

RNA-seq course- week1

Serhiy Naumenko

2024-07-28

Contents

Overview	1
Load annotations	2
Load Counts	2
Cleanup and load metadata	2
Run DESeq2	3
Run DESeq2 Wald test	3
Sample-level QC analysis	4
PCA - treatment	4
PCA - experiment	4
top 1000 variable genes	6
PCA: Controls	8
PCA: treatment	9
Visualization	12
Heatmaps	13
Functional analysis	13
Create background dataset for hypergeometric testing using all genes tested for significance in the results	13
R session	15

Overview

- Petrenko2024 RNA-seq experiment for reanalysis

Load annotations

Load Counts

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

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

#counts_tpm_csv <- "../01_data/count_matrix_tpm.csv"
counts_csv <- "../01_data/counts.csv"
if (file.exists(counts_csv)){
  counts_prepared <- read_csv(counts_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)

  write_csv(counts_prepared, counts_csv)
}
counts <- counts_prepared %>% column_to_rownames(var = "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_oil")
  metadata_raw$stimulus <- str_replace_all(metadata_raw$stimulus, "control \\(saline\\)", "saline")
  metadata_raw$stimulus <- str_replace_all(metadata_raw$stimulus, "carbon tetrachloride", "carbon_tetrachloride")
}
```

