Chapter 5. Introduction to Functional Analysis: The clusterProfiler Tool

Performing enrichment analysis on RNA-seq data aims to identify biological pathways, Gene Ontology (GO) terms, or other functional categories that are represented in a set of differentially expressed genes (DEGs).

Summary of Functional Analysis Steps:

1. Convert Ensembl IDs to gene symbols, using the “mapIds” function with “org.Mm.eg.db,” the R package that provides genome-wide annotation for mouse.

2. Perform Gene Ontology enrichment: Identify terms which demonstrate a non-random association with GO biological processes, molecular functions, or cellular components, using a list of genes. Supply a list of specific genes, for example, the significant genes common to both DESeq2 and edgeR outputs; recall that a significant gene was defined as having |log2(fold change)| >= 1 and adjusted p-value < 0.05.

3. Perform KEGG and Reactome enrichment: Identify biological pathways which demonstrate a non-random association with KEGG and Reactome biological pathways, using a list of specific genes. Note that Reactome is constructed with human genes, thus mouse orthologs will be considered instead.

4. Perform GSEA using a ranked list of all genes, with the ranking metric equal to, for example, the signed $\log_2$(fold change) multiplied by -log10(p-value). Remark: The GSEA algorithm uses a p-value instead of an adjusted p-value for the metric, because the algorithm calculates a null distribution for its “enrichment scores” and then derives its own p-values and false discovery rates for gene clusters at the top or bottom of a ranked gene list.

Visualizations:

1. Dotplots:

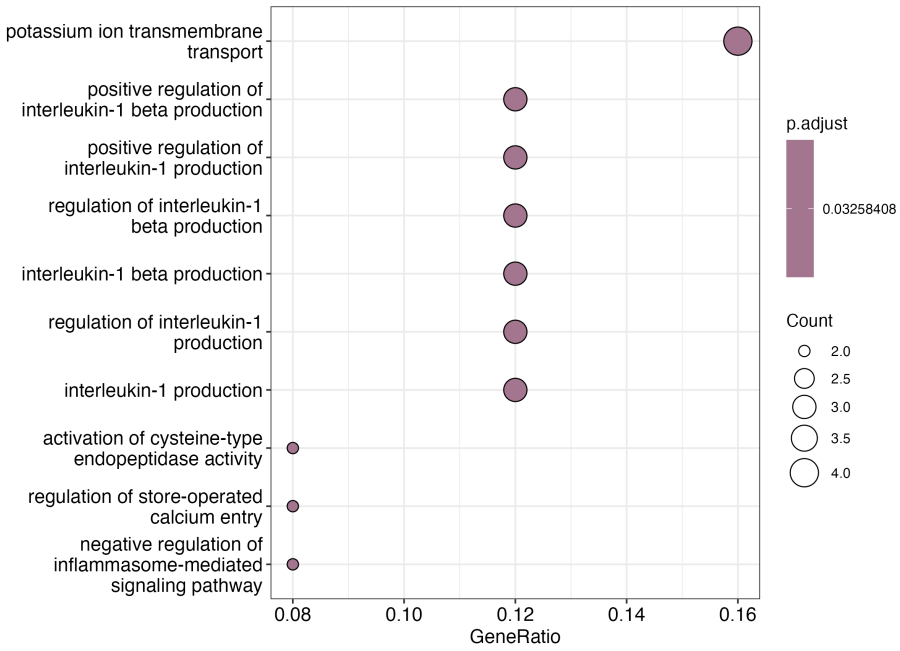

Figure 1. Dot plot: GO enrichment of a gene list, created from genes determined to be significant from DESeq2 analysis. Gene ratio = number of differentially expressed genes in GO term/total input genes.

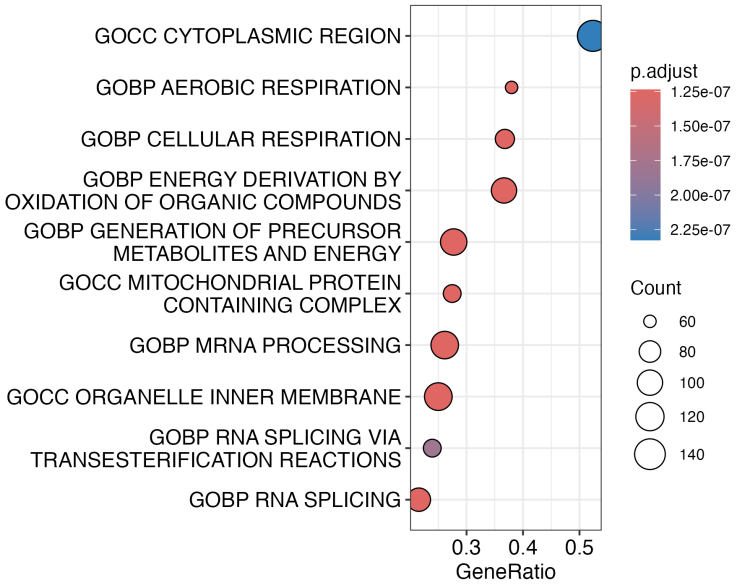

Figure 2. Dot plot: GSEA enrichment using GO terms associated with a ranked list of all genes (ranking described above).

2. Emapplot (Enrichment Map Plot): An emapplot displays the enriched terms (e.g., GO terms, KEGG pathways) as a network. Each node in the network represents an enriched term (e.g., "Immune response," "Cell adhesion"). The node size typically reflects the number of genes associated with that term. The color of the node typically represents the enrichment significance (e.g., p-value, adjusted p-value, or q-value). Edges connect two terms if they share a significant number of common genes.

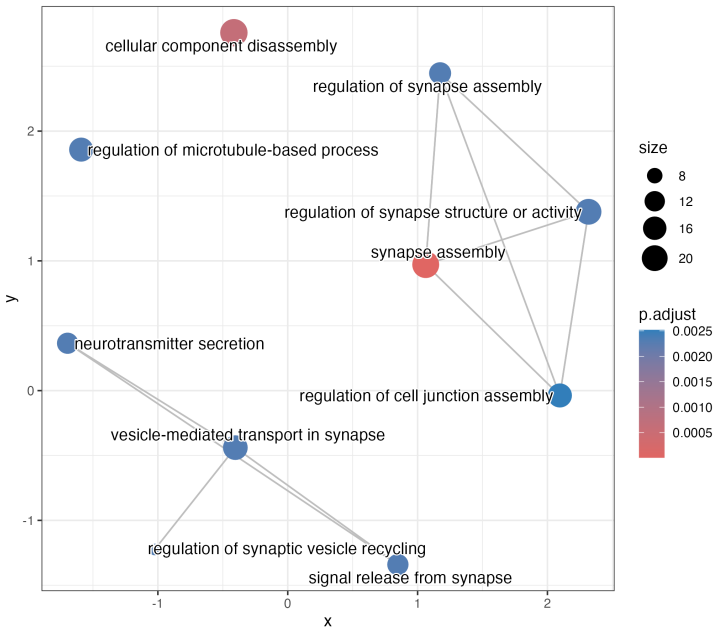

Figure 3. Emapplot of GO terms: Note that the x and y axes do not have a biological relevance. However, the relative positions are meaningful, that is, terms that are closer together are more similar (share more genes).

3. CNET Plots (Gene-Concept Network): Shows the relationships between individual genes and the enriched terms that they belong to. There are two types of nodes: 1) enriched terms (e.g., GO terms, KEGG pathways) and 2) individual genes connected to one or more enriched terms. Edges connect genes to terms or pathways. Helpful for identifying "hub" genes, meaning genes involved in multiple enriched terms. Genes and enriched terms are colored differently.

