

DAAD RNA-seq course - lessons 3-4

Serhiy Naumenko

2024-08-09

Contents

Overview	2
Load annotations	3
Load Counts	4
Cleanup and load metadata	5
Run DESeq2	6
Convenience functions	7
Sample-level QC analysis	8
PCA - treatment	8
PCA - experiment	9
Clustering using top 1000 variable genes	10
PCA: Controls	12
PCA: TAA and CCL4	13
PCA: TAA and CCL4 - treatment	14
PCA: TAA experiment	15
PCA: CCL4 experiment	16
DE in TAA	17
Visualization - Gene example	18
Heatmaps	19
Functional analysis	21
Create background dataset for hypergeometric testing using all genes tested for significance in the results	21
R session	24

Overview

- Petrenko2024 RNA-seq experiment for reanalysis

Load annotations

```
annotation_file <- "../99_technical/ensembl112_mm10_annotations.txt"
gene_symbol_file <- "../99_technical/gene_symbol.txt"
if (file.exists(annotation_file)){
  mmdb <- read_tsv(annotation_file)
  gene_symbol <- read_tsv(gene_symbol_file)
}else{
  # Connect to AnnotationHub
  ah <- AnnotationHub()
  # Query AnnotationHub
  mm_ens <- query(ah, c("Mus musculus", "EnsDb"))

  # Get Ensembl 112
  # AH116909
  mm_ens <- mm_ens[["AH116909"]]

  # Extract gene-level information
  txdb <- transcripts(mm_ens,
    return.type = "data.frame") %>%
  dplyr::select(tx_id, gene_id)

  genedb <- genes(mm_ens,
    return.type = "data.frame") %>%
  dplyr::select(gene_id, gene_name, symbol)

  gene_symbol <- genedb %>% dplyr::select(gene_id, symbol)
  write_tsv(gene_symbol, gene_symbol_file)

  mmdb <- inner_join(txdb, genedb)
  write.table(mmdb,
    file = annotation_file ,
    sep = "\t",
    row.names = F,
    quote = F)
}
tx2gene <- mmdb[, c("tx_id", "gene_id")]
```

Load Counts

```
# raw counts downloaded from
# https://www.ebi.ac.uk/biostudies/arrayexpress/studies/E-MTAB-13804

# setwd("02_classes/03_rnaseq_intro_part1/")

protein_coding_genes <- read_csv("../..99_technical/ensembl_w_description.mouse.protein_coding.csv")

counts_csv <- "../..01_data/counts.csv"
counts_tpm_csv <- "../..01_data/count_matrix_tpm.csv"
if (file.exists(counts_csv)){
  counts_prepared <- read_csv(counts_csv)
  counts_tpm <- read_csv(counts_tpm_csv)
}else{
  counts_raw_csv <- "../..01_data/count_matrix_raw.csv"
  counts_raw <- read_csv(counts_raw_csv)
  colnames(counts_raw)[1] <- "gene_name"
  # counts_tpm <- read_csv(counts_tpm_csv)
  # use Ensembl_Gene_id
  # remove genes with NA
  # filter protein coding genes

  counts_prepared <- counts_raw %>% left_join(gene_symbol, by = c("gene_name" = "symbol")) %>%
    dplyr::select(-gene_name) %>% drop_na(gene_id) %>%
    semi_join(protein_coding_genes, by = c("gene_id" = "ensembl_gene_id")) %>%
    relocate(gene_id) %>% rename(ensembl_gene_id = gene_id)

  gene_length <- read_tsv("../..99_technical/GC_lengths.tsv")
  counts <- counts_prepared %>% arrange(ensembl_gene_id)
  gene_ids <- intersect(counts$ensembl_gene_id, gene_length$ensembl_gene_id)

  v_len <- gene_length %>% dplyr::filter(ensembl_gene_id %in% gene_ids)
  counts_prepared <- counts %>% dplyr::filter(ensembl_gene_id %in% gene_ids)

  write_csv(counts_prepared, counts_csv)
  counts <- counts %>% column_to_rownames("ensembl_gene_id")

  x <- counts / v_len$Length
  counts_tpm <- t(t(x) * 1e6 / colSums(x)) %>% as.data.frame() %>% round(2) %>%
    rownames_to_column("ensembl_gene_id") %>% left_join(gene_symbol,
                                                         by = c("ensembl_gene_id" = "gene_id")) %>%
    write_csv(counts_tpm_csv)
}
counts <- counts_prepared %>% column_to_rownames(var = "ensembl_gene_id")
```

Cleanup and load metadata

```
# Load the data and metadata
# remove duplicate rows
metadata_csv <- "../01_data/metadata.csv"
if (file.exists(metadata_csv)){
  metadata <- read_csv(metadata_csv)
}else{
  metadata_raw <- read_tsv("../01_data/E-MTAB-13804.sdrf.txt") %>%
    dplyr::select(-any_of(c("Scan Name", "Comment[SUBMITTED_FILE_NAME]",
                          "Comment[ENA_RUN]", "Comment[FASTQ_URI]"))) %>% distinct() %>%
    dplyr::rename(sample_id = `Source Name`)
  colnames(metadata_raw)[7] <- "age"
  colnames(metadata_raw)[44] <- "stimulus1"
  colnames(metadata_raw)[46] <- "factor_value"

  metadata_raw$stimulus <- str_replace_all(metadata_raw$stimulus1, "control \\(olive oil\\)", "olive_")
  metadata_raw$stimulus <- str_replace_all(metadata_raw$stimulus, "control \\(saline\\)", "saline")
  metadata_raw$stimulus <- str_replace_all(metadata_raw$stimulus, "carbon tetrachloride", "carbon_tet")

  metadata <- metadata_raw %>% separate(factor_value, sep = "_",
                                       into = c("experiment", "treatment")) %>%
    dplyr::select(sample_id, experiment, treatment, stimulus)

  write_csv(metadata, metadata_csv)
}

metadata <- metadata %>% column_to_rownames(var = "sample_id")
```