```

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

```

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

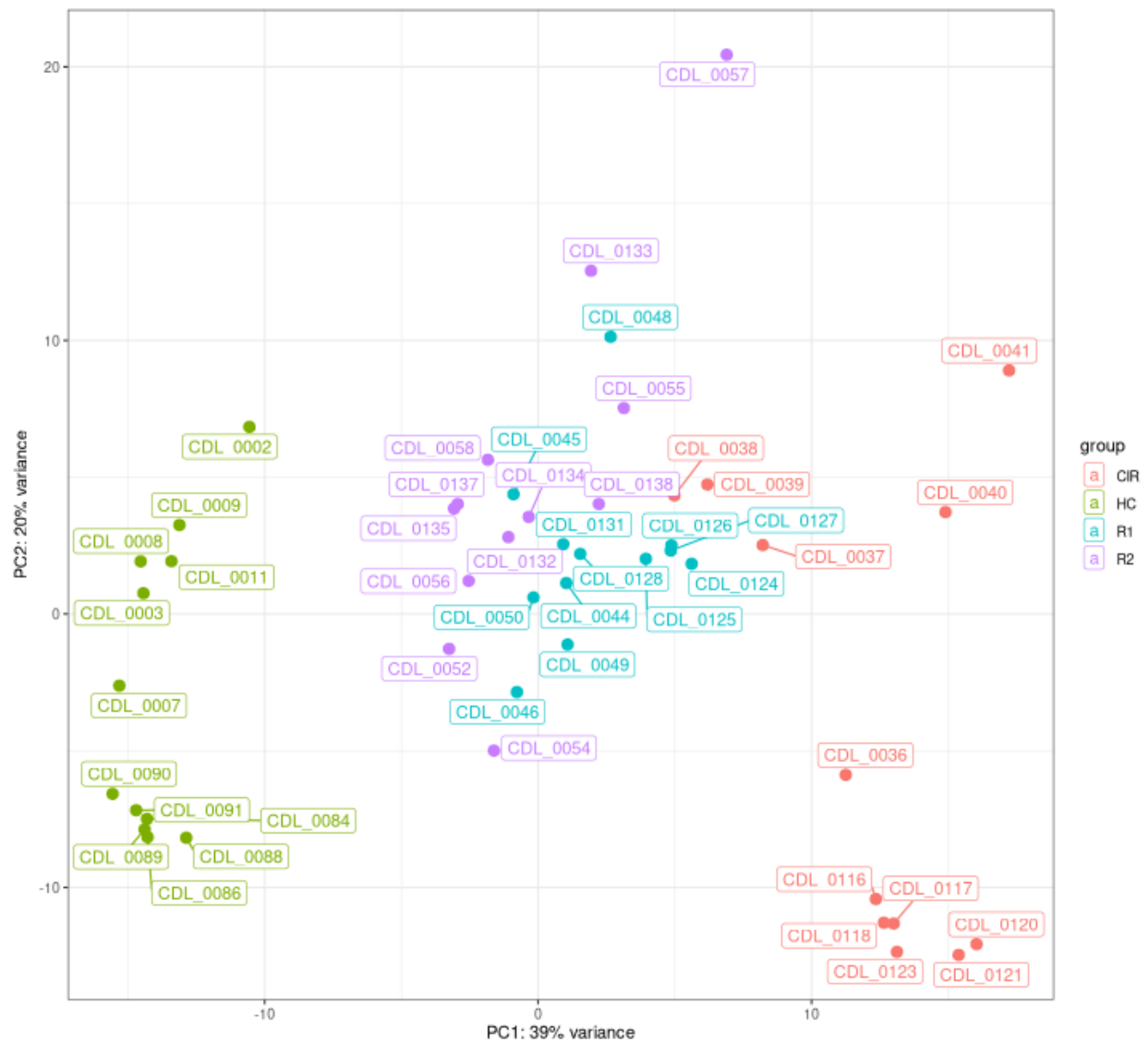