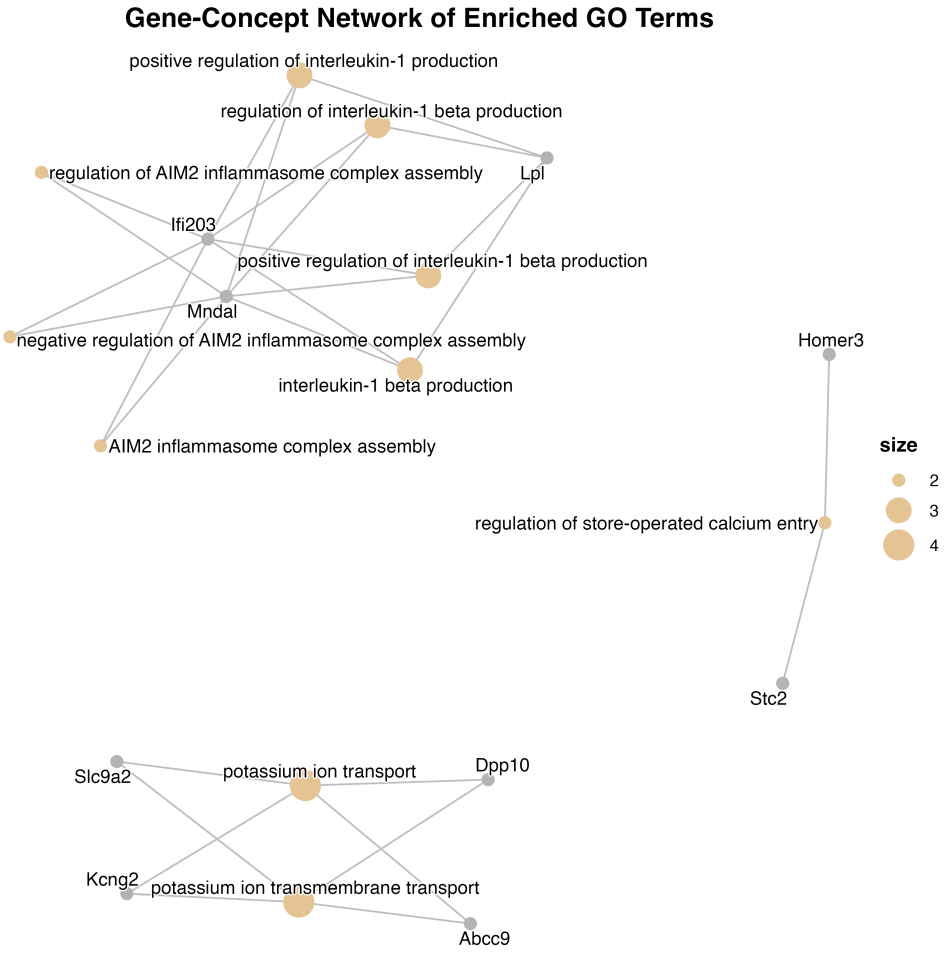

Figure 4. CNET Plot of significant genes (from DESeq2 analysis) and GO terms (biological processes). Size of a term node represents number of genes associated with that term. Size of gene nodes are constant.

One software package used in functional analysis is clusterProfiler. To be clear, there are two types of functional analysis: Over-Representation Analysis (ORA) and Gene Set Enrichment Analysis (GSEA), and clusterProfiler can be used for both types of analyses:

Over-Representation Analysis (ORA) and clusterProfiler:

These methods take a predefined list of significant genes (e.g., differentially expressed genes based on a certain threshold) and check if any functional categories (GO terms, KEGG pathways, Reactome pathways) are over-represented within that list compared to what would be expected by chance from the background gene population. The input is a binary list of genes (significant vs. not significant). Performing GO, KEGG, and Reactome enrichment analysis using functions like enrichGO(), enrichKEGG(), and enrichPathway() in clusterProfiler is a form of Over-Representation Analysis (ORA), not GSEA.

Gene Set Enrichment Analysis (GSEA) and clusterProfiler:

This method, implemented in clusterProfiler using functions like gseGO(), gseKEGG(), and gsePathway(), takes a ranked list of all genes from your experiment (e.g., ranked by their degree of differential expression) and examines whether genes belonging to a particular functional category tend to appear towards the top or bottom of the ranked list. The input is a continuous, ranked list of genes. For GO GSEA: gseGO(); For KEGG GSEA: gseKEGG(); For Reactome GSEA: gsePathway().

Remarks:

1. The enrichKEGG() in clusterProfiler performs Over-Representation Analysis (ORA) to test if certain KEGG pathways are statistically enriched in a gene list, compared to a background. It uses the hypergeometric test to compute enrichment significance for each pathway. The background is all genes in the KEGG species database (e.g., all mouse genes with KEGG annotations, not all genes in your experiment).

2. Reactome is a manually curated database of biological pathways (human-centric, with ortholog projections to mouse and other species). ReactomePA::enrichPathway() performs Over-Representation Analysis (ORA): 1) compares your list of genes (e.g., DEGs) to predefined pathways and 2) uses the hypergeometric test to determine whether a pathway has more genes from your list than expected by chance.

Installation of clusterProfiler:

```
$ conda create -n cluster_env -c conda-forge -c bioconda

$ conda activate cluster_env

$ conda install -c conda-forge libxml2 libiconv libcurl libzip pkg-config gcc_linux-64
gxx_linux-64   (all one line)

$ R

> BiocManager::install("clusterProfiler")
> library(clusterProfiler) # this also checks if correctly installed

> BiocManager::install("org.Mm.eg.db")
> library(org.Mm.eg.db) # this also checks if correctly installed

> BiocManager::install(“reactome.db")
```

```
> library(reactome.db) # this also checks if correctly installed
> BiocManager::install("ReactomePA")
> library(ReactomePA) # this also checks if correctly installed

> install.packages("dplyr")
> library(dplyr)

> install.packages("ggplot2")
> library(ggplot2)
```

Note: Load all these R packages (using library as above) every time R starts up. You only need to install the packages once, not every time you start R.

To run clusterProfiler and achieve these visualizations:

Download mouse_cluster.R from the website:

https://github.com/elinorv21/RNA-Seq_workshop/

Edit the code (as described below), give the correct permissions, and run in your terminal window (either on Savio or on your local computer/laptop):

```
$ chmod +x mouse_cluster.R
$ ./mouse_cluster.R
```

Alternatively, once you've downloaded mouse_cluster.R, you can open it in RStudio and edit it there.

mouse_cluster.R code (**Bolded terms** are items to edit):

setwd("~/Desktop") # You may want to modify your working directory

```
############
### ORA ### (not GSEA)
############

########################################
# ORA: enrichGO with common sig genes ##
########################################

> library(clusterProfiler)
> library(org.Mm.eg.db) # mouse
> library(dplyr)
> library(tibble)   # rownames_to_column()
> library(enrichplot)
> library(DOSE)
```

load data (created from mouse_dge.R) **You may want to modify your files' paths:**

```
#readRDS("path/to/res_deseq2.rds")
> deseq2_results_table_filtered <- readRDS("~/Desktop/mouse_data/res_deseq2.rds")

#readRDS("path/to/res_edger.rds")
```

> top_tags_lrt <- readRDS("**~/Desktop/mouse_data/res_edger.rds**")

significant genes from DESeq2
> sig_genes_deseq2 <- readRDS("**~/Desktop/mouse_data/sig_genes_deseq2.rds**")

significant genes from edgeR
> sig_genes_edger <- readRDS("**~/Desktop/mouse_data/sig_genes_edger.rds**")

