Package Downloads

```r
# Run this chunk top-to-bottom in a fresh R session to ensure all required
# packages are available. It safely skips anything already installed.

cran_repo <- "https://cloud.r-project.org"
options(repos = c(CRAN = cran_repo))

if (!requireNamespace("BiocManager", quietly = TRUE)) {
  install.packages("BiocManager", repos = cran_repo)
}

BiocManager::install(version = "3.21", ask = FALSE, update = FALSE)

install_if_missing <- function(pkgs, installer) {
  missing_pkgs <- pkgs[!vapply(pkgs, requireNamespace,
                               FUN.VALUE = logical(1), quietly = TRUE)]
  if (length(missing_pkgs) > 0) {
    installer(missing_pkgs)
  }
  invisible(NULL)
}

install_if_missing(
  pkgs = c("devtools", "GenomeInfoDb", "tidyverse"),
  installer = function(pkgs) install.packages(pkgs, repos = cran_repo)
)

install_if_missing(
  pkgs = c("DESeq2", "org.Hs.eg.db"),
  installer = function(pkgs) BiocManager::install(pkgs, ask = FALSE, update = FALSE)
)
```

Section 1

Section 1a

```r
# Attach the library
library(org.Hs.eg.db)
```

```
## Loading required package: AnnotationDbi

## Loading required package: stats4

## Loading required package: BiocGenerics

## Loading required package: generics
```

```
##
## Attaching package: 'generics'

## The following objects are masked from 'package:base':
##
##     as.difftime, as.factor, as.ordered, intersect, is.element, setdiff,
##     setequal, union

##
## Attaching package: 'BiocGenerics'

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

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

## Loading required package: Biobase

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

## Loading required package: IRanges

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:utils':
##
##     findMatches

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

##
## Attaching package: 'IRanges'
## The following object is masked from 'package:grDevices':
##
##     windows

##
```

```r
# We will need this so we can use the pipe: %>%
library(magrittr)

# Create the data folder if it doesn't exist
if (!dir.exists("data")) {
  dir.create("data")
}

# Define the file path to the plots directory
plots_dir <- "plots"

# Create the plots folder if it doesn't exist
if (!dir.exists(plots_dir)) {
  dir.create(plots_dir)
}

# Define the file path to the results directory
results_dir <- "results"

# Create the results folder if it doesn't exist
if (!dir.exists(results_dir)) {
  dir.create(results_dir)
}

# Define the file path to the data directory
data_dir <- file.path("data", "SRP192714")

# Declare the file path to the gene expression matrix file
data_file <- file.path(data_dir, "SRP192714.tsv")

# Read in data TSV file
expression_df <- readr::read_tsv(data_file) %>%
  tibble::column_to_rownames("Gene")
```

```
## Rows: 43363 Columns: 1022

## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr    (1): Gene
## dbl (1021): SRR8907879, SRR8907880, SRR8907881, SRR8907882, SRR8907883, SRR8...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Load refine.bio metadata
refinebio_meta <- readr::read_tsv(file.path(data_dir, "metadata_SRP192714.tsv"))
```

```
## Rows: 1021 Columns: 25
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (11): refinebio_accession_code, experiment_accession, refinebio_organism...
## dbl  (3): refinebio_age, refinebio_processor_id, MetaSRA_age
## lgl (11): refinebio_cell_line, refinebio_compound, refinebio_developmental_s...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
rownames(refinebio_meta) <- refinebio_meta$refinebio_accession_code
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```
# Load GEO metadata (CSV)
geo_meta <- readr::read_csv("data/GSE129882_PhenoData.transcript.csv")
```

```
## New names:
## Rows: 261 Columns: 21
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (15): ...1, Sample, Time, Sex, DENV.at.Inception, DENV.Exposure.at.Time.... dbl
## (6): Patient, Visit, Age, DENV.Infections, lib.size, norm.factors
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
# Merge on 'refinebio_title' (refine.bio) and 'Sample' (GEO), keep 'refinebio_title' as column name
if ("Sample" %in% colnames(geo_meta) && "refinebio_title" %in% colnames(refinebio_meta)) {
  merged_meta <- dplyr::left_join(refinebio_meta, geo_meta, by = c("refinebio_title" = "Sample"))
} else {
  merged_meta <- refinebio_meta
}

# Choose how many samples to keep for this run (set to Inf to keep all samples)
sample_limit <- 100

all_samples <- colnames(expression_df)
selected_samples <- all_samples
if (is.finite(sample_limit)) {
  selected_samples <- all_samples[seq_len(min(sample_limit, length(all_samples)))]
}

is_trimmed_run <- length(selected_samples) < length(all_samples)
run_label <- if (is_trimmed_run) paste0("trimmed_", length(selected_samples)) else "full"

expression_df <- expression_df[, selected_samples, drop = FALSE]

merged_meta <- merged_meta %>%
  dplyr::filter(refinebio_accession_code %in% selected_samples)
rownames(merged_meta) <- merged_meta$refinebio_accession_code
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```r
merged_meta_path <- file.path(data_dir, "metadata_SRP192714_merged.tsv")
readr::write_tsv(merged_meta, merged_meta_path)
```

Section 1b

```r
# Bring back the "Gene" column in preparation for mapping
expression_df <- expression_df %>%
  tibble::rownames_to_column("Gene")

# Map Ensembl IDs to their first mapped Symbol
gene_symbols <- mapIds(
  org.Hs.eg.db,
  keys = expression_df$Gene,
  keytype = "ENSEMBL",
  column = "SYMBOL",
  multiVals = "first"
)
```

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

```r
# Add the mapped symbols as a new column
expression_df$Symbol <- gene_symbols[expression_df$Gene]

# Reorder columns to have Gene, Symbol, then the rest
expression_df <- expression_df %>%
  dplyr::select(Gene, Symbol, dplyr::everything(), -Gene)

# Write mapped data frame to output file
readr::write_tsv(expression_df, file.path(
  results_dir,
  "SRP192714_Symbols.tsv"
))
```

Section 1c

```r
# Get matrix size
matrix_dim <- dim(expression_df)
cat("Expression matrix dimensions (genes x samples):", matrix_dim[1], "x", matrix_dim[2], "\n")
```

```
## Expression matrix dimensions (genes x samples): 43363 x 101
```