Run DESeq2

estimating size factors
estimating dispersions
gene-wise dispersion estimates
mean-dispersion relationship
final dispersion estimates
fitting model and testing

- Estimating size factors and count normalization
- Gene-wise dispersions
- Mean-dispersion(variance) relationship and the Negative Binomial Model
- Model fitting and hypothesis testing

```
## Create DESeq2Dataset object
dds_file <- "../data/dds.RDS"

if (file.exists(dds_file)){
  dds <- readRDS(dds_file)
}else{
  dds <- DESeqDataSetFromMatrix(countData = counts,
                                colData = metadata,
                                design = ~treatment)

  # subset protein-coding genes
  pc_genes <- intersect(protein_coding_genes$ensembl_gene_id, row.names(dds))
  dds <- dds[pc_genes,]
  # 9509 genes left
  keep <- rowMeans(counts(dds)) >= 100
  dds <- dds[keep, ]

  # Run DESeq2
  dds <- DESeq(dds)
  saveRDS(dds, dds_file)
}
```

Convenience functions

```
# return mean counts for a group of sample in a column
get_counts_for_samples <- function(ctpm, samples, column_name){
  tpm_counts <- ctpm %>%
    column_to_rownames("ensembl_gene_id") %>%
    dplyr::select(any_of(samples)) %>%
    rowMeans() %>%
    as.data.frame() %>%
    round(2) %>%
    rownames_to_column("ensembl_gene_id")

  colnames(tpm_counts) <- c("ensembl_gene_id", "tpm")

  tpm_counts <- tpm_counts %>%
    dplyr::mutate("{column_name}" := round(tpm, 2)) %>%
    dplyr::select(-tpm)

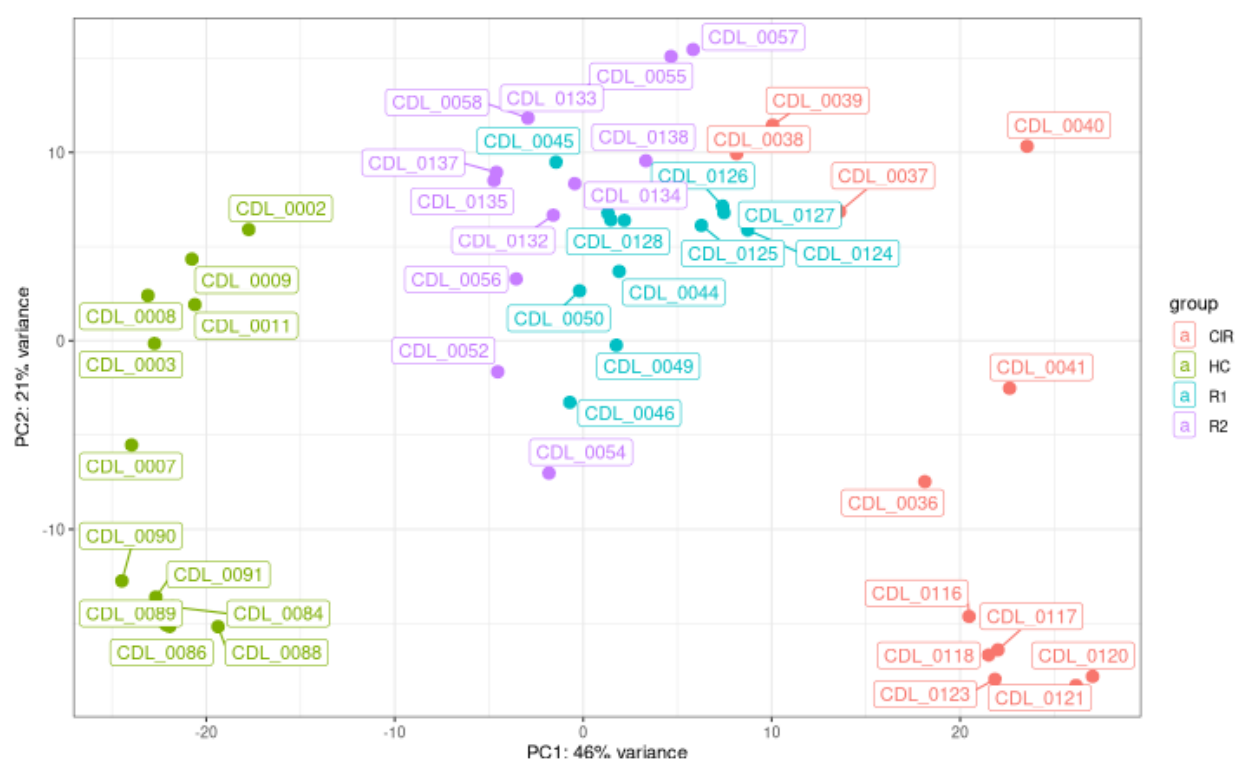
  return(tpm_counts)
}

# get rid of excess precision
comb_de_result_table <- function(results){
  results <- results %>%
    mutate(baseMean = round(baseMean, 2),
           log2FoldChange = round(log2FoldChange, 2),
           lfcSE = round(lfcSE, 2),
           stat = round(stat, 2),
           pvalue = format(pvalue, scientific = TRUE, digits = 2),
           padj = format(padj, scientific = TRUE, digits = 2))
  return(results)
}
```