Run DEseq2 Wald test

Here we subset protein coding genes.

Sample-level QC analysis

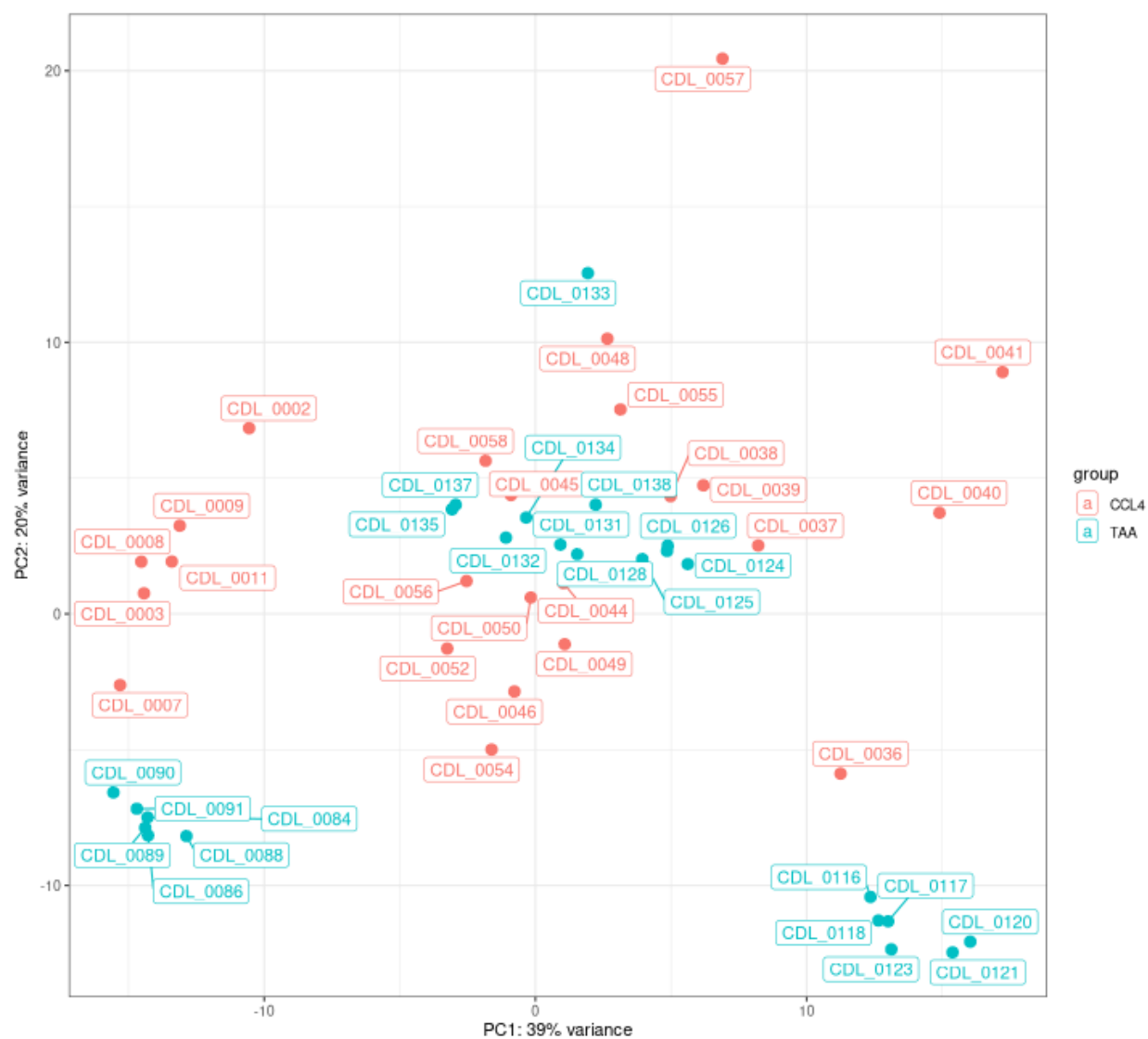
PCA - treatment

