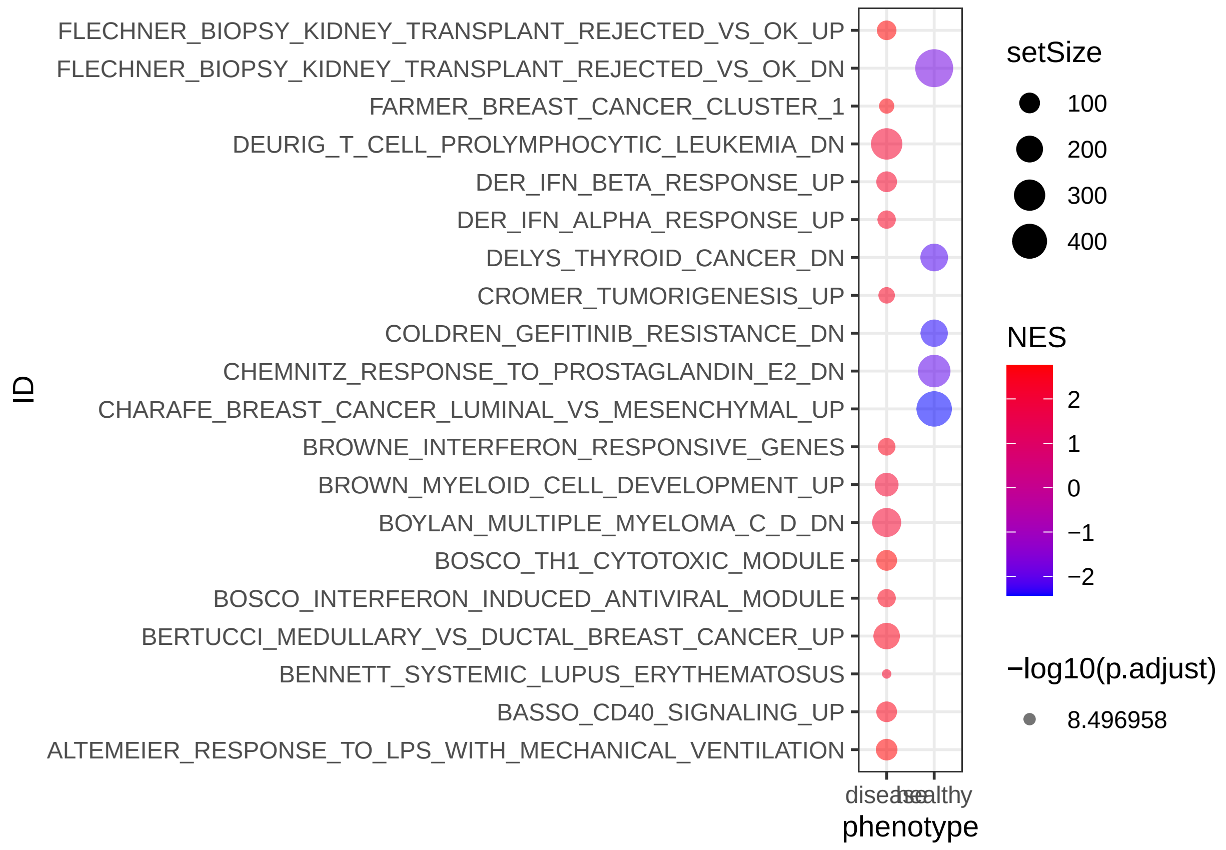
ID	Description
ALTEMEIER_RESPONSE_TO_LPS_WITH_MECHANICAL_VENTILATION	ALTEMEIER_RESPONSE_TO_LI
BASSO_CD40_SIGNALING_UP	BASSO_CD40_SIGNALING_UP
BENNETT_SYSTEMIC_LUPUS_ERYTHEMATOSUS	BENNETT_SYSTEMIC_LUPUS_I

BERTUCCI_MEDULLARY_VS_DUCTAL_BREAST_CANCER_UP	BERTUCCI_MEDULLARY_VS_D
BOSCO_INTERFERON_INDUCED_ANTIVIRAL_MODULE	BOSCO_INTERFERON_INDUCE
BOSCO_TH1_CYTOTOXIC_MODULE	BOSCO_TH1_CYTOTOXIC_MOI

```r
# create enrichment plots using the enrichplot package
gseaplot2(myGSEA.res,
          geneSetID = 47, #can choose multiple signatures to overlay in this plot
          pvalue_table = FALSE, #can set this to FALSE for a cleaner plot
          title = myGSEA.res$Description[47]) #can also turn off this title
```



```r
# add a variable to this result that matches enrichment direction with phenotype
myGSEA.df <- myGSEA.df %>%
  mutate(phenotype = case_when(
    NES > 0 ~ "disease",
    NES < 0 ~ "healthy"))

# create 'bubble plot' to summarize y signatures across x phenotypes
ggplot(myGSEA.df[1:20,], aes(x=phenotype, y=ID)) +
  geom_point(aes(size=setSize, color = NES, alpha=-log10(p.adjust))) +
  scale_color_gradient(low="blue", high="red") +
  theme_bw()
```