Sample-level QC analysis

PCA - treatment

```
plotPCA(rld, intgroup = c("treatment")) +  
  geom_label_repel(aes(label = name)) +  
  theme_bw()
```



PCA - experiment

```
plotPCA(rld, intgroup = c("experiment")) + geom_label_repel(aes(label = name)) + theme_bw()
```



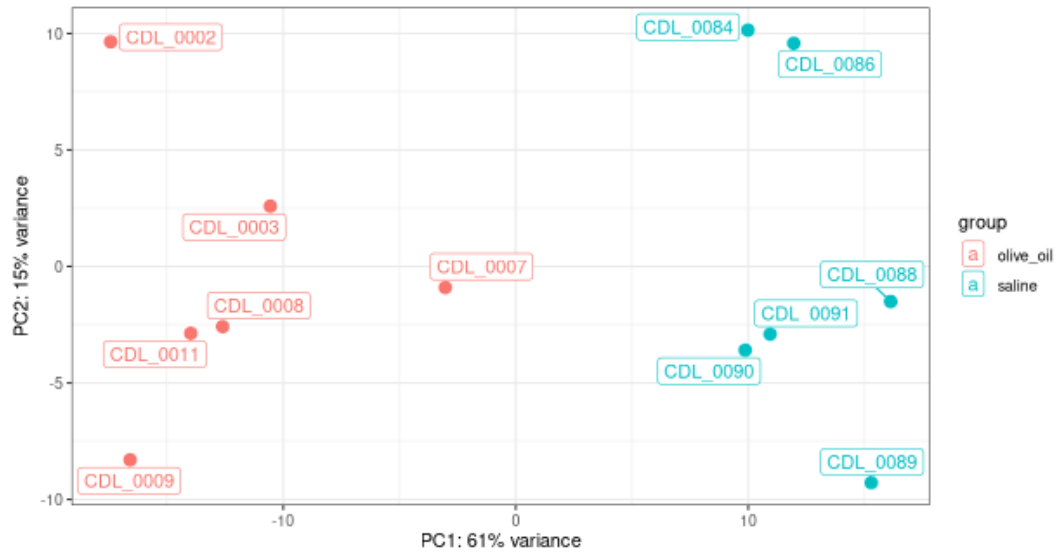
Clustering using top 1000 variable genes

```
rv <- rowVars(rld_mat)
rv <- order(rv, decreasing = TRUE) %>% head(1000)
rld_mat_1000 <- rld_mat[rv,]
annotation <- metadata[, c("experiment", "treatment")]

# Change colors
heat.colors <- brewer.pal(6, "Blues")
rld_cor <- cor(rld_mat_1000)
# Plot heatmap
pheatmap(rld_cor,
          annotation = annotation,
          border = NA,
          fontsize = 20)
```


PCA: Controls

```
rld.sub <- rld[ , rld$stimulus %in% c("saline", "olive_oil") ]  
plotPCA(rld.sub, intgroup = c("stimulus")) +  
  geom_label_repel(aes(label = name)) +  
  theme_bw()
```



PCA: TAA and CCL4

```
rld.sub <- rld[ , rld$stimulus %in% c("carbon_tetrachloride", "thioacetamide")]  
plotPCA(rld.sub, intgroup = c("stimulus")) +  
  geom_label_repel(aes(label = name)) +  
  theme_bw()
```



PCA: TAA and CCL4 - treatment

```
rld.sub <- rld[ , rld$stimulus %in% c("carbon_tetrachloride", "thioacetamide")]
plotPCA(rld.sub, intgroup = c("treatment", "stimulus")) +
  geom_label_repel(aes(label = name)) +
  theme_bw()
```



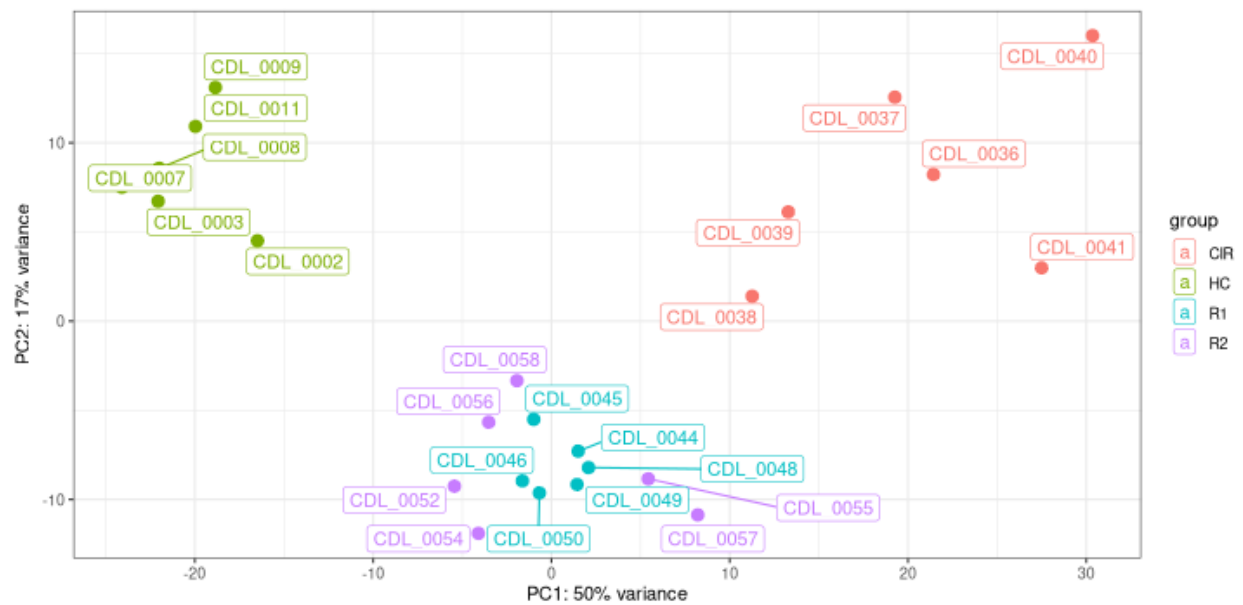
PCA: TAA experiment

```
rld.sub <- rld[ , rld$experiment %in% c("TAA")]  
plotPCA(rld.sub, intgroup = c("treatment")) +  
  geom_label_repel(aes(label = name)) + theme_bw(base_size = 15)
```