```
# Use the DESeq2 function  
plotPCA(rld, intgroup = c("treatment")) + geom_label_repel(aes(label = name)) + theme_bw()
```



PCA - experiment

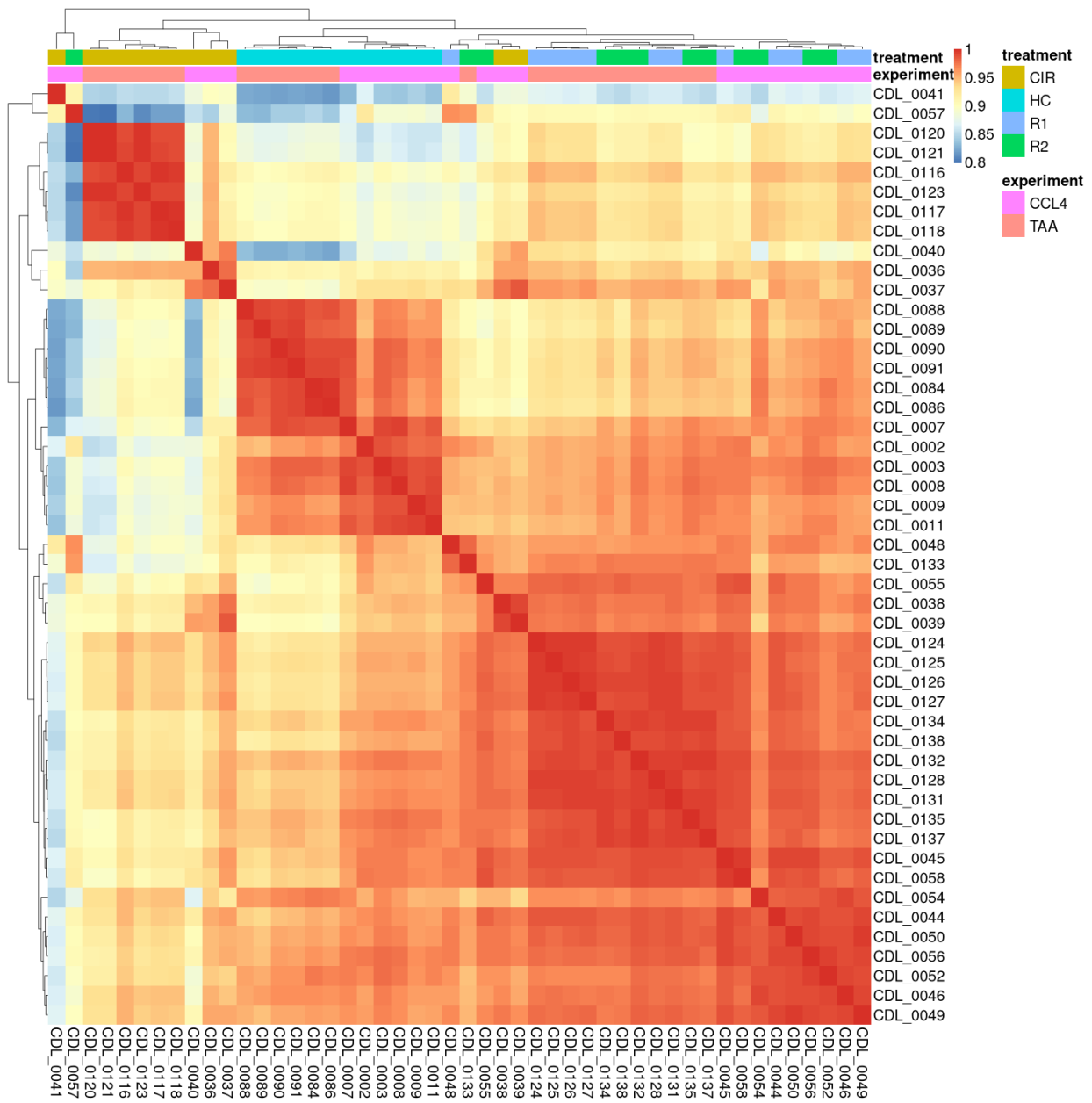
```
plotPCA(rld, intgroup = c("experiment")) + geom_label_repel(aes(label = name)) + theme_bw()
```



top 1000 variable genes

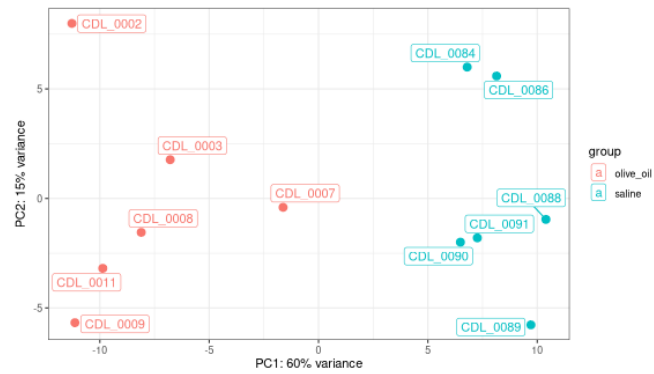
```
rv <- rowVars(rld_mat)
rv <- order(rv, decreasing = TRUE) %>% head(1000)
rld_mat_1000 <- rld_mat[rv,]
annotation <- metadata[, c("experiment", "treatment")]