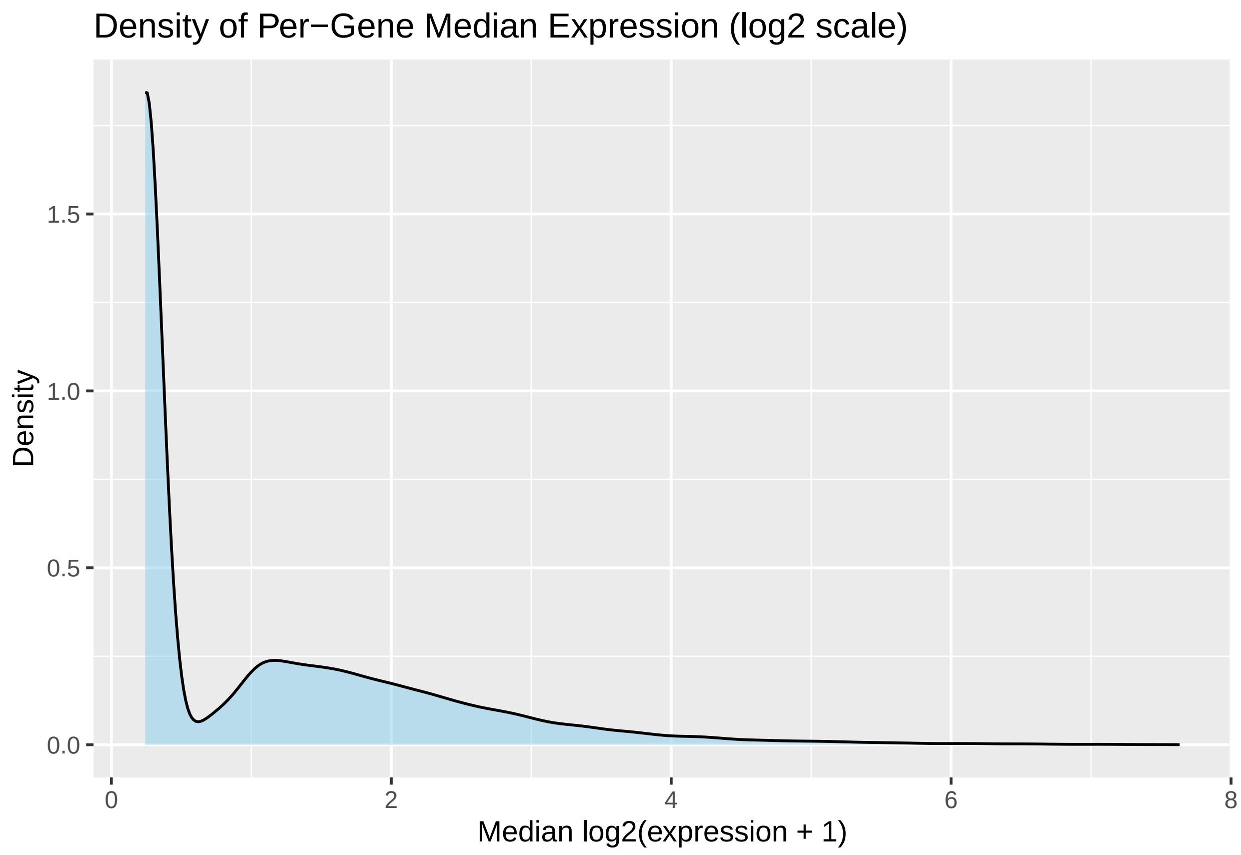
```r
# Number of genes
cat("Number of genes:", nrow(expression_df), "\n")
```

```
## Number of genes: 43363
```

```r
# Select only numeric columns for log transformation
expr_numeric <- expression_df %>% dplyr::select(where(is.numeric))

# Log-scale the data (add pseudocount to avoid log(0))
log_expr <- log2(expr_numeric + 1)

# Calculate per-gene median expression
gene_medians <- apply(log_expr, 1, median, na.rm = TRUE)

# Show summary statistics for gene medians
summary(gene_medians)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2416  0.2442  0.2696  1.0726  1.7001  7.6328
```

```r
# Density plot of per-gene median expression
library(ggplot2)
plot_obj <- ggplot(data.frame(median=gene_medians), aes(x=median)) +
  geom_density(fill="skyblue", alpha=0.5) +
  labs(title="Density of Per-Gene Median Expression (log2 scale)",
       x="Median log2(expression + 1)",
       y="Density")

# Save plot to the plots directory
plot_path <- file.path(plots_dir, "per_gene_median_density.png")
ggsave(plot_path, plot=plot_obj, width=6, height=4, dpi=300)

# Also print the plot in the notebook
plot_obj
```



The dataset contains 43,363 genes measured across 101 samples. The summary statistics show that most genes have low median expression (median ≈ 0.27 on the log2 scale), with a long tail of higher expression values (max ≈ 7.63). From our research, this is typical for transcriptome data, where a small number of genes are highly expressed while the majority have low expression. The density plot visualizes this distribution, showing a peak at lower expression values and a gradual decline towards higher values. Log transformation helps to reduce the impact of extreme values and makes the distribution more interpretable. # Section 2 ## Section 2a-c

```
suppressPackageStartupMessages({
  library(DESeq2)
  library(dplyr)
  library(magrittr)
  library(readr)
  library(tibble)
  library(stringr)
})

if (!exists("results_dir")) {
  results_dir <- "results"
  if (!dir.exists(results_dir)) dir.create(results_dir, recursive = TRUE)
}
if (!exists("plots_dir")) {
  plots_dir <- "plots"
  if (!dir.exists(plots_dir)) dir.create(plots_dir, recursive = TRUE)
}
if (!exists("data_dir")) {
```

```r
  data_dir <- file.path("data", "SRP192714")
}

if (!exists("expression_df") || !exists("merged_meta")) {
  sample_limit_local <- if (exists("sample_limit")) sample_limit else 100
  expression_df <- readr::read_tsv(file.path(data_dir, "SRP192714.tsv"), show_col_types = FALSE) %>%
    tibble::column_to_rownames("Gene")

  refinebio_meta <- readr::read_tsv(file.path(data_dir, "metadata_SRP192714.tsv"), show_col_types = FAL
  rownames(refinebio_meta) <- refinebio_meta$refinebio_accession_code

  all_samples <- colnames(expression_df)
  selected_samples <- all_samples
  if (is.finite(sample_limit_local)) {
    selected_samples <- all_samples[seq_len(min(sample_limit_local, length(all_samples)))]
  }

  expression_df <- expression_df[, selected_samples, drop = FALSE]
  merged_meta <- refinebio_meta %>%
    dplyr::filter(refinebio_accession_code %in% selected_samples)
  rownames(merged_meta) <- merged_meta$refinebio_accession_code

  is_trimmed_run <- length(selected_samples) < length(all_samples)
  run_label <- if (is_trimmed_run) paste0("trimmed_", length(selected_samples)) else "full"
}

expr_mat <- expression_df %>% dplyr::select(where(is.numeric))
meta <- merged_meta

common_samples <- intersect(colnames(expr_mat), meta$refinebio_accession_code)
expr_mat <- expr_mat %>% dplyr::select(all_of(common_samples))
meta <- meta %>%
  dplyr::filter(refinebio_accession_code %in% common_samples)
rownames(meta) <- meta$refinebio_accession_code
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```r
cts <- as.matrix(expr_mat)
cts <- cts[, rownames(meta), drop = FALSE]
cts[cts < 0] <- 0
cts <- round(cts)

stopifnot(ncol(cts) == nrow(meta))

plot_title <- sprintf("PCA of %d Samples (colored by Exposure)", length(common_samples))
plot_file <- paste0(run_label, "_pca_exposure.png")

dds <- DESeqDataSetFromMatrix(countData = cts,
                              colData = meta,
                              design = ~ refinebio_title)
```

```
## converting counts to integer mode
## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
```