PCA: CCL4 experiment

```
rld.sub <- rld[, rld$experiment %in% c("CCL4")]  
plotPCA(rld.sub, intgroup = c("treatment")) +  
  geom_label_repel(aes(label = name)) +  
  theme_bw()
```



DE in TAA

```
ddsTAA <- subset(dds, select = colData(dds)$experiment == "TAA")
ddsTAA <- subset(ddsTAA, select = colData(ddsTAA)$treatment == "HC" | colData(ddsTAA)$treatment == "CIR")
ddsTAA$treatment <- droplevels(ddsTAA$treatment)
ddsTAA$treatment <- relevel(ddsTAA$treatment, ref = "HC")

contrast <- c("treatment", "CIR", "HC")
ddsTAA <- DESeq(ddsTAA)

resTreatment <- results(dds, contrast = contrast, alpha = 0.05)
length(which(resTreatment$padj < 0.05))

## [1] 5447

# Add annotations
resTreatment_tb <- resTreatment %>%
  data.frame() %>%
  rownames_to_column(var = "gene") %>%
  as_tibble() %>%
  left_join(gene_symbol, by = c("gene" = "gene_id"))

resTreatment_tb_significant <- dplyr::filter(resTreatment_tb, padj < 0.05) %>%
  dplyr::filter(abs(log2FoldChange) > 1) %>%
  comb_de_result_table()

write_csv(resTreatment_tb_significant, "../03_outputs/T4.TAA_results.csv")

samples_control <- metadata %>% rownames_to_column("sample") %>%
  dplyr::filter(experiment == "TAA" & treatment == "HC") %>% pull(sample)

counts_tpm$symbol <- NULL
tpm_control <- get_counts_for_samples(counts_tpm, samples_control, "HC_mean_tpm")

samples_effect <- metadata %>% dplyr::filter(experiment == "TAA" & treatment == "CIR") %>% row.names()
tpm_effect <- get_counts_for_samples(counts_tpm, samples_effect, "CIR_tpm")

tpm_counts <- tpm_effect %>%
  left_join(tpm_control,
    by = c("ensembl_gene_id" = "ensembl_gene_id"))

resTreatment_tb_significant <- resTreatment_tb_significant %>%
  left_join(tpm_counts, by = c("gene" = "ensembl_gene_id")) %>%
  arrange(log2FoldChange)

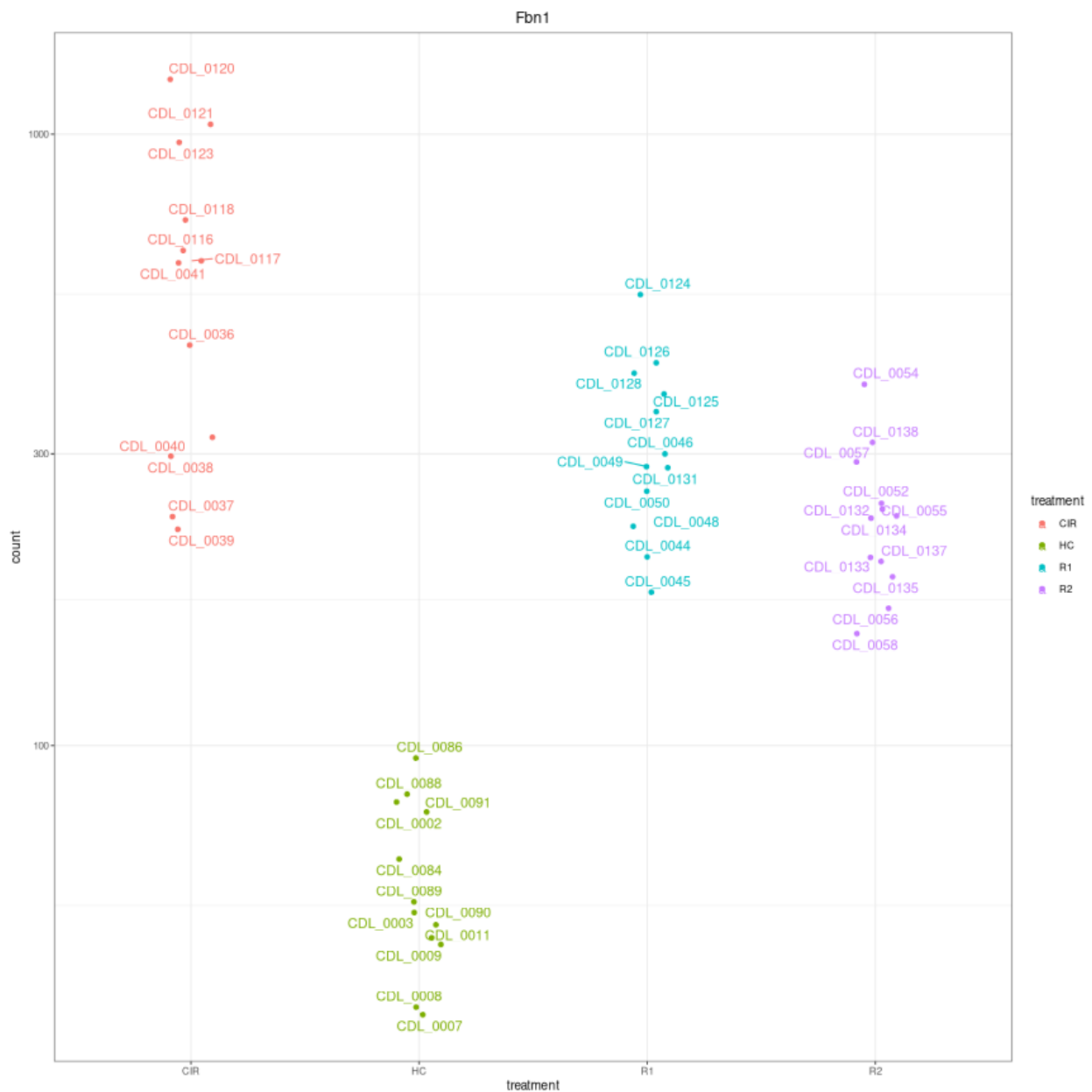
write_xlsx(list(T4.TAA_results = resTreatment_tb_significant),
  "../03_outputs/T4.DE_TAA.xlsx")

# Separate into up and down-regulated gene sets
sigTreatment_up <- rownames(resTreatment)[which(resTreatment$padj < 0.01 & resTreatment$log2FoldChange > 1)]
sigTreatment_down <- rownames(resTreatment)[which(resTreatment$padj < 0.01 & resTreatment$log2FoldChange < -1)]
```