# 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)
```



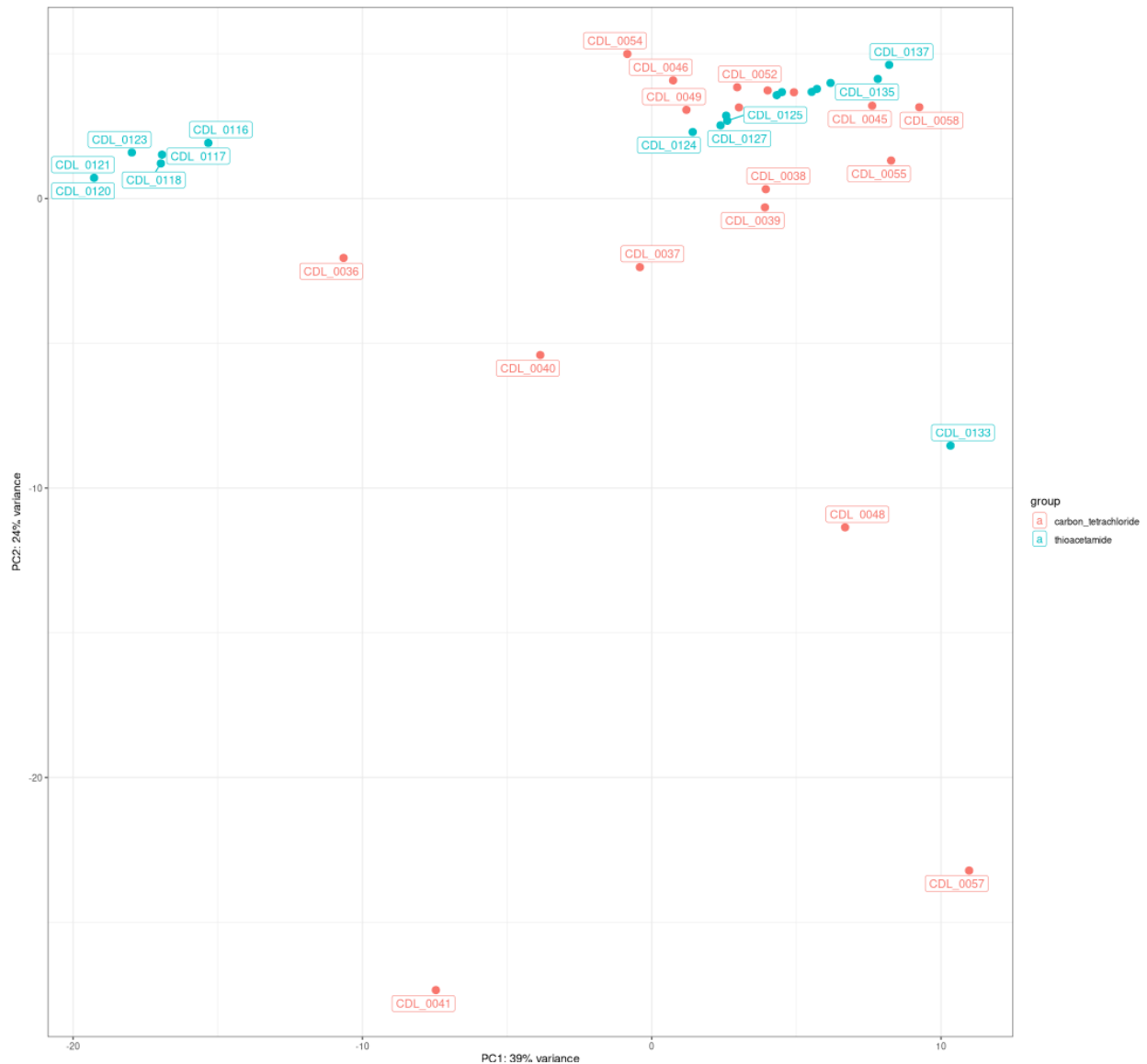
PCA: Controls

```
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: treatment

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



```
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("ensembl_gene_id") %>%
#  dplyr::filter(treatment == "DMSO") %>% pull("ensembl_gene_id")

```

```

#tpm_control <- get_counts_for_samples(counts_tpm, samples_control, "DMSO_mean_tpm")

#samples_effect <- metadata %>% dplyr::filter(treatment == "adapalene") %>% row.names()
#tpm_effect <- get_counts_for_samples(counts_tpm, samples_effect, "adapalene_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(T2.DE_adapalene = resTreatment_tb_significant),
#       "tables/T2.DE_adapalene.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 = "ENSG00000198846",  
#               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("TOX") +  
#  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: adapalene vs DMSO",
#   fontsize = 20)
```