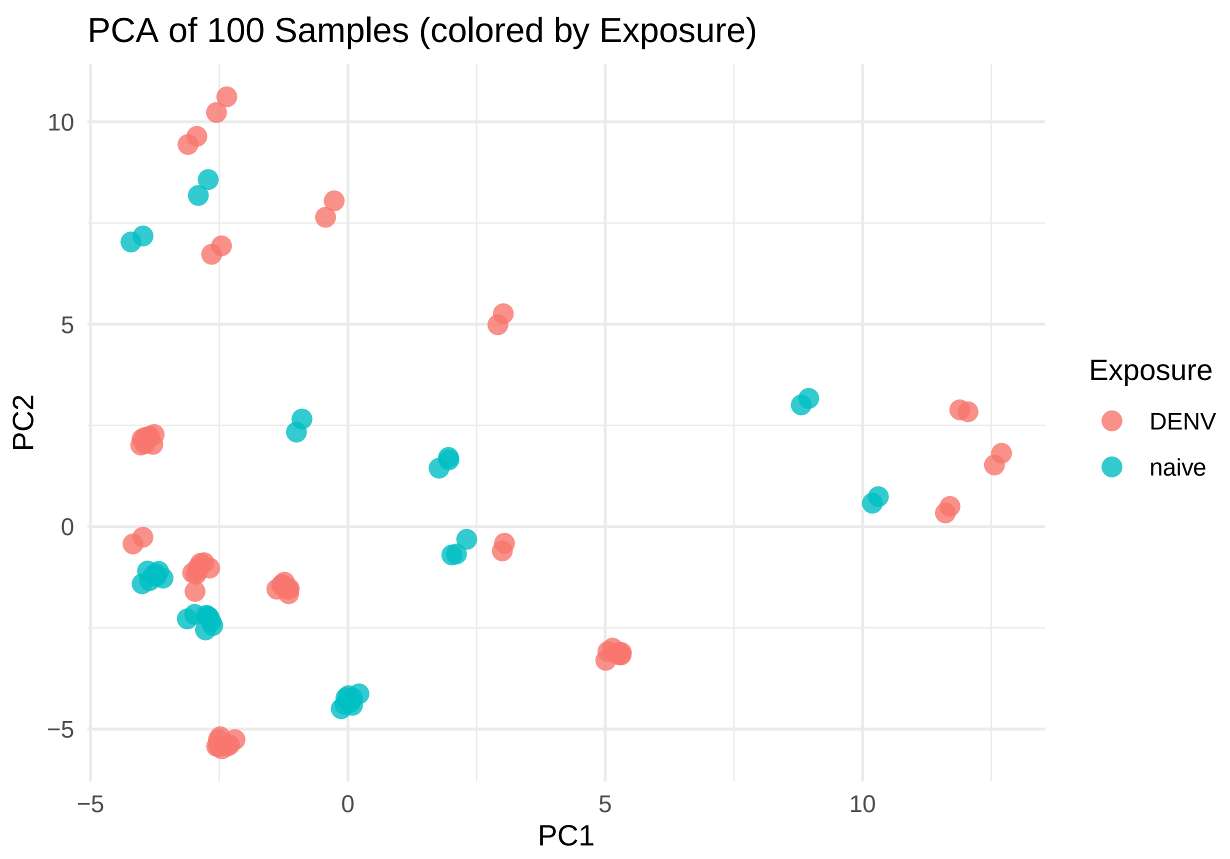
```
dds <- DESeq(dds)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## -- note: fitType='parametric', but the dispersion trend was not well captured by the
##    function: y = a/x + b, and a local regression fit was automatically substituted.
##    specify fitType='local' or 'mean' to avoid this message next time.

## final dispersion estimates

## fitting model and testing
```

```
vsd <- vst(dds, blind = FALSE)
pca_data <- plotPCA(vsd, intgroup = c("Exposure"), returnData = TRUE)
```

```
## using ntop=500 top features by variance
```

```
pca_data$Sample <- rownames(pca_data)
pca_data <- dplyr::left_join(
  pca_data,
  meta %>% dplyr::select(refinebio_accession_code, Exposure),
  by = c("Sample" = "refinebio_accession_code")
)
if (!"Exposure" %in% colnames(pca_data) || all(is.na(pca_data$Exposure))) {
  pca_data$Exposure <- meta$Exposure[match(pca_data$Sample, meta$refinebio_accession_code)]
}
pca_plot <- ggplot(pca_data, aes(x = PC1, y = PC2, color = Exposure)) +
  geom_point(size = 3, alpha = 0.8) +
  labs(title = plot_title, x = "PC1", y = "PC2") +
  theme_minimal()
ggsave(filename = file.path(plots_dir, plot_file), plot = pca_plot, width = 6, height = 4, dpi = 300)
pca_plot
```



Section 2d packages

```r
if (is.null(getOption("repos")) || identical(getOption("repos")[["CRAN"]], "@CRAN@")) {
  options(repos = c(CRAN = "https://cloud.r-project.org"))
}
if (!requireNamespace("M3C", quietly = TRUE)) install.packages("M3C", repos = getOption("repos")[["CRAN
```

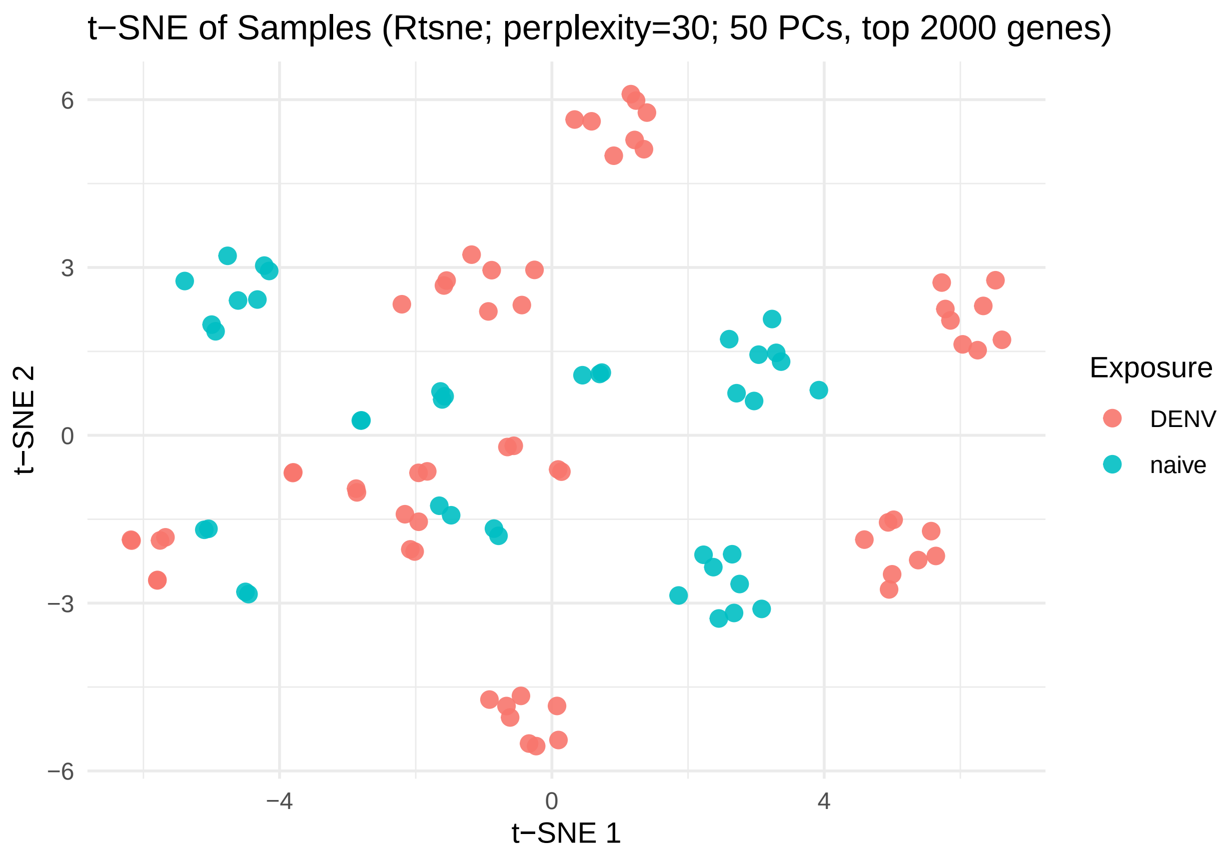
```
## Installing package into 'C:/Users/Taylo/AppData/Local/R/win-library/4.5'
## (as 'lib' is unspecified)
```

```
## Warning: package 'M3C' is not available for this version of R
##
## A version of this package for your version of R might be available elsewhere,
## see the ideas at
## https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages
```

```r
if (!requireNamespace("Rtsne", quietly = TRUE)) install.packages("Rtsne", repos = getOption("repos")[["
if (!requireNamespace("umap", quietly = TRUE)) install.packages("umap", repos = getOption("repos")[["CR
if (!requireNamespace("matrixStats", quietly = TRUE)) install.packages("matrixStats", repos = getOption
```