1. Choose the significant genes from DESeq2 and edgeR(from dge.R output) for our analysis:

> deseq2_df <- as.data.frame(sig_genes_deseq2) %>% tibble::rownames_to_column("gene")
> edger_df <- as.data.frame(sig_genes_edger) %>% tibble::rownames_to_column("gene")
> common_sig_genes <- intersect(deseq2_df$gene, edger_df$gene) # gene_list vector

> cat("number of significant genes common to both DESeq2 and edgeR")
> print(length(common_sig_genes)) # number of significant genes common to both DESeq2 and edgeR

> deseq2_common <- deseq2_df[deseq2_df$gene %in% common_sig_genes,]
> edger_common <- edger_df[edger_df$gene %in% common_sig_genes,]
> sig_genes <- inner_join(deseq2_common, edger_common, by = "gene") # dataframe

2. Convert gene IDs to Entrez IDs:

> library(clusterProfiler)

> gene_list <- common_sig_genes
Remove the version number (e.g., ".2" etc) from the Ensembl gene IDs in row names
> gene_list <- gsub("\\.\\d+$", "", gene_list)

> gene_entrez <- bitr(gene_list, fromType = "ENSEMBL", toType = "ENTREZID", OrgDb = org.Mm.eg.db)

3. Perform GO enrichment analysis:

> ego <- enrichGO(gene = gene_entrez$ENTREZID,
            OrgDb = org.Mm.eg.db,
            keyType = "ENTREZID",
            # You may want to modify your term type:
            ont = "BP", # Biological Process (can also be "CC" for Cellular Component, "MF" for Molecular Function, or "ALL")
            pvalueCutoff = 0.05, # Significance cutoff for p-values
            qvalueCutoff = 0.05, # Significance cutoff for adjusted p-values (FDR)
            readable = TRUE) # Make gene IDs in the results readable (gene symbols)

4. Visualization examples

> dotplot(ego, showCategory = 20)   # bubble plot (see below)
> emapplot(pairwise_termsim(ego))   # term similarity graph
> cnetplot(ego, categorySize = "pvalue")# gene-concept network

> library(enrichplot)
> library(ggplot2)

```
> g1 <- dotplot(ego)
```

 # You may want to modify your file path and filename:

```
> ggsave(filename = "mouse_data/cluster/ora/deseq2_GO_BP_dotplot_revised.png", plot =
g1, width = 8, height = 6, dp = 300)  # all one line

> deseq2_go_bp_sim <- pairwise_termsim(ego)
> g2 <- emapplot(deseq2_go_bp_sim,
            showCategory = 10,   # adjust as you like: default 30
            layout       = "kk")  # "kk" or "fruchterman.reingold" = "fr"
```

 # You may want to modify your file path and filename:

```
> ggsave(filename = "mouse_data/cluster/ora/deseq2_GO_BP_emapplot.png", plot = g2,
width = 8, height = 6, dpi = 300) # all one line

# 5. Publication quality plots
# a. bubble plot

> library(ggplot2)
> library(enrichplot)  # enhances clusterProfiler visuals

# Set up the plot
> g1 <- dotplot(ego,
            showCategory = 10,
            font.size = 12,
            # You may want to modify the title
            title = "GO Enrichment (Biological Process)",
            color = "p.adjust") +  # color by adjusted p-value
  scale_color_gradient(low = "#56B1F7", high = "#132B43") +  # custom color
  theme_minimal(base_size = 14) +  # cleaner theme
  theme(
    plot.title = element_text(face = "bold", size = 16, hjust = 0.5),
    axis.text.y = element_text(size = 12, face = "bold"),
    axis.text.x = element_text(size = 12),
    legend.title = element_text(face = "bold", size = 12),
    legend.text = element_text(size = 10),
    panel.grid.major.y = element_blank()
  ) +
  labs(x = "Gene Ratio", y = NULL, color = "Adj. p-value")

# Print the plot
```