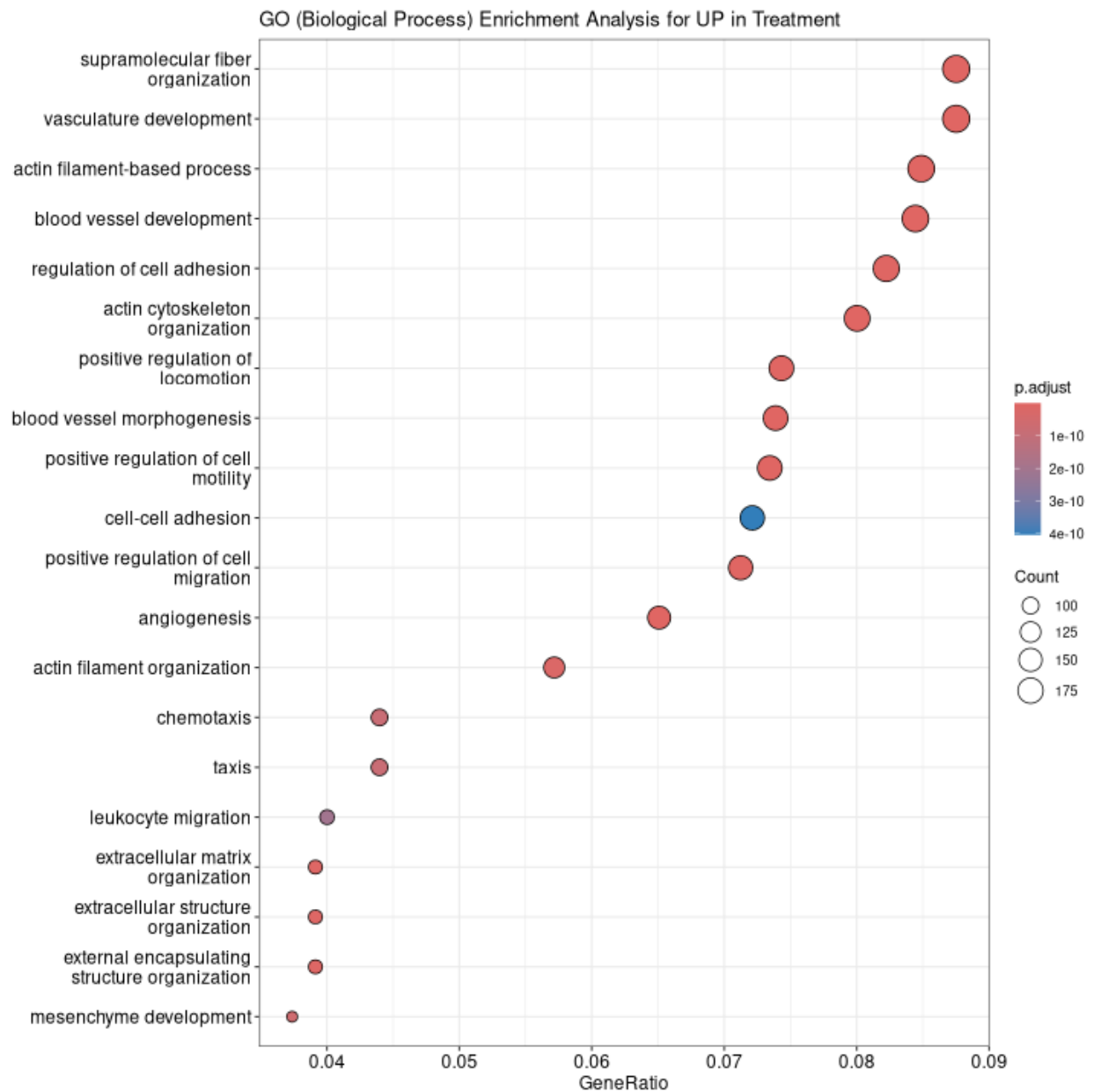
Functional analysis

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")
```

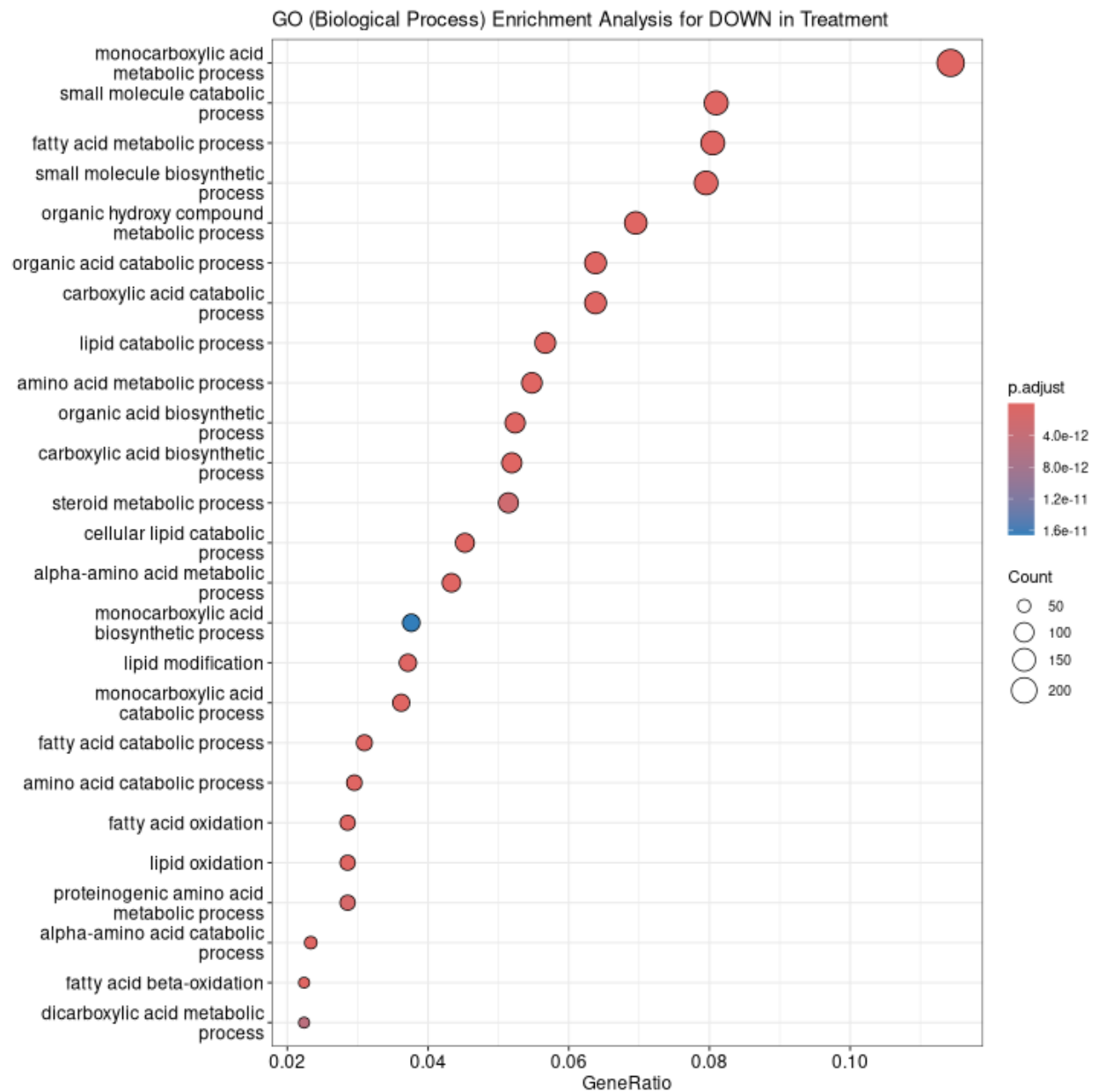