Section 2d i

```r
## --- 2d i: t-SNE using Rtsne (colored by Exposure) ---
# deps
library(SummarizedExperiment)   # for assay()
library(matrixStats)            # for row/col variance
library(Rtsne)
library(ggplot2)

set.seed(42)

# 1) pull variance-stabilized matrix and align meta
vsd_gxs <- SummarizedExperiment::assay(vsd)        # genes x samples
if (!identical(rownames(meta), colnames(vsd_gxs))) {
  stopifnot("refinebio_accession_code" %in% colnames(meta))
  rownames(meta) <- meta$refinebio_accession_code
  meta <- meta[colnames(vsd_gxs), , drop = FALSE]
}
# keep only labeled samples
keep <- !is.na(meta$Exposure)
vsd_gxs <- vsd_gxs[, keep, drop = FALSE]
meta     <- meta[keep, , drop = FALSE]
meta$Exposure <- factor(meta$Exposure)

# 2) select top variable genes and reduce to ~50 PCs (denoising)
ngenes <- min(2000, nrow(vsd_gxs))                          # 2k or fewer if dataset smaller
top_genes <- head(order(matrixStats::rowVars(vsd_gxs), decreasing = TRUE), ngenes)
X <- t(vsd_gxs[top_genes, , drop = FALSE])                  # samples x genes

pcs   <- prcomp(X, center = TRUE, scale. = TRUE)
pcmat <- pcs$x[, 1:min(50, ncol(pcs$x)), drop = FALSE]       # samples x PCs

# 3) run t-SNE (safe perplexity ~ n/3 clamped to 5..30)
n  <- nrow(pcmat)
px <- max(5, min(30, floor((n - 1) / 3)))
tsne_out <- Rtsne(
  pcmat,
  perplexity = px,
  max_iter = 1000,
  check_duplicates = FALSE,
  verbose = FALSE
)

# 4) plot
tsne_df <- data.frame(
  tSNE1 = tsne_out$Y[, 1],
  tSNE2 = tsne_out$Y[, 2],
  Exposure = meta$Exposure
)

p_tsne <- ggplot(tsne_df, aes(tSNE1, tSNE2, color = Exposure)) +
  geom_point(size = 2.6, alpha = 0.9) +
  labs(
    title = paste0("t-SNE of Samples (Rtsne; perplexity=", px, "; 50 PCs, top ", ngenes, " genes)"),
    x = "t-SNE 1", y = "t-SNE 2"
  ) +
  theme_minimal()

if (!dir.exists(plots_dir)) dir.create(plots_dir, recursive = TRUE)
outfile_tsne <- paste0(run_label, "_tsne_Rtsne.png")
ggsave(file.path(plots_dir, outfile_tsne), p_tsne, width = 7, height = 5, dpi = 300)
p_tsne
```