 # You may want to modify your file path and filename:

```
> ggsave(filename = "mouse_data/cluster/ora/deseq2_GO_BP_dotplot_pub.png", plot = g1,
width = 8, height = 6, dp = 300)

# b. term similarity graph ### publication-level plot

> library(clusterProfiler)
> library(enrichplot)
> library(ggplot2)
```

```
# Convert gene IDs
> sig_genes_entrez_ids <- mapIds(
  org.Mm.eg.db,
  keys      = gene_list,
  column    = "ENTREZID",
  keytype   = 'ENSEMBL',
  multiVals = "first"
) %>% unname() %>% na.omit()

# test if dataset is empty (if “true,” it’s empty)
> is.null(sig_genes_entrez_ids)

> new_keys <- gsub("\\.\\d+$", "", rownames(deseq2_results_table_filtered))

> universe_entrez <- mapIds(
  org.Mm.eg.db,
  keys      = new_keys,
  column    = "ENTREZID",
  keytype   = 'ENSEMBL',
  multiVals = "first"
) %>% unname() %>% na.omit()

# test if dataset is empty (if “true,” it’s empty)
> is.null(universe_entrez)

> deseq2_go_bp <- enrichGO(
  gene          = sig_genes_entrez_ids,
  universe      = universe_entrez,
  OrgDb         = org.Mm.eg.db,
  keyType       = “ENTREZID",
  # you may want to modify your ontology term
  ont           = "BP", # Biological Process or try "MF" (Molecular Function) or "CC" (Cellular
Component)
  pAdjustMethod = "BH",
  pvalueCutoff  = 0.05,
  qvalueCutoff  = 0.05,
  minGSSize     = 10,   # adjust if your list is small
  maxGSSize     = 500,
  readable      = TRUE  # adds gene symbols in result
)

# Compute term similarity
> deseq2_go_bp_sim <- pairwise_termsim(deseq2_go_bp)

# Generate the emapplot with default size and edge style
> emap_obj <- emapplot(
  deseq2_go_bp_sim,
  showCategory = 10,         # Show top 10 enriched terms
  layout       = "kk",       # Kamada–Kawai layout
  color        = "p.adjust"  # Node color by adjusted p-value
)

# Enhance plot styling
```

```
> g2 <- emap_obj +
  ggtitle("Semantic Similarity Network of Enriched GO Terms (BP)") +
  theme_minimal(base_size = 14) +
  theme(
    plot.title    = element_text(hjust = 0.5, face = "bold", size = 16),
    legend.title  = element_text(size = 12, face = "bold"),
    legend.text   = element_text(size = 10),
    axis.text     = element_blank(),
    axis.ticks    = element_blank()
  ) +
  scale_color_gradient(low = "#56B1F7", high = "#132B43", name = "Adj. p-value")

# Save high-resolution plot
# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/ora/deseq2_GO_BP_emapplot_pub.png", plot = g2, width =
10, height = 8, dpi = 400)

# You may want to modify your file path and filename:
> ggsave(filename = "dge_output_github_test/ora/deseq2_GO_BP_emapplot_pub.pdf", plot
= g2, width = 10, height = 8)

# c.gene-concept network

> g3 <- cnetplot(ego, categorySize = “pvalue")

# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/ora/deseq2_GO_BP_emapplot_pub.png", plot = g3, width =
10, height = 8, dpi = 400)

> library(clusterProfiler)
> library(enrichplot)
> library(ggplot2)

# Gene-concept network with custom aesthetics
> g3 <- cnetplot(ego,
            showCategory = 10,              # top N GO terms
            categorySize = "pvalue",         # can also use "geneNum"
            foldChange = NULL,               # or supply a named vector of log2FCs
            circular = FALSE,               # circular layout? (optional)
            colorEdge = TRUE,               # edge color = gene-node connections
            node_label = "all") +           # show both gene & term names
  ggtitle("Gene-Concept Network of Enriched GO Terms") +
  theme_minimal(base_size = 14) +
  theme(
    plot.title    = element_text(size = 16, face = "bold", hjust = 0.5),
    legend.title  = element_text(size = 12, face = "bold"),
    legend.text   = element_text(size = 10),
    axis.text     = element_blank(),
    axis.ticks    = element_blank()
  )

# Save as high-resolution PNG and PDF
# You may want to modify your file path and filename:
```

```
> ggsave("mouse_data/cluster/ora/deseq2_GO_BP_cnetplot_pub.png", plot = g3, width =
10, height = 8, dpi = 400) # all one line

# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/ora/deseq2_GO_BP_cnetplot_pub.pdf", plot = g3, width =
10, height = 8) # all one line

# revised

> library(clusterProfiler)
> library(enrichplot)
> library(ggplot2)

# Step 1: Create a base plot
> g3 <- cnetplot(
  ego,
  showCategory = 10,
  categorySize = "pvalue",
  node_label = "all",        # show both gene and term names
  colorEdge = TRUE,          # show colored edges
  circular = FALSE           # normal layout, not circular
)

# Step 2: Customize the theme (white background + clearer fonts)
> g3_pub <- g3 +
  ggtitle("Gene-Concept Network of Enriched GO Terms") +
  theme_void(base_size = 14) +  # fully blank canvas
  theme(
    plot.background = element_rect(fill = "white", color = NA),
    panel.background = element_rect(fill = "white", color = NA),
    plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
    legend.title = element_text(size = 12, face = "bold"),
    legend.text = element_text(size = 10)
  )

# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/ora/deseq2_GO_BP_cnetplot_pub_revised2.png", plot =
g3_pub, width = 8, height = 8, dpi = 400) # all one line

##########################################
# ORA: KEGG Pathway with common sig genes ##
##########################################

> library(clusterProfiler)
> library(org.Mm.eg.db)

# Perform KEGG pathway enrichment analysis
> kegg_path <- enrichKEGG(gene = sig_genes_entrez_ids,
                          organism = 'mmu', # Mouse organism
                          pvalueCutoff = 1,
                          pAdjustMethod = "BH",
                          qvalueCutoff = 1
)
```

```
# troubleshooting in case we get null results
> head(kegg_path)
> is.character(sig_genes_entrez_ids)
> clusterProfiler::download_KEGG("mmu")

> res_deseq2 <- deseq2_results_table_filtered
> res_edger <- top_tags_lrt

> sig_genes_deseq2_relaxed <- res_deseq2[res_deseq2$padj < 0.05 &
abs(res_deseq2$log2FoldChange) >= 0.5, ]

> sig_genes_edger_relaxed <- res_edger[res_edger$FDR < 0.05 & abs(res_edger$logFC) >=
0.5, ]

> common_sig_genes_relaxed <- intersect(rownames(sig_genes_deseq2_relaxed),
rownames(sig_genes_edger_relaxed)) # gene_list vector

> new_keys_entrez <- gsub("\\.\\d+$", "", common_sig_genes_relaxed)

> sig_genes_entrez_ids_relaxed <- mapIds(
  org.Mm.eg.db,
  keys      = new_keys_entrez,
  column    = "ENTREZID",
  keytype   = 'ENSEMBL',
  multiVals = "first"
) %>% unname() %>% na.omit()

> kegg_path <- enrichKEGG(gene = sig_genes_entrez_ids_relaxed,
                          organism = 'mmu', # Mouse organism
                          pvalueCutoff = 0.05,
                          pAdjustMethod = "BH",
                          qvalueCutoff = 0.05
                          )

> kegg_path@result$GeneRatio <- parse_ratio(kegg_path@result$GeneRatio)

> print(as.data.frame(kegg_path)) # prints the list of pathways
# Set pvalueCutoff = qvalueCutoff = 1 to check for errors if no pathways

#######@@
> library(ggplot2)

> plot_data <- kegg_path@result %>%
  arrange(p.adjust) %>% # Sort by p.adjust (ascending)
  head(10) # Take the top 10 categories

# Create the dotplot manually using ggplot2
> p_manual_dotplot <- ggplot(plot_data,
                             aes(x = GeneRatio,
                                 y = reorder(Description, GeneRatio), # Order by GeneRatio for aesthetic
                                 size = Count,
                                 color = p.adjust)) +
  geom_point() +
```

```
  scale_color_gradient(low = "red", high = "blue", name = "Adjusted P-value") + # Adjust colors
as desired
  scale_size_continuous(range = c(2, 10), name = "Gene Count") + # Adjust size range
  labs(title = "KEGG Pathway Enrichment (ORA)",
       x = "GeneRatio",
       y = "Pathway Description") +
  theme_bw() +
  theme(axis.text.y = element_text(angle = 0, hjust = 1, size = 10), # Adjust y-axis text