Visualization - Gene example

```
d <- plotCounts(dds,
  gene = "ENSMUSG00000027204",
  intgroup = "treatment",
  returnData = TRUE)

ggplot(d, aes(x = treatment, y = count, color = treatment)) +
  geom_point(position = position_jitter(w = 0.1, h = 0)) +
  geom_text_repel(aes(label = rownames(d))) +
  theme_bw(base_size = 10) +
  ggtitle("Fbn1") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_y_log10()
```



Heatmaps

```
# Create a matrix of normalized expression
sig_up <- resTreatment_tb_significant %>% arrange(-log2FoldChange) %>% head(50) %>% pull(gene)
sig_down <- resTreatment_tb_significant %>% arrange(log2FoldChange) %>% head(50) %>% pull(gene)
sig <- c(sig_up, sig_down)

row_annotation <- gene_symbol %>%
  as_tibble() %>%
  dplyr::filter(gene_id %in% sig)

plotmat <- counts_tpm %>% column_to_rownames("ensembl_gene_id") %>%
  dplyr::select(any_of(c(samples_control, samples_effect)))

plotmat <- plotmat[c(sig_up, sig_down),] %>% as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id") %>%
  left_join(gene_symbol, by = c("ensembl_gene_id" = "gene_id")) %>%
  drop_na(symbol)

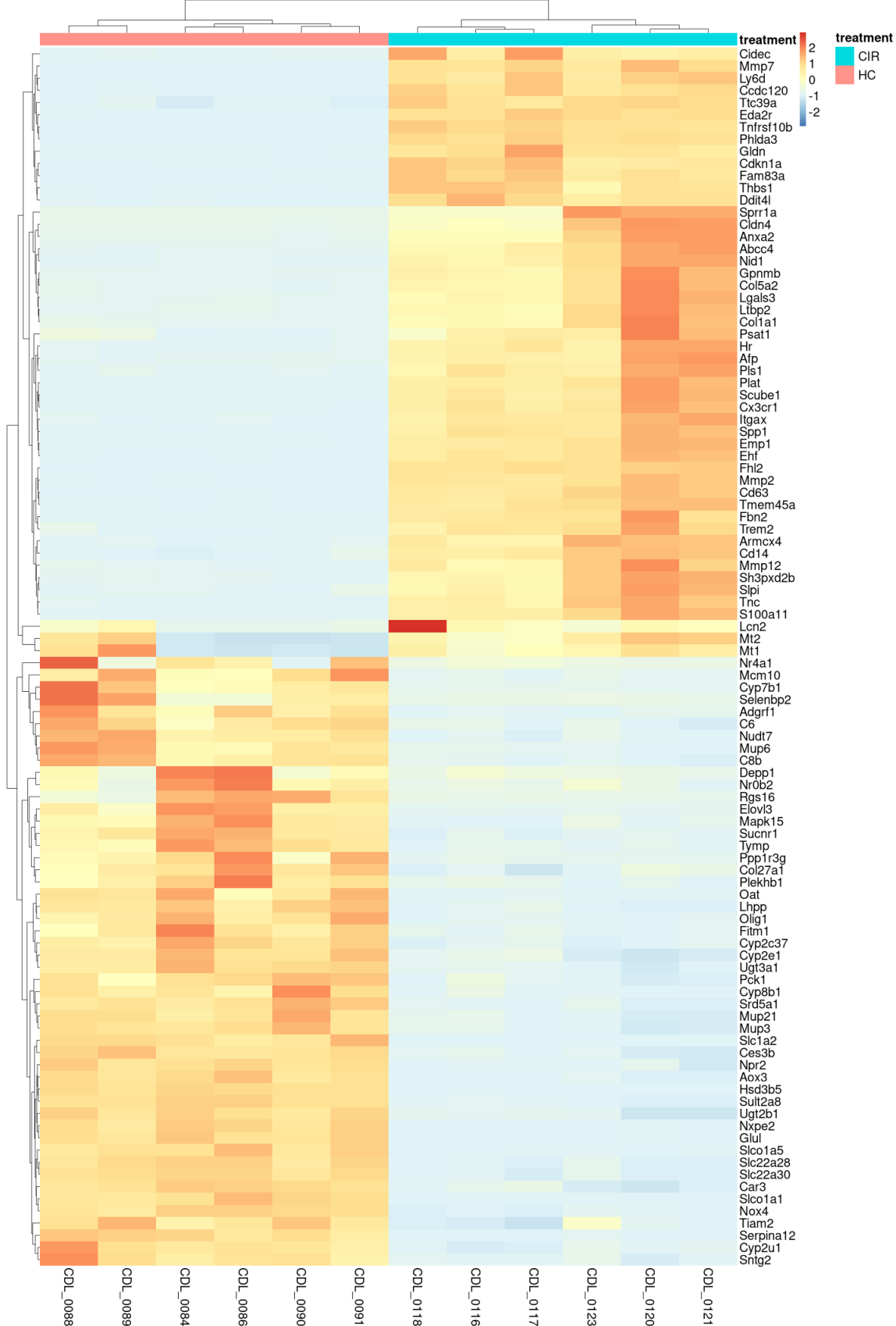
plotmat$ensembl_gene_id <- NULL

plotmat <- plotmat %>% column_to_rownames(var = "symbol") %>% as.matrix()

# Color palette
heat.colors <- brewer.pal(6, "YlOrRd")

# Plot heatmap
# color = heat.colors,
pheatmap(plotmat,
  scale = "row",
  show_rownames = TRUE,
  border = FALSE,
  annotation = metadata[, c("treatment"), drop = FALSE],
  main = "Top 50 Up- and Down- regulated genes in treatment: CIR vs HC",
  fontsize = 20)
```