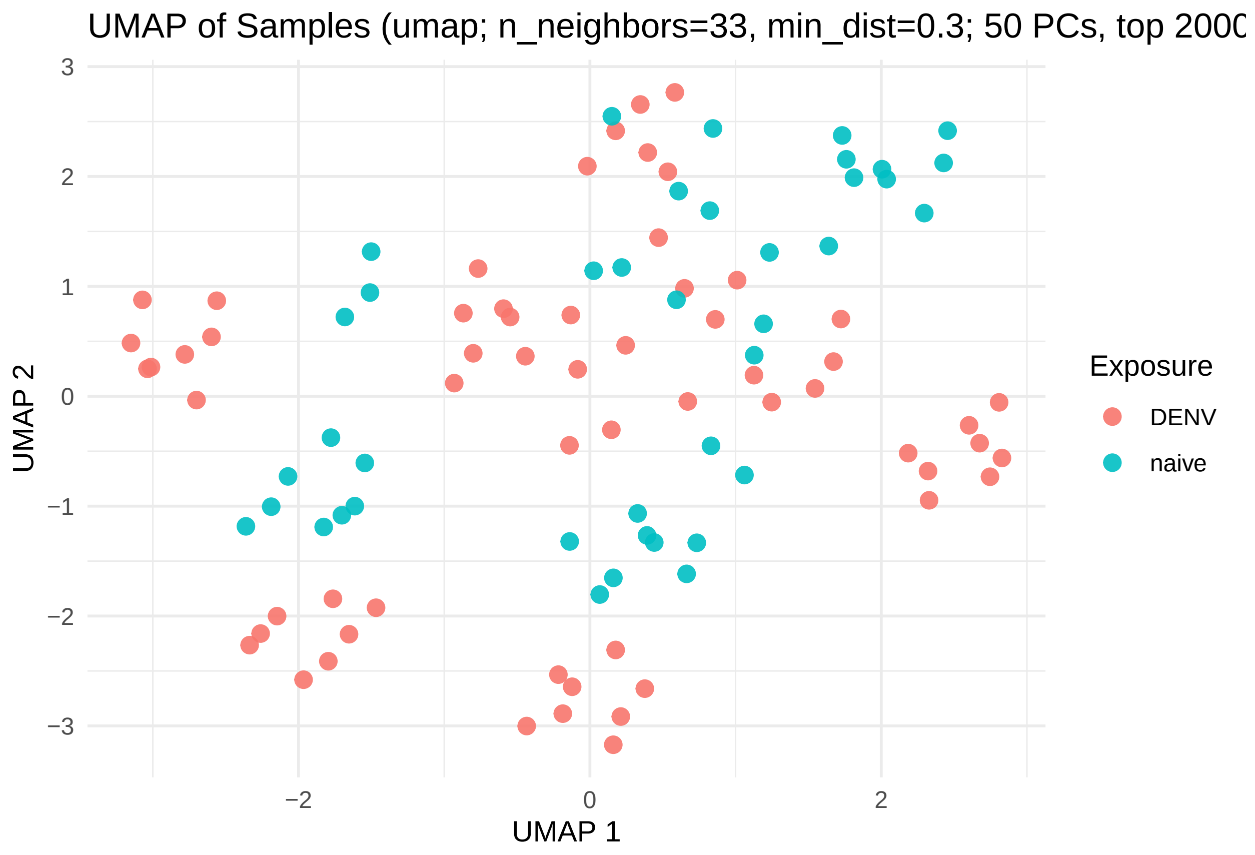


Section 2d ii

```
## --- 2d ii: UMAP using 'umap' (colored by Exposure) ---
library(SummarizedExperiment)
library(matrixStats)
library(umap)
library(ggplot2)

set.seed(42)

vsd_gxs <- SummarizedExperiment::assay(vsd)
if (!identical(rownames(meta), colnames(vsd_gxs))) {
  stopifnot("refinebio_accession_code" %in% colnames(meta))
  rownames(meta) <- meta$refinebio_accession_code
  meta <- meta[colnames(vsd_gxs), , drop = FALSE]
}
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```
keep <- !is.na(meta$Exposure)
vsd_gxs <- vsd_gxs[, keep, drop = FALSE]
meta     <- meta[keep, , drop = FALSE]
meta$Exposure <- factor(meta$Exposure)

ngenes <- min(2000, nrow(vsd_gxs))
top_genes <- head(order(matrixStats::rowVars(vsd_gxs), decreasing = TRUE), ngenes)
X <- t(vsd_gxs[top_genes, , drop = FALSE])

pcs   <- prcomp(X, center = TRUE, scale. = TRUE)
pcmat <- pcs$x[, 1:min(50, ncol(pcs$x)), drop = FALSE]

n <- nrow(pcmat)
nn <- max(10, min(50, round(n / 3)))

cfg <- umap::umap.defaults
cfg$n_neighbors <- nn
cfg$min_dist    <- 0.3
cfg$metric      <- "euclidean"

um <- umap::umap(pcmat, config = cfg)

umap_df <- data.frame(
  UMAP1 = um$layout[, 1],
  UMAP2 = um$layout[, 2],
  Exposure = meta$Exposure
)

p_umap <- ggplot(umap_df, aes(UMAP1, UMAP2, color = Exposure)) +
  geom_point(size = 2.6, alpha = 0.9) +
  labs(
    title = paste0("UMAP of Samples (umap; n_neighbors=", nn, ", min_dist=0.3; 50 PCs, top ", ngenes, "
    x = "UMAP 1", y = "UMAP 2"
  ) +
  theme_minimal()

if (!dir.exists(plots_dir)) dir.create(plots_dir, recursive = TRUE)
outfile_umap <- paste0(run_label, "_umap.png")
ggsave(file.path(plots_dir, outfile_umap), p_umap, width = 7, height = 5, dpi = 300)
p_umap
```



Section 2e

Similarities • All three methods (PCA, t-SNE, UMAP) reduce the high-dimensional gene expression data into 2D, letting us see patterns between samples. • In each, samples tend to group by Exposure status (or whichever grouping variable you're using). • Outliers and variability across samples are visible in all three.

Differences • PCA: Captures global variance structure, shows the main linear directions of variability. Often looks more "spread out" but may miss subtle clusters. • t-SNE: Focuses on local structure, separates clusters more strongly. However, distances between clusters aren't always meaningful (clusters may look far but be closer in high-D space). • UMAP: Balances global and local structure, sometimes keeps a more interpretable overall shape while still revealing clusters.

Section 2f

Findings • Across all three dimensionality reduction methods, samples show grouping consistent with Exposure categories, suggesting exposure status drives significant variation in gene expression. • PCA highlights the major global variance, but t-SNE and UMAP give a clearer view of sub-clusters. • UMAP provides a balance, maintaining both separation of clusters and an interpretable global structure. • Together, these plots confirm that exposure effects are strong and detectable across different approaches.

Section 3a-b

```r
if (is.null(getOption("repos")) || identical(getOption("repos")[["CRAN"]], "@CRAN@")) {
  options(repos = c(CRAN = "https://cloud.r-project.org"))
}
if (!requireNamespace("ggrepel", quietly = TRUE)) {
  install.packages("ggrepel", repos = getOption("repos")["CRAN"])
}
```

```r
metadata_file <- file.path(data_dir, "metadata_SRP192714_merged.tsv")
metadata <- readr::read_tsv(metadata_file)
```

```
## New names:
## Rows: 100 Columns: 45
## -- Column specification
## ----------------------------------------------------------- Delimiter: "\t" chr
## (25): refinebio_accession_code, experiment_accession, refinebio_organism... dbl
## (9): refinebio_age, refinebio_processor_id, MetaSRA_age, Patient, Visit... lgl
## (11): refinebio_cell_line, refinebio_compound, refinebio_developmental_s...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `...1` -> `...26`
```

```r
# ---- 3a-b: Differential expression: DENV vs Naive ----
suppressPackageStartupMessages({
  library(readr); library(dplyr); library(stringr); library(tibble)
  library(DESeq2); library(ggplot2)
})

# paths (match earlier sections)
expr_path <- "data/SRP192714/SRP192714.tsv"
meta_path <- "data/SRP192714/metadata_SRP192714_merged.tsv"

# 1) read counts and merged metadata
counts <- readr::read_tsv(expr_path, show_col_types = FALSE) |>
  tibble::column_to_rownames("Gene") |>
  as.matrix()
mode(counts) <- "numeric"

metadata <- readr::read_tsv(meta_path, show_col_types = FALSE)
```

```
## New names:
## * `...1` -> `...26`
```

```r
if (exists("selected_samples")) {
  keep_ids <- intersect(colnames(counts), selected_samples)
  counts <- counts[, keep_ids, drop = FALSE]
  metadata <- metadata %>% dplyr::filter(refinebio_accession_code %in% keep_ids)
}

# 2) make a clean two-level grouping: Exposure2 = Naive vs DENV
```

```r
stopifnot(all(c("refinebio_accession_code","Exposure") %in% names(metadata)))
metadata <- metadata |>
  mutate(
    Exposure2 = case_when(
      str_to_lower(Exposure) %in% c("naive","mock","control","uninfected","healthy") ~ "Naive",
      str_detect(str_to_lower(Exposure), "denv|dengue") ~ "DENV",
      TRUE ~ NA_character_
    )
  )

# 3) align samples (use only samples with Exposure2)
metadata <- metadata |> filter(!is.na(Exposure2))
common_ids <- intersect(colnames(counts), metadata$refinebio_accession_code)
if (length(common_ids) == 0) stop("No overlapping sample IDs between counts and metadata.")

# subset BOTH objects to the same ids and order identically
common_ids <- sort(common_ids)  # deterministic order
counts  <- counts[, common_ids, drop = FALSE]
metadata <- metadata |>
  filter(refinebio_accession_code %in% common_ids) |>
  arrange(match(refinebio_accession_code, common_ids))

stopifnot(identical(colnames(counts), metadata$refinebio_accession_code))

# finalize colData
metadata$Exposure2 <- factor(metadata$Exposure2, levels = c("Naive","DENV"))
rownames(metadata) <- metadata$refinebio_accession_code
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```r
# 4) basic QC on counts
counts[counts < 0] <- 0
counts <- round(counts)
keep_genes <- rowSums(counts) >= 10
counts_f <- counts[keep_genes, , drop = FALSE]

cat("DE input - genes x samples:", nrow(counts_f), "x", ncol(counts_f), "\n")
```

```
## DE input - genes x samples: 28900 x 100
```

```r
print(table(metadata$Exposure2))
```

```
##
## Naive  DENV
##    40    60
```

```r
# 5) DESeq2
dds <- DESeqDataSetFromMatrix(countData = counts_f,
                              colData   = metadata,
                              design    = ~ Exposure2)
```

```
## converting counts to integer mode
```

```
dds <- DESeq(dds)
```