        plot.title = element_text(hjust = 0.5, face = "bold", size = 14))

# Save the plot
# You may want to modify your file path and filename:
> ggsave(filename = "mouse_data/cluster/ora/kegg_dotplot_manual.png",
       plot = p_manual_dotplot,
       width = 8, height = 6, dpi = 300)



# publication-quality plot for KEGG:
####@
> library(ggplot2) # Ensure ggplot2 is loaded

# Make sure kegg_path is your enrichResult object after GeneRatio conversion
# kegg_path@result$GeneRatio <- parse_ratio(kegg_path@result$GeneRatio)

# 1. Extract the results table and select top 10 categories
> plot_data <- kegg_path@result %>%
  arrange(p.adjust) %>% # Sort by p.adjust (ascending)
  head(10) # Take the top 10 categories

# 2. Create the ggplot
> publication_dotplot <- ggplot(plot_data,
                          aes(x = GeneRatio,
                              y = reorder(Description, GeneRatio), # Reorder Y-axis by GeneRatio
                              size = Count,
                              color = p.adjust)) +
  geom_point() + # Creates the dots
  # Custom color scale
  scale_color_gradient(low = "#56B1F7", high = "#132B43", name = "Adj. p-value") +
  # Custom size scale for dots (adjust range if needed)
  scale_size_continuous(range = c(3, 10), name = "Gene Count") +
  # Labels and Title
  labs(x = "Gene Ratio",
       y = NULL, # No label on Y-axis as descriptions are sufficient
       title = "KEGG Enrichment") +
  # Theme and styling
  theme_bw() + # Start with a clean theme
  theme(
    plot.title = element_text(face = "bold", size = 16, hjust = 0.5), # Title bold, larger, centered
    axis.text.y = element_text(size = 12, face = "bold"), # Y-axis text bold
    axis.text.x = element_text(size = 12),
    legend.title = element_text(face = "bold", size = 12),
    legend.text = element_text(size = 10),
```

```
   panel.grid.major.y = element_blank(), # Remove horizontal grid lines
   panel.grid.minor.x = element_blank(), # Remove minor vertical grid lines
   panel.grid.major.x = element_line(color = "grey80", linetype = "dotted") # Dotted vertical grid
lines
 )

# 3. Print the plot (if in an interactive session)
> print(publication_dotplot)

# 4. Save the plot
# You may want to modify your file path and filename:
> ggsave(filename = "mouse_data/cluster/ora/kegg_publication_dotplot.png",
      plot = publication_dotplot,
      width = 8, height = 6, dpi = 300)
####@

#########################
# ORA: Reactome Pathways #
#########################

> library(ReactomePA)
> library(org.Mm.eg.db) # mouse

# Perform Reactome pathway enrichment analysis
> reactome_enrichment <- enrichPathway(gene         = sig_genes_entrez_ids_relaxed,
                                       organism     = 'mouse',
                                       pvalueCutoff = 0.05,
                                       pAdjustMethod = "BH",
                                       qvalueCutoff = 0.05,
                                       universe     = universe_entrez)

> head(reactome_enrichment)

> reactome_enrichment@result$GeneRatio <-
parse_ratio(reactome_enrichment@result$GeneRatio)

> plot_data <- reactome_enrichment@result %>%
  arrange(p.adjust) %>% # Sort by p.adjust (ascending)
  head(10)

# Create the dotplot manually using ggplot2
> p_manual_dotplot <- ggplot(plot_data,
                             aes(x = GeneRatio,
                                 y = reorder(Description, GeneRatio), # Order by GeneRatio for aesthetic
                                 size = Count,
                                 color = p.adjust)) +
  geom_point() +
  scale_color_gradient(low = "red", high = "blue", name = "Adjusted P-value") + # Adjust colors
as desired
  scale_size_continuous(range = c(2, 10), name = "Gene Count") + # Adjust size range
  labs(title = "Reactome Enrichment (ORA)",
       x = "Gene Ratio",
       y = "Pathway Description") +
```

```
  theme_bw() +
  theme(axis.text.y = element_text(angle = 0, hjust = 1, size = 10), # Adjust y-axis text
        plot.title = element_text(hjust = 0.5, face = "bold", size = 14))

# Save the plot
# You may want to modify your file path and filename:
> ggsave(filename = "mouse_data/cluster/ora/reactome_ORA_dotplot_manual.png",
       plot = p_manual_dotplot,
       width = 8, height = 6, dpi = 300)



#####################################
# GSEA (Gene Set Enrichment Analysis) #
#####################################

#####################################
### GSEA for GO Terms with DESeq2 ###
#####################################

# BiocManager::install("msigdbr")

# Load R packages
> library(dplyr)
> library(tibble)
> library(edgeR)
> library(msigdbr)
> library(AnnotationDbi)
> library(org.Mm.eg.db)
> library(enrichplot)
> library(clusterProfiler)
> library(ggplot2)

# Rank DESeq2 results:
> ranked_genes_deseq2 <- as.data.frame(deseq2_results_table_filtered) %>%
  rownames_to_column(var = "gene_id") %>%
  dplyr::select(gene_id, log2FoldChange, padj) %>%
  na.omit() %>%
  # Create the GSEA ranking metric
  dplyr::mutate(ranking_score = sign(log2FoldChange) * (-log10(padj))) %>%
  dplyr::arrange(desc(ranking_score)) %>%
  dplyr::pull(ranking_score, name = gene_id)

> cat("Glimpse ranked genes DESeq2:")
> head(ranked_genes_deseq2)

> new_keys_ranked <- gsub("\\.\\d+$", "", names(ranked_genes_deseq2))

# For GSEA with MSigDB GO gene sets, map Ensembl IDs to gene symbols:
> ranked_genes_symbol <- mapIds(org.Mm.eg.db,
                          keys = new_keys_ranked,
                          keytype = "ENSEMBL",
                          column = "SYMBOL")
```

```
# Remove unmapped IDs
> ranked_genes_symbol <- ranked_genes_symbol[!is.na(ranked_genes_symbol)]

# We need to create a temporary vector where the names are the UNVERSIONED Ensembl IDs
#    to match with `ranked_genes_symbol`.
ranked_scores_unversioned <- ranked_genes_deseq2
names(ranked_scores_unversioned) <- new_keys_ranked # Now, names are UNVERSIONED
Ensembl IDs

# Filter the unversioned ranked scores to keep only those that successfully mapped to a
symbol.
#    This comparison is now valid: unversioned IDs %in% unversioned IDs
filtered_ranked_scores <- ranked_scores_unversioned[names(ranked_scores_unversioned)
%in% names(ranked_genes_symbol)]

# Assign the actual gene symbols as names to the filtered scores.
#    Here, we use the `ranked_genes_symbol` vector to look up symbols
#    using the UNVERSIONED Ensembl IDs that are currently the names of
`filtered_ranked_scores`.
names(filtered_ranked_scores) <- ranked_genes_symbol[names(filtered_ranked_scores)]

# Handle potential duplicate gene symbols that might arise if different Ensembl IDs
#    (even unversioned ones) map to the same gene symbol. GSEA typically requires unique
gene names.
#    If duplicates exist, common practice is to keep the score of the gene with the highest
absolute ranking score.
ranked_list_for_gsea_symbol_unique <- data.frame(
  symbol = names(filtered_ranked_scores),
  score = filtered_ranked_scores
) %>%
  dplyr::group_by(symbol) %>%
  dplyr::summarise(score = score[which.max(abs(score))]) %>% # Keep the score with max
absolute value
  dplyr::ungroup() %>%
  tibble::deframe() # Convert back to a named vector

names(ranked_list_for_gsea_symbol_unique) <-
toupper(names(ranked_list_for_gsea_symbol_unique))

# Ensure the list is sorted by score (descending) as GSEA expects.
ranked_list_for_gsea_symbol_unique <- sort(ranked_list_for_gsea_symbol_unique, decreasing
= TRUE)

# This `ranked_list_for_gsea_symbol` is your final input for GSEA.
head(ranked_list_for_gsea_symbol_unique)

# Examine your ranking_score distribution:
# You can get the full filtered_ranked_scores and plot its histogram
hist(filtered_ranked_scores, breaks=50, main="Distribution of Ranking Scores")
# Also check unique values
length(unique(filtered_ranked_scores)) / length(filtered_ranked_scores) * 100 # Percentage of