Top 50 Up- and Down-regulated genes in treatment: CIR vs HC



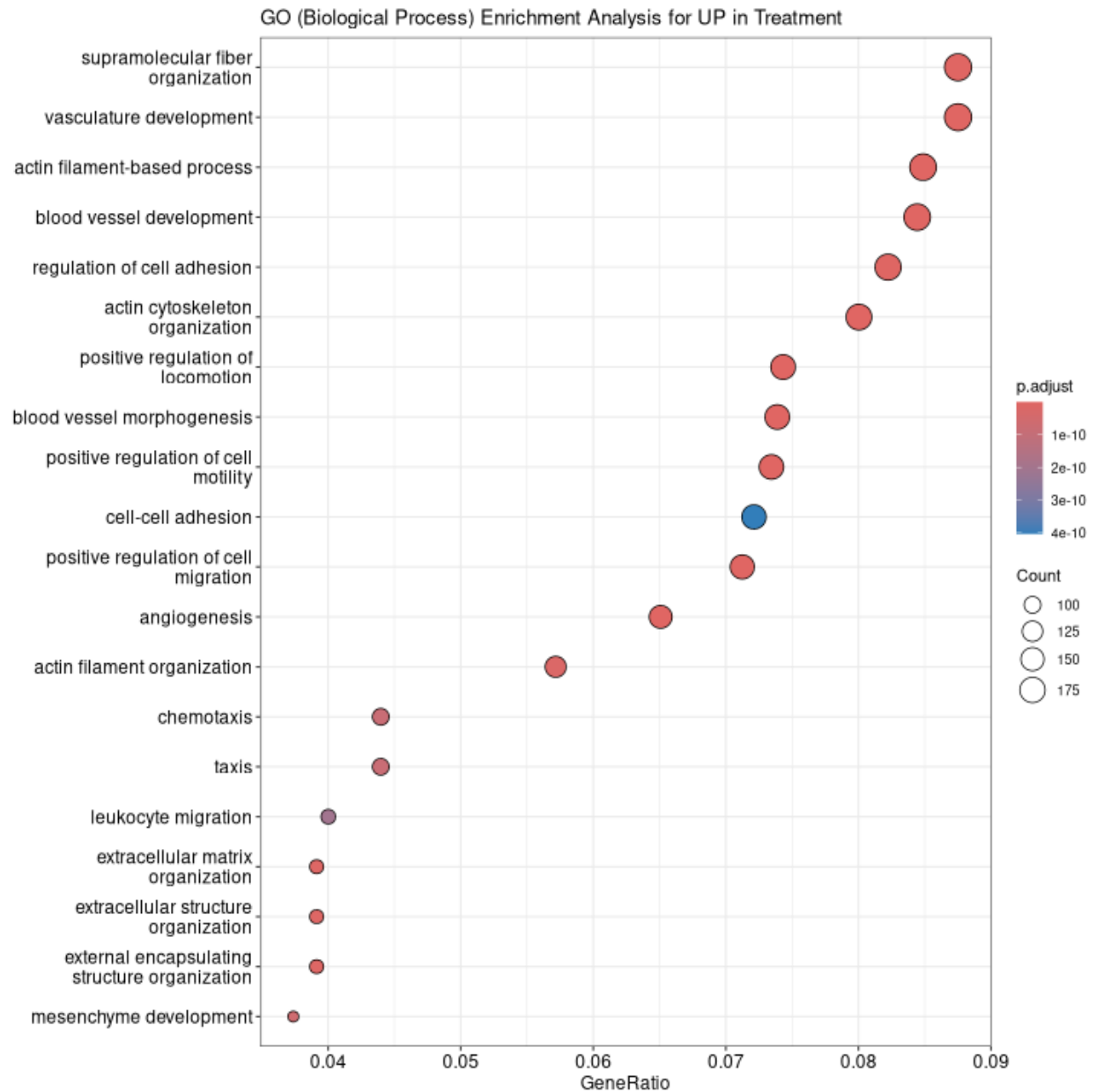
Functional analysis

Create background dataset for hypergeometric testing using all genes tested for significance in the results

```
bg_genes <- resTreatment_tb$gene

## Run GO enrichment analysis
compGO <- enrichGO(gene = sigTreatment_up,
                    universe = bg_genes,
                    keyType = "ENSEMBL",
                    OrgDb = "org.Mm.eg.db",
                    ont = "BP",
                    qvalueCutoff = 0.05,
                    pAdjustMethod = "BH")

dotplot(compGO,
        showCategory = 20,
        title = "GO (Biological Process) Enrichment Analysis for UP in Treatment")
```