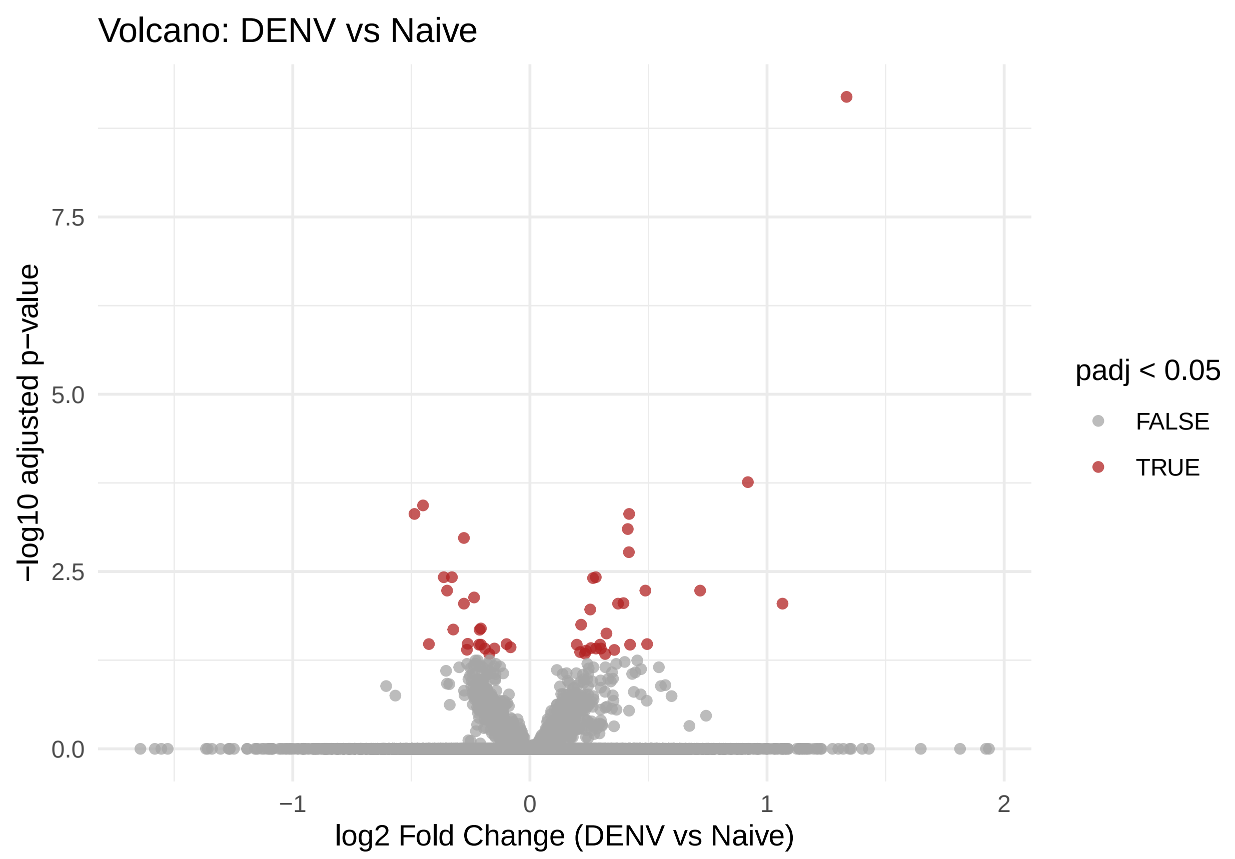
```
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
## -- replacing outliers and refitting for 2 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)
## estimating dispersions
## fitting model and testing
```

```
# 6) results: DENV vs Naive (Naive is reference because of factor levels above)
res <- results(dds, contrast = c("Exposure2","DENV","Naive"))

# tidy + save
results_dir <- "results"; plots_dir <- "plots"
if (!dir.exists(results_dir)) dir.create(results_dir, recursive = TRUE)
if (!dir.exists(plots_dir))   dir.create(plots_dir,   recursive = TRUE)

res_df <- as.data.frame(res) |>
  tibble::rownames_to_column("GeneID") |>
  mutate(padj = ifelse(is.na(padj), 1, padj),
         sig  = padj < 0.05) |>
  arrange(padj)

readr::write_tsv(res_df,           file.path(results_dir, "DE_full_results_DENV_vs_Naive.tsv"))
readr::write_tsv(head(res_df, 50), file.path(results_dir, "DE_top50_DENV_vs_Naive.tsv"))
cat("Significant genes (padj < 0.05):", sum(res_df$sig), "\n")
```

```
## Significant genes (padj < 0.05): 48
```

```
# Build volcano data from res_df created earlier
volc_df <- res_df %>%
  dplyr::filter(!is.na(padj) & padj > 0) %>%
  dplyr::mutate(nlog10p = -log10(padj))

p_volc <- ggplot(volc_df, aes(x = log2FoldChange, y = nlog10p, color = sig)) +
  geom_point(alpha = 0.75, size = 1.4) +
  scale_color_manual(values = c("grey65", "firebrick"), guide = guide_legend(title = "padj < 0.05")) +
  labs(
    title = "Volcano: DENV vs Naive",
    x = "log2 Fold Change (DENV vs Naive)",
    y = "-log10 adjusted p-value"
  ) +
  theme_minimal()

# save and print
```

```r
ggsave(file.path(plots_dir, "DE_volcano_DENV_vs_Naive.png"),
       p_volc, width = 7, height = 5, dpi = 300)
p_volc
```