unique scores
# If this percentage is very low (e.g., under 50%), then the ties are severe.
```

```
# Get gene sets for GSEA (using MSigDB as an example)
# Get MSigDB GO gene sets (using symbols)

gene_sets_for_gsea <- msigdbr(
  species     = "Mus musculus",
  collection  = "C5"#,
  # subcategory = "GO:BP"        # optional – pick the GO branch
) %>%
  dplyr::select(gs_name, gene_symbol) %>%
  dplyr::mutate(gene_symbol = toupper(as.character(gene_symbol)))


#######################################
# GSEA using GO gene sets with DESeq2 #
#######################################

# Perform GSEA using symbols
> gsea_results_go_symbol <- GSEA(geneList = ranked_list_for_gsea_symbol_unique,
                         TERM2GENE = gene_sets_for_gsea,
                         pvalueCutoff = 0.05,
                         pAdjustMethod = "BH",
                         minGSSize = 10,
                         maxGSSize = 500,
                         eps = 1e-10)

# Visualization

> gsea1_go <- dotplot(gsea_results_go_symbol, showCategory = 10)
> gsea2_go <- emapplot(
              pairwise_termsim(gsea_results_go_symbol),
              showCategory = 10,
              layout = "fr"
              ) +
              theme_bw() +
              labs(title = "Enrichment Map of Top 10 GO Terms")

# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/gsea/deseq2_gsea_go_dotplot.png", plot = gsea1_go, width
= 8, height = 6, dpi = 300)  # all one line

# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/gsea/deseq2_gsea_go_emapplot.png", plot = gsea2_go,
width = 8, height = 6, dpi = 300)   # all one line

############################
## GSEA for KEGG Pathways ## ranked_genes_deseq2
############################

> library(AnnotationDbi)
> library(org.Mm.eg.db)
```

```
# These two vectors are the results from the previous code:
# ranked_genes_deseq2: Named numeric vector (names = VERSIONED Ensembl IDs, values =
ranking_score)
# new_keys_ranked: Character vector (UNVERSIONED Ensembl IDs derived from
names(ranked_genes_deseq2))

## 1. Create a ranked numeric vector with UNVERSIONED Ensembl IDs as names:
> deseq2_scores_unversioned <- ranked_genes_deseq2
> names(deseq2_scores_unversioned) <- new_keys_ranked

> cat("Glimpse deseq2_scores_unversioned (names are UNVERSIONED Ensembl IDs):\n")
> print(head(deseq2_scores_unversioned))

## 2. Ensembl (UNVERSIONED) to Entrez mapping:
> entrez_ids <- mapIds(org.Mm.eg.db,
                keys    = names(deseq2_scores_unversioned), # Use UNVERSIONED Ensembl IDs
as keys
                keytype = "ENSEMBL",
                column  = "ENTREZID",
                multiVals = "first") # Take the first Entrez ID if multiples exist

> cat("Glimpse entrez_ids (names are UNVERSIONED Ensembl IDs, values are Entrez IDs):\n")
> print(head(entrez_ids))

## 3. Filter out unmapped genes and create geneList_for_kegg:
# We need to filter both the scores AND the Entrez IDs based on successful mapping.
# The `entrez_ids` vector has NAs for unmapped genes, and its names are already aligned
# with `deseq2_scores_unversioned`.

# Filter the unversioned scores to keep only those that successfully mapped to an Entrez ID:
> geneList_for_kegg_filtered_scores <- deseq2_scores_unversioned[!is.na(entrez_ids)]

# Filter the Entrez IDs to keep only the mapped ones:
> mapped_entrez_ids <- entrez_ids[!is.na(entrez_ids)]

# Now, create geneList_for_kegg by setting the names of the filtered scores
# to the corresponding mapped Entrez IDs.
# The order of names in `geneList_for_kegg_filtered_scores` and `mapped_entrez_ids`
# is implicitly aligned because they both derived from the same filtering based on `entrez_ids`:
> geneList_for_kegg <- setNames(geneList_for_kegg_filtered_scores, mapped_entrez_ids)

> cat("Glimpse geneList_for_kegg (names are Entrez IDs, values are scores, before duplicate
handling):\n")
> print(head(geneList_for_kegg))

## 4. Collapse duplicates: keep the entry with the largest |score|
> geneList_for_kegg <- tapply(geneList_for_kegg,
                      names(geneList_for_kegg),
                      function(z) z[which.max(abs(z))]) |>
                      unlist()

## 5. ensure numeric class and decreasing order
> geneList_for_kegg <- setNames(as.numeric(geneList_for_kegg), names(geneList_for_kegg))
> geneList_for_kegg <- sort(geneList_for_kegg, decreasing = TRUE)
```

```
> cat("Final geneList_for_kegg (sorted and ready for gseKEGG):\n")
> print(head(geneList_for_kegg))

# --- Run gseKEGG ---
> library(clusterProfiler)
> library(DOSE)

# You typically don't need a `gene_sets_for_gsea` for gseKEGG
# unless you're providing a custom TERM2GENE.
# gseKEGG fetches pathways automatically based on 'organism'.

> deseq2_gsea_results_kegg <- gseKEGG(geneList = geneList_for_kegg,
                                      organism = 'mmu', # Use appropriate KEGG organism code
                                      pvalueCutoff = 0.05,
                                      pAdjustMethod = "BH",
                                      minGSSize = 10,
                                      maxGSSize = 500,
                                      eps = 1e-10)

# Check that your results are not NULL:
> cat("GSEA KEGG Results (partial view):\n")

# Check if there are any results before trying to print
> if (length(deseq2_gsea_results_kegg@result$ID) > 0) {
  print(head(deseq2_gsea_results_kegg@result))
} else {
  message("No KEGG terms enriched under the specified pvalueCutoff.")
}

> deseq2_gsea_kegg <- dotplot(deseq2_gsea_results_kegg, showCategory = 5)
> deseq2_gsea_kegg <- deseq2_gsea_kegg + labs(x = "Normalized Enrichment Score (NES)",
title = "GSEA of KEGG Pathways")

# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/gsea/deseq2_gsea_kegg_dotplot.png", plot =
deseq2_gsea_kegg, width = 6, height = 5, dpi = 300)  # all one line

######################
# GSEA for Reactome ## deseq2_results_table_filtered
######################

# Load necessary libraries first
> library(clusterProfiler)
> library(ReactomePA) # This package provides gsePathway for Reactome GSEA
> library(org.Mm.eg.db) # For mouse annotations, crucial for ID conversion
> library(dplyr) # For data manipulation (e.g., arrange, filter)
> library(tibble) # For rownames_to_column if your DESeq2 results have row names as gene
IDs

# --- 1. Create the ranked gene list (geneList) for GSEA ---
# GSEA geneList format: numeric vector with names as Entrez IDs, sorted decreasingly by
statistic.
```

```
# Calculate the ranking score and create the initial ranked list with Ensembl IDs
> ranked_genes_ensembl <- deseq2_results_table_filtered %>%
  # Explicitly convert to a data frame or tibble at the start of the chain
  as.data.frame() %>% # Add this line
  # If your ENSEMBL IDs are in row names, bring them into a column for easier dplyr
manipulation
  rownames_to_column(var = "gene_id") %>%
  # Select the gene_id, log2FoldChange, and padj columns
  dplyr::select(gene_id, log2FoldChange, padj) %>%
  # Remove any rows with NA values in these critical columns for ranking
  na.omit() %>% # This na.omit will now definitely operate on a data frame
  # Create the GSEA ranking metric: sign(log2FoldChange) * (-log10(padj))
  dplyr::mutate(ranking_score = sign(log2FoldChange) * (-log10(padj))) %>%
  # Arrange the genes by this ranking_score in decreasing order
  dplyr::arrange(desc(ranking_score)) %>%
  # Extract the ranking_score as a named vector, with gene_id as names
  dplyr::pull(ranking_score, name = gene_id)

# You can inspect the new 'ranked_genes_ensembl'
> cat("Length of ranked_genes_ensembl (after scoring):", length(ranked_genes_ensembl), "\n")
> cat("Number of names in ranked_genes_ensembl (after scoring):",
length(names(ranked_genes_ensembl)), "\n")
> cat("First few names (after scoring):", head(names(ranked_genes_ensembl)), "\n")