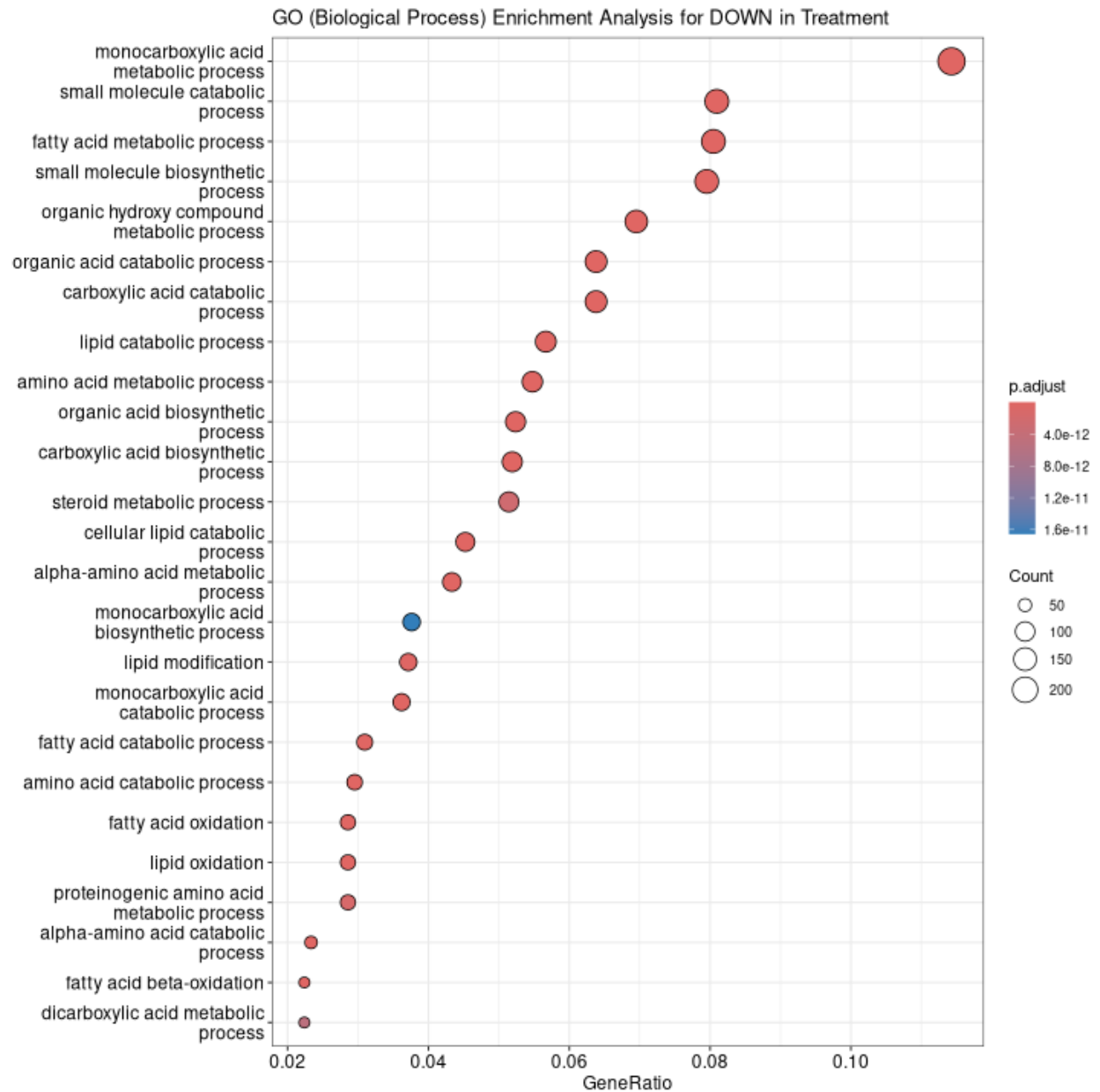


```
# image pdf 12 x 12
```

```
## Run GO enrichment analysis
```

```
compGO <- enrichGO(gene = sigTreatment_down,
                    universe = bg_genes,
                    keyType = "ENSEMBL",
                    OrgDb = org.Mm.eg.db,
                    ont = "BP",
                    qvalueCutoff = 0.05,
                    pAdjustMethod = "BH")
```

```
dotplot(compGO, showCategory = 25, title = "GO (Biological Process) Enrichment Analysis for DOWN in Tre
```