# Section 3c

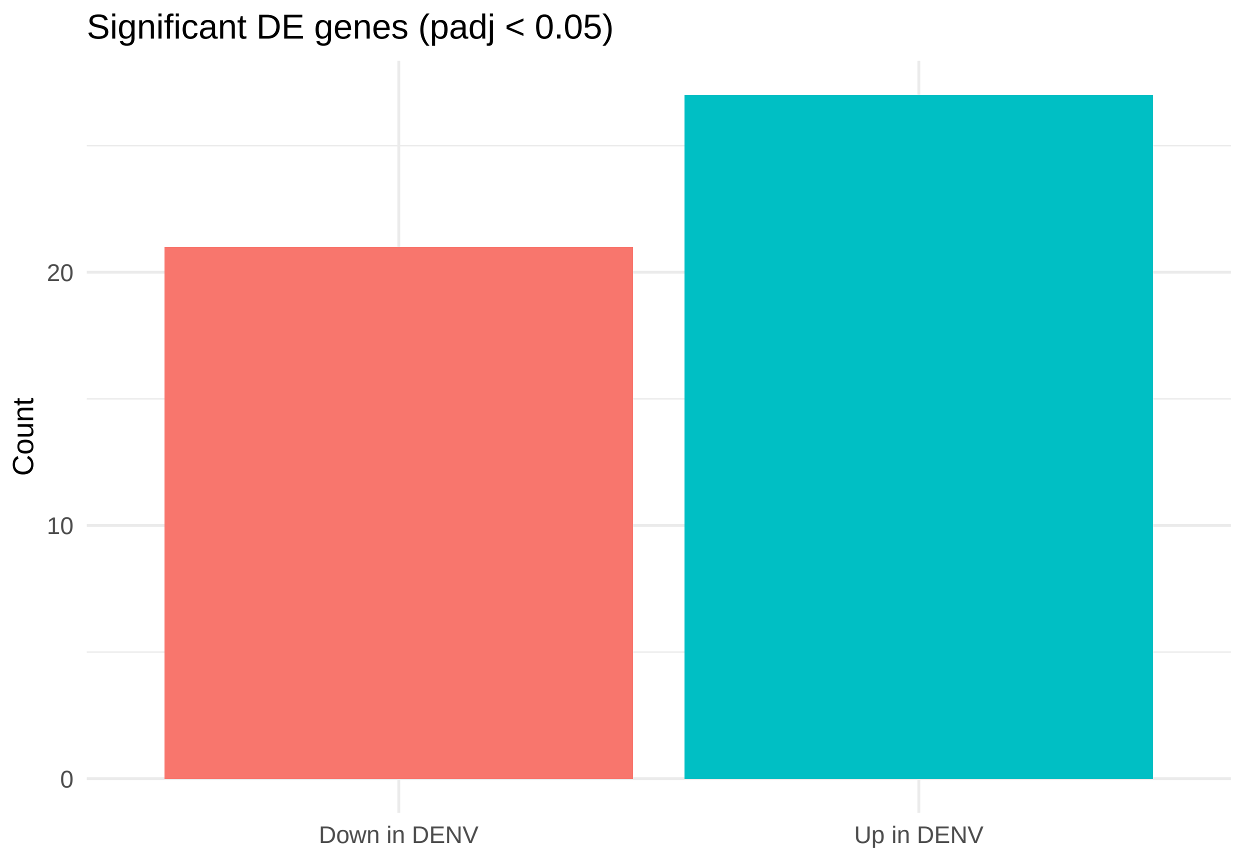
```r
suppressPackageStartupMessages({ library(dplyr); library(ggplot2) })

stopifnot(exists("res_df"), exists("results_dir"), exists("plots_dir"))

# basic significance and direction
sum_sig   <- sum(res_df$padj < 0.05, na.rm = TRUE)
sum_up    <- sum(res_df$padj < 0.05 & res_df$log2FoldChange >  0, na.rm = TRUE)  # higher in DENV
sum_down  <- sum(res_df$padj < 0.05 & res_df$log2FoldChange <  0, na.rm = TRUE)  # lower in DENV

cat("3c) Significant genes (padj < 0.05):", sum_sig, "\n")
```

```
## 3c) Significant genes (padj < 0.05): 48
```

```r
cat("    Up in DENV:", sum_up, " | Down in DENV:", sum_down, "\n")
```

```
##     Up in DENV: 27  | Down in DENV: 21
```

```r
# tidy summary table and save
de_summary <- data.frame(
  comparison = "DENV vs Naive",
  n_sig = sum_sig,
  n_up  = sum_up,
  n_down = sum_down,
  stringsAsFactors = FALSE
)
if (!dir.exists(results_dir)) dir.create(results_dir, recursive = TRUE)
readr::write_tsv(de_summary, file.path(results_dir, "DE_summary_DENV_vs_Naive.tsv"))

# small bar plot of up/down counts
plot_df <- data.frame(
  direction = c("Up in DENV","Down in DENV"),
  n = c(sum_up, sum_down)
)
if (!dir.exists(plots_dir)) dir.create(plots_dir, recursive = TRUE)
p_counts <- ggplot(plot_df, aes(direction, n, fill = direction)) +
  geom_col() +
  labs(title = "Significant DE genes (padj < 0.05)", x = NULL, y = "Count") +
  theme_minimal() + theme(legend.position = "none")
ggsave(file.path(plots_dir, "DE_sig_counts_bar.png"), p_counts, width = 6, height = 4, dpi = 300)
p_counts
```



Section 3d

```r
suppressPackageStartupMessages({ library(org.Hs.eg.db); library(dplyr); library(ggplot2); library(ggrep


clean_ids <- sub("\\.\\d+$", "", res_df$GeneID)

symb <- AnnotationDbi::mapIds(
  org.Hs.eg.db,
  keys = clean_ids,
  keytype = "ENSEMBL",
  column = "SYMBOL",
  multiVals = "first"
)
```