> cat("First few values (after scoring):", head(ranked_genes_ensembl), "\n")

# Remove any NAs from the ranking statistic
> ranked_genes_ensembl <- na.omit(ranked_genes_ensembl)

# Remove the version number (e.g., ".2" etc) from the Ensembl gene IDs in row names
> names(ranked_genes_ensembl) <- gsub("\\.\\d+$", "", names(ranked_genes_ensembl))

# Map Ensembl IDs to Entrez IDs
# This uses the org.Mm.eg.db annotation package to perform the mapping
> entrez_map <- mapIds(org.Mm.eg.db,
                       keys = names(ranked_genes_ensembl),
                       keytype = "ENSEMBL",
                       column = "ENTREZID",
                       multiVals = "first") # If one ENSEMBL maps to multiple ENTREZ, take the first

# Filter out genes that could not be mapped to Entrez IDs
> entrez_map <- na.omit(entrez_map)

# Keep only the genes that have a valid Entrez ID mapping in our ranked list
> ranked_genes_entrez_initial <- ranked_genes_ensembl[names(entrez_map)]
> names(ranked_genes_entrez_initial) <- entrez_map # Assign Entrez IDs as names

# Handle duplicate Entrez IDs: keep the one with the highest absolute statistic
# This is crucial for GSEA, as it expects unique gene identifiers
> ranked_genes_entrez_sorted_abs <-
ranked_genes_entrez_initial[order(abs(ranked_genes_entrez_initial), decreasing = TRUE)]
> geneList_for_reactome <- ranked_genes_entrez_sorted_abs[!
duplicated(names(ranked_genes_entrez_sorted_abs))]
```

```
# Finally, sort the gene list by the statistic value itself in decreasing order
> geneList_for_reactome <- sort(geneList_for_reactome, decreasing = TRUE)

# You can inspect the geneList:
> head(geneList_for_reactome)
> str(geneList_for_reactome)
> length(geneList_for_reactome)

# --- 2. Perform Reactome GSEA using gsePathway ---
# Note: gsePathway is part of the ReactomePA package, which you loaded earlier.

> gsea_results_reactome <- gsePathway(
  geneList      = geneList_for_reactome,
  organism      = "mouse", # Use "mouse" for ReactomePA mouse pathways
  pvalueCutoff  = 0.05,
  minGSSize     = 10,
  maxGSSize     = 500,
  eps           = 0 # Set to 0 to avoid issues with very small values if needed
)

# --- 3. Check results and visualize ---
> if (is.null(gsea_results_reactome) || nrow(gsea_results_reactome@result) == 0) {
  print("No significant Reactome pathways found by GSEA.")
} else {
  print("GSEA for Reactome pathways completed. Top results:")
  print(head(as.data.frame(gsea_results_reactome)))

  # Optional: Visualize results (requires enrichplot)
  > library(enrichplot)

  # Dotplot of top Reactome pathways
  > deseq2_reactome_gsea_dotplot <- dotplot(gsea_results_reactome, showCategory = 10,
title = "Reactome GSEA Dotplot")

# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/gsea/deseq2_reactome_gsea_dotplot.png",
deseq2_reactome_gsea_dotplot, width = 7, height = 7, dpi = 300)   # all one line


  # Enriched map of top Reactome pathways (shows relationships between pathways)
  > deseq2_reactome_gsea_emapplot <- emapplot(pairwise_termsim(gsea_results_reactome),
showCategory = 10)

# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/gsea/deseq2_reactome_gsea_emapplot.png",
deseq2_reactome_gsea_emapplot, width = 7, height = 7, dpi = 300)


  # Gene-concept network (shows genes involved in pathways)
  > deseq2_reactome_gsea_cnetplot <- cnetplot(gsea_results_reactome,
categorySize="pvalue", foldChange=geneList_for_reactome, showCategory = 5)

# You may want to modify your file path and filename:
```

```
> ggsave("mouse_data/cluster/gsea/deseq2_reactome_gsea_cnetplot.png",
deseq2_reactome_gsea_cnetplot, width = 7, height = 7, dpi = 300)

}

#########################
# edgeR plots for GSEA ##
#########################

###################################
### GSEA for GO Terms with edgeR ###
###################################

# BiocManager::install("msigdbr")

# Load R packages
> library(dplyr)
> library(tibble)
> library(edgeR)
> library(msigdbr)
> library(AnnotationDbi)
> library(org.Mm.eg.db)
> library(enrichplot)
> library(clusterProfiler)
> library(ggplot2)

# Rank edgeR results:
> ranked_genes_edger <- as.data.frame(top_tags_lrt) %>%
  rownames_to_column(var = "gene_id") %>%
  # Select the necessary columns
  dplyr::select(gene_id, logFC, FDR) %>%
  na.omit() %>%
  # Create a ranking metric: sign(logFC) * -log10(FDR)
  dplyr::mutate(ranking_score = sign(logFC) * (-log10(FDR))) %>%
  dplyr::arrange(desc(ranking_score)) %>%
  # Pull the ranking score with gene IDs as names
  dplyr::pull(ranking_score, name = gene_id)

> cat("Glimpse ranked genes edgeR:")
> head(ranked_genes_edger)

#####################################
# GSEA for GO gene sets with edgeR ##
#####################################

# Remove the version number (e.g., ".2" etc) from the Ensembl gene IDs in row names
> names(ranked_genes_edger) <- gsub("\\.\\d+$", "", names(ranked_genes_edger))

# For GSEA with MSigDB GO gene sets, map Ensembl IDs to gene symbols:
> ranked_genes_symbol <- mapIds(org.Mm.eg.db,
                          keys = names(ranked_genes_edger),
                          keytype = "ENSEMBL",
                          column = "SYMBOL")
```

```
# Remove unmapped IDs
> ranked_genes_symbol <- ranked_genes_symbol[!is.na(ranked_genes_symbol)]

# Create a ranked list with gene symbols, handling duplicates
> ranked_list_for_gsea_symbol <- ranked_genes_edger[names(ranked_genes_edger) %in%
names(ranked_genes_symbol)]
> mapped_symbols <- sapply(ranked_genes_symbol[names(ranked_list_for_gsea_symbol)], '[',
1)

# Last check: Filter out any NA symbols and then assign names
> valid_symbols_idx <- !is.na(mapped_symbols)
> ranked_list_for_gsea_symbol <- ranked_list_for_gsea_symbol[valid_symbols_idx]
> mapped_symbols <- mapped_symbols[valid_symbols_idx]
> names(ranked_list_for_gsea_symbol) <- mapped_symbols

# Remove duplicate symbols in the ranked list, keeping the one with the highest absolute score
> ranked_list_for_gsea_symbol_unique <- tapply(ranked_list_for_gsea_symbol,
names(ranked_list_for_gsea_symbol), function(x) x[which.max(abs(x))])

# Clean up names after tapply
> ranked_list_for_gsea_symbol_unique <- ranked_list_for_gsea_symbol_unique[!
is.na(names(ranked_list_for_gsea_symbol_unique)) ]
> names(ranked_list_for_gsea_symbol_unique) <-
toupper(names(ranked_list_for_gsea_symbol_unique))

# Convert to a named vector and sort in decreasing order, and remove any NAs
###ranked_list_for_gsea_symbol_unique <- as.vector(ranked_list_for_gsea_symbol_unique)
> ranked_list_for_gsea_symbol_unique <-
  setNames(as.numeric(ranked_list_for_gsea_symbol_unique),
           names(ranked_list_for_gsea_symbol_unique))
> ranked_list_for_gsea_symbol_unique <- na.omit(ranked_list_for_gsea_symbol_unique)
#remove any remaining NAs
> ranked_list_for_gsea_symbol_unique <- sort(ranked_list_for_gsea_symbol_unique,
decreasing = TRUE)

# 3. Get gene sets for GSEA (using MSigDB as an example)
# Get MSigDB GO gene sets (using symbols)

> gene_sets_for_gsea <- msigdbr(
  species     = "Mus musculus",
  collection  = "C5"#,
  # subcategory = "GO:BP"        # optional – pick the GO branch
  ) %>%
  dplyr::select(gs_name, gene_symbol) %>%
  dplyr::mutate(gene_symbol = toupper(as.character(gene_symbol)))

# test for errors (overlap > 0 is correct; overlap = 0 is an error)
> overlap <- intersect(names(ranked_list_for_gsea_symbol_unique),
                       gene_sets_for_gsea$gene_symbol)
> print(length(overlap)) # [1] 13007

# Perform GSEA using symbols
> gsea_results_go_symbol <- GSEA(geneList = ranked_list_for_gsea_symbol_unique,
```

```
                        TERM2GENE = gene_sets_for_gsea,
                        pvalueCutoff = 0.05,
                        pAdjustMethod = "BH",
                        minGSSize = 10,
                        maxGSSize = 500,
                        eps = 1e-10)