R session

```
sessionInfo()
```

```
## R version 4.4.1 (2024-06-14)
## Platform: x86_64-redhat-linux-gnu
## Running under: Fedora Linux 40 (Workstation Edition)
##
## Matrix products: default
## BLAS/LAPACK: FlexiBLAS OPENBLAS-OPENMP; LAPACK version 3.11.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8      LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8  LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8     LC_NAME=C
##  [9] LC_ADDRESS=C             LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## time zone: America/Toronto
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats4      stats      graphics  grDevices  utils      datasets  methods
## [8] base
##
## other attached packages:
##  [1] org.Mm.eg.db_3.19.1      clusterProfiler_4.12.1
##  [3] writexl_1.5.0           ggplotify_0.1.2
##  [5] knitr_1.48              ggrepel_0.9.5
##  [7] tximport_1.32.0         DESeq2_1.40.1
##  [9] pheatmap_1.0.12         DESeq2_1.44.0
## [11] SummarizedExperiment_1.34.0 MatrixGenerics_1.16.0
## [13] matrixStats_1.3.0       RColorBrewer_1.1-3
## [15] ensemblDb_2.28.0        AnnotationFilter_1.28.0
## [17] GenomicFeatures_1.56.0  AnnotationDbi_1.66.0
## [19] Biobase_2.64.0          GenomicRanges_1.56.1
## [21] GenomeInfoDb_1.40.1     IRanges_2.38.1
## [23] S4Vectors_0.42.1       AnnotationHub_3.12.0
## [25] BiocFileCache_2.12.0    dbplyr_2.5.0
## [27] BiocGenerics_0.50.0     lubridate_1.9.3
## [29] forcats_1.0.0          stringr_1.5.1
## [31] dplyr_1.1.4            purrr_1.0.2
## [33] readr_2.1.5            tidyr_1.3.1
## [35] tibble_3.2.1           ggplot2_3.5.1
## [37] tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] splines_4.4.1           BiocIO_1.14.0
##  [3] bitops_1.0-7           filelock_1.0.3
##  [5] polyclip_1.10-7        XML_3.99-0.17
##  [7] lifecycle_1.0.4        edgeR_4.2.1
##  [9] doParallel_1.0.17      vroom_1.6.5
## [11] lattice_0.22-6         MASS_7.3-60.2
```



```

## [13] backports_1.5.0
## [15] limma_3.60.4
## [17] yaml_2.3.9
## [19] DBI_1.2.3
## [21] abind_1.4-5
## [23] gggraph_2.2.1
## [25] yulab.utils_0.1.5
## [27] rappdirs_0.3.3
## [29] GenomeInfoDbData_1.2.12
## [31] tidytree_0.4.6
## [33] DelayedArray_0.30.1
## [35] ggforce_0.4.2
## [37] shape_1.4.6.1
## [39] UCSC.utils_1.0.0
## [41] viridis_0.6.5
## [43] jsonlite_1.8.8
## [45] tidygraph_1.3.1
## [47] foreach_1.5.2
## [49] treeio_1.28.0
## [51] glue_1.7.0
## [53] mnormt_2.1.1
## [55] xfun_0.45
## [57] withr_3.0.0
## [59] fastmap_1.2.0
## [61] digest_0.6.36
## [63] R6_2.5.1
## [65] colorspace_2.1-0
## [67] RSQLite_2.3.7
## [69] generics_0.1.3
## [71] rtracklayer_1.64.0
## [73] httr_1.4.7
## [75] scatterpie_0.2.3
## [77] gtable_0.3.5
## [79] ComplexHeatmap_2.20.0
## [81] shadowtext_0.1.4
## [83] fgsea_1.30.0
## [85] clue_0.3-65
## [87] logging_0.10-108
## [89] ggfun_0.1.5
## [91] rstudioapi_0.16.0
## [93] reshape2_1.4.4
## [95] nlme_3.1-164
## [97] cachem_1.1.0
## [99] BiocVersion_3.19.1
## [101] HDO.db_0.99.1
## [103] pillar_1.9.0
## [105] reshape_0.8.9
## [107] cluster_2.1.6
## [109] cli_3.6.3
## [111] compiler_4.4.1
## [113] rlang_1.1.4
## [115] labeling_0.4.3
## [117] fs_1.6.4
## [119] psych_2.4.6.26

magrittr_2.0.3
rmarkdown_2.27
cowplot_1.1.3
ConsensusClusterPlus_1.68.0
zlibbioc_1.50.0
RCurl_1.98-1.16
tweenr_2.0.3
circlize_0.4.16
enrichplot_1.24.2
codetools_0.2-20
DOSE_3.30.2
tidyselect_1.2.1
aplot_0.2.3
farver_2.1.2
GenomicAlignments_1.40.0
GetoptLong_1.0.5
iterators_1.0.14
tools_4.4.1
Rcpp_1.0.13
gridExtra_2.3
SparseArray_1.4.8
qvalue_2.36.0
BiocManager_1.30.23
fansi_1.0.6
timechange_0.3.0
gridGraphics_0.5-1
GO.db_3.19.1
utf8_1.2.4
data.table_1.15.4
graphlayouts_1.1.1
S4Arrays_1.4.1
pkgconfig_2.0.3
blob_1.2.4
XVector_0.44.0
htmltools_0.5.8.1
ProtGenerics_1.36.0
scales_1.3.0
png_0.1-8
ggdendro_0.2.0
tzdb_0.4.0
rjson_0.2.21
curl_5.2.1
GlobalOptions_0.1.2
parallel_4.4.1
restfulr_0.0.15
grid_4.4.1
vctrs_0.6.5
evaluate_0.24.0
locfit_1.5-9.10
Rsamtools_2.20.0
crayon_1.5.3
plyr_1.8.9
stringi_1.8.4
viridisLite_0.4.2

```

## [121] BiocParallel_1.38.0	munsell_0.5.1
## [123] Biostrings_2.72.1	lazyeval_0.2.2
## [125] GOSemSim_2.30.0	Matrix_1.7-0
## [127] patchwork_1.2.0	hms_1.1.3
## [129] bit64_4.0.5	KEGGREST_1.44.1
## [131] statmod_1.5.0	highr_0.11
## [133] igraph_2.0.3	broom_1.0.6
## [135] memoise_2.0.1	ggtree_3.12.0
## [137] fastmatch_1.1-4	bit_4.0.5
## [139] gson_0.1.0	ape_5.8