Conclusions

Describe the results in your own words. Some things to think about:

- What are the key takeaways from the analysis?
- What types of analyses would you want to do next?
- Based on your analysis, are there any wet-lab experiments would might priortize?
- How could you expand on or otherwise enhance this Rmarkdown report?

Session info

The output from running 'sessionInfo' is shown below and details all packages and version necessary to reproduce the results in this report.

```r
sessionInfo()
```

```
## R version 4.0.0 (2020-04-24)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19041)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=French_France.1252  LC_CTYPE=French_France.1252    LC_MONETARY=French_France.1252 LC_
##
## attached base packages:
```

```
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] enrichplot_1.8.1         msigdbr_7.1.1            clusterProfiler_3.16.0   gprofiler2_0.1.9
##  [7] graph_1.66.0             annotate_1.66.0          XML_3.99-0.3             RColorBrewer_1.1-
## [13] gt_0.2.1                 DT_0.13                  cowplot_1.0.0            matrixStats_0.56.
## [19] EnsDb.Hsapiens.v86_2.99.0 ensembldb_2.12.1        AnnotationFilter_1.12.0  GenomicFeatures_1
## [25] GenomicRanges_1.40.0     GenomeInfoDb_1.24.0      IRanges_2.22.1           S4Vectors_0.26.1
## [31] forcats_0.5.0            stringr_1.4.0            dplyr_0.8.5              purrr_0.3.4
## [37] tibble_3.0.1             ggplot2_3.3.0            tidyverse_1.3.0          knitr_1.28
##
## loaded via a namespace (and not attached):
##   [1] tidyselect_1.1.0            RSQLite_2.2.0               htmlwidgets_1.5.1           grid_4.0.0
##   [7] munsell_0.5.0               withr_2.2.0                 colorspace_1.4-2            GOSemSim_2
##  [13] labeling_0.3                urltools_1.7.3              GenomeInfoDbData_1.2.3      polyclip_1
##  [19] rhdf5_2.32.0                downloader_0.4              vctrs_0.3.0                 generics_0
##  [25] R6_2.4.1                    graphlayouts_0.7.0          locfit_1.5-9.4              bitops_1.0
##  [31] DelayedArray_0.14.0         assertthat_0.2.1            promises_1.1.0              scales_1.1
##  [37] tidygraph_1.2.0             rlang_0.4.6                 splines_4.0.0               rtracklaye
##  [43] broom_0.5.6                 europepmc_0.3               BiocManager_1.30.10         yaml_2.2.1
##  [49] crosstalk_1.1.0.1           backports_1.1.7             httpuv_1.5.2                qvalue_2.2
##  [55] ellipsis_0.3.1              ggridges_0.5.2              Rcpp_1.0.4.6                plyr_1.8.6
##  [61] RCurl_1.98-1.2              prettyunits_1.1.1           openssl_1.4.1               viridis_0.
##  [67] ggrepel_0.8.2               fs_1.4.1                    magrittr_1.5                data.table
##  [73] reprex_0.3.0                ProtGenerics_1.20.0         hms_0.5.3                   mime_0.9.1
##  [79] readxl_1.3.1                gridExtra_2.3               compiler_4.0.0              biomaRt_2.
##  [85] htmltools_0.4.0             later_1.0.0                 snow_0.4-3                  lubridate_
##  [91] dbplyr_1.4.3                MASS_7.3-51.6               rappdirs_0.3.1              Matrix_1.3
##  [97] igraph_1.2.5                pkgconfig_2.0.3             rvcheck_0.1.8               GenomicAli
## [103] rvest_0.3.5                 digest_0.6.25               Biostrings_2.56.0           cellranger
## [109] commonmark_1.7              shiny_1.4.0.2               Rsamtools_2.4.0             gtools_3.8
## [115] jsonlite_1.6.1              Rhdf5lib_1.10.0             viridisLite_0.3.0           askpass_1.
## [121] lattice_0.20-41             fastmap_1.0.1               httr_1.4.1                  GO.db_3.11
## [127] bit_1.1-15.2                ggforce_0.3.1               stringi_1.4.6               blob_1.2.1
```