# Visualization

> gsea1_go <- dotplot(gsea_results_go_symbol, showCategory = 10)
> gsea2_go <- emapplot(
  pairwise_termsim(gsea_results_go_symbol),
  showCategory = 10,
  layout = "fr"
) +
  theme_bw()

# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/gsea/edger_gsea_go_dotplot.png", plot = gsea1_go, width
= 6, height = 5, dpi = 300)

# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/gsea/edger_gsea__go_emapplot.png", plot = gsea2_go,
width = 7, height = 6, dpi = 300)

############################
## GSEA for KEGG Pathways ## ranked_genes_edger
############################

> library(AnnotationDbi)
> library(org.Mm.eg.db)

## 1. ranked numeric vector from edger (names = Ensembl IDs)
> edger_scores <- ranked_genes_edger

## 2. Ensembl to Entrez (first hit only, suppress NA)
> entrez_ids <- mapIds(org.Mm.eg.db,
                keys     = names(edger_scores),
                keytype  = "ENSEMBL",
                column   = "ENTREZID",
                multiVals = "first")

## 3. keep mapped genes and attach Entrez IDs as names
> geneList_for_kegg <- setNames(edger_scores[!is.na(entrez_ids)],
                        entrez_ids[!is.na(entrez_ids)])

## 4. collapse duplicates: keep the entry with the largest |score|
> geneList_for_kegg <- tapply(geneList_for_kegg,
                        names(geneList_for_kegg),
                        function(z) z[which.max(abs(z))]) %>%
                        unlist()

## 5. ensure numeric class and decreasing order
```

```
> geneList_for_kegg <- setNames(as.numeric(geneList_for_kegg), names(geneList_for_kegg))
> geneList_for_kegg <- sort(geneList_for_kegg, decreasing = TRUE)

> edger_gsea_results_kegg <- gseKEGG(geneList = geneList_for_kegg,
                             organism = 'mmu', # Use appropriate KEGG organism code
                             pvalueCutoff = 0.05,
                             pAdjustMethod = "BH",
                             minGSSize = 10,
                             maxGSSize = 500,
                             eps = 1e-10)

> edger_gsea_results_kegg <- dotplot(edger_gsea_results_kegg, showCategory = 5)

# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/gsea/edger_gsea_kegg_dotplot.png", plot =
edger_gsea_results_kegg, width = 6, height = 5, dpi = 300)  # all one line

######################
# GSEA for Reactome ## top_tags_lrt
######################

# load R packages
> library(clusterProfiler)
> library(ReactomePA) # This package provides gsePathway for Reactome GSEA
> library(org.Mm.eg.db) # For mouse annotations, crucial for ID conversion
> library(dplyr) # For data manipulation (e.g., arrange, filter)
> library(tibble) # For rownames_to_column if your edgeR results have row names as gene IDs

# --- 1. Prepare your input data: top_tags_lrt ---

# --- 2. Create the ranked gene list (geneList) for GSEA ---
# GSEA geneList format: numeric vector with names as Entrez IDs, sorted decreasingly by
statistic.

# Calculate the ranking score and create the initial ranked list with Ensembl IDs
> ranked_genes_ensembl <- top_tags_lrt %>%
  # Explicitly convert to a data frame or tibble at the start of the chain
  as.data.frame() %>% # Add this line
  # If your ENSEMBL IDs are in row names, bring them into a column for easier dplyr
manipulation
  rownames_to_column(var = "gene_id") %>%
  # Select the gene_id, logFC, and FDR columns
  dplyr::select(gene_id, logFC, FDR) %>%
  # Remove any rows with NA values in these critical columns for ranking
  na.omit() %>% # This na.omit will now definitely operate on a data frame
  # Create the GSEA ranking metric: sign(log2FoldChange) * (-log10(padj))
  dplyr::mutate(ranking_score = sign(logFC) * (-log10(FDR))) %>%
  # Arrange the genes by this ranking_score in decreasing order
  dplyr::arrange(desc(ranking_score)) %>%
  # Extract the ranking_score as a named vector, with gene_id as names
  dplyr::pull(ranking_score, name = gene_id)

# You can inspect the new 'ranked_genes_ensembl'
```

```
> cat("Length of ranked_genes_ensembl (after scoring):", length(ranked_genes_ensembl), "\n")
> cat("Number of names in ranked_genes_ensembl (after scoring):",
length(names(ranked_genes_ensembl)), "\n")
> cat("First few names (after scoring):", head(names(ranked_genes_ensembl)), "\n")
> cat("First few values (after scoring):", head(ranked_genes_ensembl), "\n")

# Remove any NAs from the ranking statistic
> ranked_genes_ensembl <- na.omit(ranked_genes_ensembl)

# Remove the version number (e.g., ".2" etc) from the Ensembl gene IDs in row names
> names(ranked_genes_ensembl) <- gsub("\\.\\d+$", "", names(ranked_genes_ensembl))

# Map Ensembl IDs to Entrez IDs
# This uses the org.Mm.eg.db annotation package to perform the mapping
> entrez_map <- mapIds(org.Mm.eg.db,
                       keys = names(ranked_genes_ensembl),
                       keytype = "ENSEMBL",
                       column = "ENTREZID",
                       multiVals = "first") # If one ENSEMBL maps to multiple ENTREZ, take the first

# Filter out genes that could not be mapped to Entrez IDs
> entrez_map <- na.omit(entrez_map)

# Keep only the genes that have a valid Entrez ID mapping in our ranked list
> ranked_genes_entrez_initial <- ranked_genes_ensembl[names(entrez_map)]
> names(ranked_genes_entrez_initial) <- entrez_map # Assign Entrez IDs as names

# Handle duplicate Entrez IDs: keep the one with the highest absolute statistic
# This is crucial for GSEA, as it expects unique gene identifiers
> ranked_genes_entrez_sorted_abs <-
ranked_genes_entrez_initial[order(abs(ranked_genes_entrez_initial), decreasing = TRUE)]
> geneList_for_reactome <- ranked_genes_entrez_sorted_abs[!
duplicated(names(ranked_genes_entrez_sorted_abs))]

# Finally, sort the gene list by the statistic value itself in decreasing order
> geneList_for_reactome <- sort(geneList_for_reactome, decreasing = TRUE)

# You can inspect the geneList:
> head(geneList_for_reactome)
> str(geneList_for_reactome)
> length(geneList_for_reactome)

# --- 3. Perform Reactome GSEA using gsePathway ---
# Note: gsePathway is part of the ReactomePA package, which you loaded earlier.

> gsea_results_reactome <- gsePathway(
  geneList      = geneList_for_reactome,
  organism      = "mouse", # Use "mouse" for ReactomePA mouse pathways
  pvalueCutoff  = 0.05,
  minGSSize     = 10,
  maxGSSize     = 500,
  eps           = 0 # Set to 0 to avoid issues with very small values if needed
)
```

```
# --- 4. Check results and visualize ---
> if (is.null(gsea_results_reactome) || nrow(gsea_results_reactome@result) == 0) {
  print("No significant Reactome pathways found by GSEA.")
} else {
  print("GSEA for Reactome pathways completed. Top results:")
  print(head(as.data.frame(gsea_results_reactome)))

  # Optional: Visualize results (requires enrichplot)
 library(enrichplot)

  # Dotplot of top Reactome pathways
> edger_reactome_gsea_dotplot <- dotplot(gsea_results_reactome, showCategory = 10, title =
"Reactome GSEA Dotplot")

# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/gsea/edger_reactome_gsea_dotplot.png",
edger_reactome_gsea_dotplot, width = 7, height = 7, dpi = 300)  # all one line


  # Enriched map of top Reactome pathways (shows relationships between pathways)
> edger_reactome_gsea_emapplot <- emapplot(pairwise_termsim(gsea_results_reactome),
showCategory = 10)

# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/gsea/edger_reactome_gsea_emapplot.png",
edger_reactome_gsea_emapplot, width = 7, height = 7, dpi = 300)  # all one line


  # Gene-concept network (shows genes involved in pathways)
> edger_reactome_gsea_cnetplot <- cnetplot(gsea_results_reactome, categorySize="pvalue",
foldChange=geneList_for_reactome, showCategory = 5)

# You may want to modify your file path and filename:
> ggsave("mouse_data/cluster/gsea/edger_reactome_gsea_cnetplot.png",
edger_reactome_gsea_cnetplot, width = 7, height = 7, dpi = 300)

}

# End
```