```
# image pdf 12 x 12
```

```
## 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         DEGreport_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         magrittr_2.0.3
## [15] limma_3.60.4            rmarkdown_2.27
## [17] yaml_2.3.9              cowplot_1.1.3
## [19] DBI_1.2.3               ConsensusClusterPlus_1.68.0
## [21] abind_1.4-5             zlibbioc_1.50.0
## [23] ggraph_2.2.1            RCurl_1.98-1.16
## [25] yulab.utils_0.1.5       tweenr_2.0.3
## [27] rappdirs_0.3.3          circlize_0.4.16
## [29] GenomeInfoDbData_1.2.12 enrichplot_1.24.2
## [31] tidytree_0.4.6          codetools_0.2-20

```


## [33] DelayedArray_0.30.1	DOSE_3.30.2
## [35] ggforce_0.4.2	tidyselect_1.2.1
## [37] shape_1.4.6.1	aplot_0.2.3
## [39] UCSC.utils_1.0.0	farver_2.1.2
## [41] viridis_0.6.5	GenomicAlignments_1.40.0
## [43] jsonlite_1.8.8	GetoptLong_1.0.5
## [45] tidygraph_1.3.1	iterators_1.0.14
## [47] foreach_1.5.2	tools_4.4.1
## [49] treeio_1.28.0	Rcpp_1.0.13
## [51] glue_1.7.0	gridExtra_2.3
## [53] mnormt_2.1.1	SparseArray_1.4.8
## [55] xfun_0.45	qvalue_2.36.0
## [57] withr_3.0.0	BiocManager_1.30.23
## [59] fastmap_1.2.0	fansi_1.0.6
## [61] digest_0.6.36	timechange_0.3.0
## [63] R6_2.5.1	gridGraphics_0.5-1
## [65] colorspace_2.1-0	G0.db_3.19.1
## [67] RSQLite_2.3.7	utf8_1.2.4
## [69] generics_0.1.3	data.table_1.15.4
## [71] rtracklayer_1.64.0	graphlayouts_1.1.1
## [73] httr_1.4.7	S4Arrays_1.4.1
## [75] scatterpie_0.2.3	pkgconfig_2.0.3
## [77] gtable_0.3.5	blob_1.2.4
## [79] ComplexHeatmap_2.20.0	XVector_0.44.0
## [81] shadowtext_0.1.4	htmltools_0.5.8.1
## [83] fgsea_1.30.0	ProtGenerics_1.36.0
## [85] clue_0.3-65	scales_1.3.0
## [87] logging_0.10-108	png_0.1-8
## [89] ggfun_0.1.5	ggdendro_0.2.0
## [91] rstudioapi_0.16.0	tzdb_0.4.0
## [93] reshape2_1.4.4	rjson_0.2.21
## [95] nlme_3.1-164	curl_5.2.1
## [97] cachem_1.1.0	GlobalOptions_0.1.2
## [99] BiocVersion_3.19.1	parallel_4.4.1
## [101] HDO.db_0.99.1	restfulr_0.0.15
## [103] pillar_1.9.0	grid_4.4.1
## [105] reshape_0.8.9	vctrs_0.6.5
## [107] cluster_2.1.6	evaluate_0.24.0
## [109] cli_3.6.3	locfit_1.5-9.10
## [111] compiler_4.4.1	Rsamtools_2.20.0
## [113] rlang_1.1.4	crayon_1.5.3
## [115] labeling_0.4.3	plyr_1.8.9
## [117] fs_1.6.4	stringi_1.8.4
## [119] psych_2.4.6.26	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