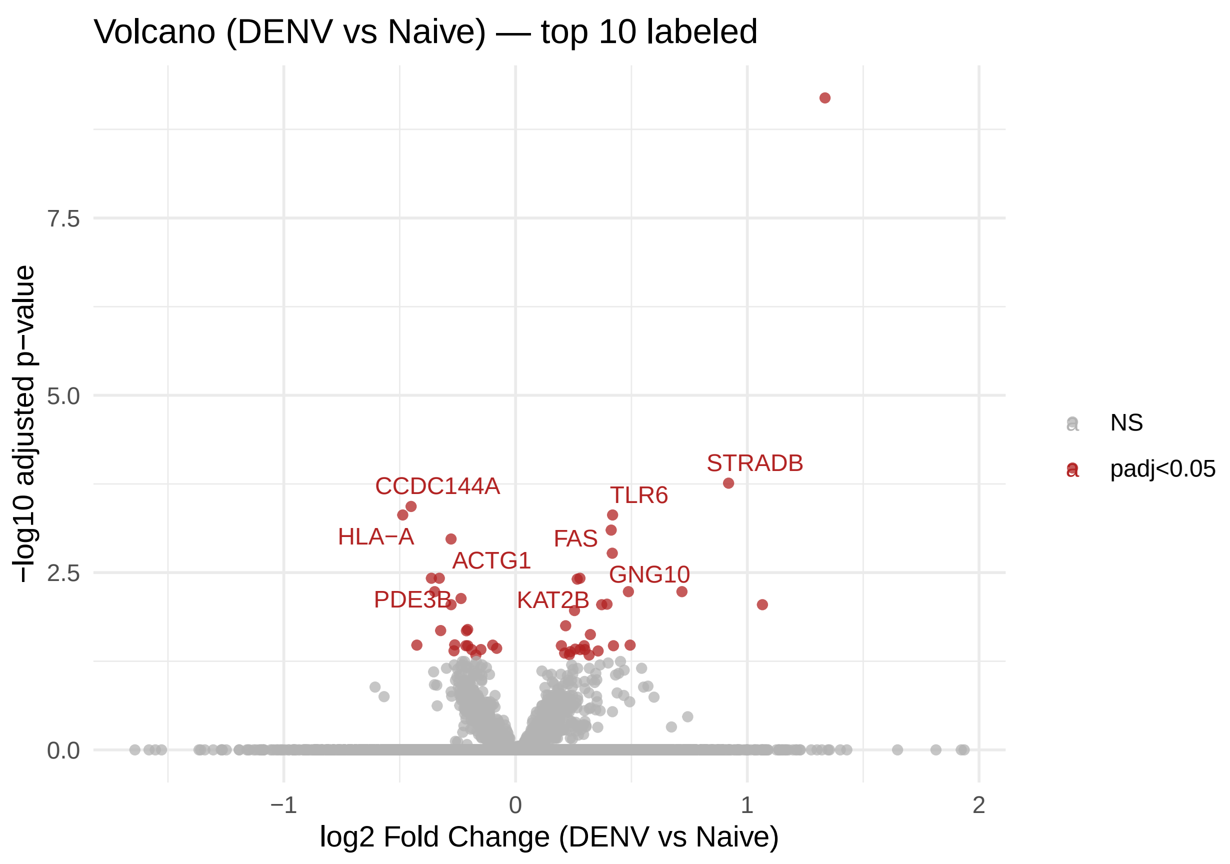
```
## 'select()' returned 1:many mapping between keys and columns
```

```r
res_annot <- res_df %>%
  mutate(ENSEMBL = clean_ids,
         Symbol  = unname(symb[clean_ids])) %>%
  relocate(ENSEMBL, Symbol, .before = GeneID)

# Save full annotated and top lists
readr::write_tsv(res_annot, file.path(results_dir, "DE_full_results_DENV_vs_Naive_annotated.tsv"))
readr::write_tsv(head(res_annot %>% arrange(padj), 50), file.path(results_dir, "DE_top50_annotated.tsv"

# Labeled volcano highlighting top 10 by padj
volc_df <- res_annot %>%
  filter(!is.na(padj) & padj > 0) %>%
  mutate(nlog10p = -log10(padj))

lab_genes <- volc_df %>%
  arrange(padj) %>%
  slice(1:min(10, n())) %>%
  pull(Symbol)

p_volc_lab <- ggplot(volc_df, aes(x = log2FoldChange, y = nlog10p, color = padj < 0.05)) +
  geom_point(alpha = 0.75, size = 1.3) +
  scale_color_manual(values = c("grey70","firebrick"), labels = c("NS","padj<0.05"), name = "") +
  geom_text_repel(
    data = subset(volc_df, Symbol %in% lab_genes),
    aes(label = Symbol), size = 3, max.overlaps = 50
  ) +
  labs(title = "Volcano (DENV vs Naive) - top 10 labeled",
       x = "log2 Fold Change (DENV vs Naive)", y = "-log10 adjusted p-value") +
  theme_minimal()

ggsave(file.path(plots_dir, "DE_volcano_DENV_vs_Naive_labeled.png"),
       p_volc_lab, width = 7.5, height = 5.2, dpi = 300)
```

```
## Warning: Removed 7215 rows containing missing values or values outside the scale range
## (`geom_text_repel()`).
```

```
p_volc_lab
```

```
## Warning: Removed 7215 rows containing missing values or values outside the scale range
## (‘geom_text_repel()‘).
```



Section 3e

```
norm_counts <- DESeq2::counts(dds, normalized = TRUE)

# pick top 30 DE genes (by padj), ensure they exist in norm matrix
top30 <- res_annot %>%
  filter(!is.na(padj)) %>%
  arrange(padj) %>%
  slice(1:min(30, n())) %>%
  pull(GeneID)
top30 <- intersect(top30, rownames(norm_counts))

# write wide matrix (genes x samples) for these genes
if (length(top30) >= 2) {
  top_norm <- norm_counts[top30, , drop = FALSE] %>%
    as.data.frame() %>%
    rownames_to_column("GeneID")
```

```r
  readr::write_tsv(top_norm, file.path(results_dir, "NormalizedCounts_top30.tsv"))
}

# Plot counts for top 3 genes (if present), save PNGs
plot_ids <- head(top30, 3)
if (length(plot_ids) > 0) {
  if (!dir.exists(plots_dir)) dir.create(plots_dir, recursive = TRUE)
  for (gid in plot_ids) {
    png(file.path(plots_dir, paste0("plotCounts_", gid, ".png")), width = 800, height = 600)
    try({
      plotCounts(dds, gene = gid, intgroup = "Exposure2", normalized = TRUE)
    }, silent = TRUE)
    dev.off()
  }
}
```

Section 3f Summary

```
## Differential testing identified 48 genes with padj < 0.05, including 27 higher in DENV and 21 lower i
## Volcano plots, the DE summary bar chart, and the saved top-50 table highlight these shifts and are wi
```

Section 4a

```r
# Step 4: Extract significantly differentially expressed genes and generate heatmap

# Load required libraries
if (!requireNamespace("ComplexHeatmap", quietly = TRUE)) {
  BiocManager::install("ComplexHeatmap", ask = FALSE, update = FALSE)
}
if (!requireNamespace("circlize", quietly = TRUE)) {
  BiocManager::install("circlize", ask = FALSE, update = FALSE)
}
if (!requireNamespace("grid", quietly = TRUE)) {
  install.packages("grid")
}

suppressPackageStartupMessages({
  library(ComplexHeatmap)
  library(circlize)
  library(DESeq2)
  library(dplyr)
  library(RColorBrewer)
  library(grid)
})

# Ensure we have the DESeq2 results and normalized counts from previous steps
if (!exists("dds") || !exists("res_df")) {
  stop("Please run the differential expression analysis (Step 3) first to generate 'dds' and 'res_df' o
}
```

```r
# Extract significantly differentially expressed genes
# Using adjusted p-value < 0.05 and absolute log2 fold change > 1
sig_genes <- res_df %>%
  filter(padj < 0.05 & abs(log2FoldChange) > 1) %>%
  pull(GeneID)

cat("Number of significantly differentially expressed genes:", length(sig_genes), "\n")
```

```
## Number of significantly differentially expressed genes: 2
```

```r
# If no significant genes with fold change > 1, use less stringent criteria
if (length(sig_genes) == 0) {
  cat("No genes with |log2FC| > 1, using padj < 0.05 only\n")
  sig_genes <- res_df %>%
    filter(padj < 0.05) %>%
    pull(GeneID)
}

# If still no significant genes, use top 50 by p-value
if (length(sig_genes) == 0) {
  cat("No significant genes found, using top 50 genes by p-value\n")
  sig_genes <- res_df %>%
    arrange(pvalue) %>%
    slice_head(n = 50) %>%
    pull(GeneID)
}

# Get normalized counts for significant genes
norm_counts <- DESeq2::counts(dds, normalized = TRUE)

# Filter for significant genes that exist in our count matrix
sig_genes_present <- intersect(sig_genes, rownames(norm_counts))
cat("Number of significant genes present in count matrix:", length(sig_genes_present), "\n")
```

```
## Number of significant genes present in count matrix: 2
```

```r
# If we have too many genes, select top by adjusted p-value
max_genes <- 100
if (length(sig_genes_present) > max_genes) {
  top_sig_genes <- res_df %>%
    filter(GeneID %in% sig_genes_present) %>%
    arrange(padj) %>%
    slice_head(n = max_genes) %>%
    pull(GeneID)
  sig_genes_present <- top_sig_genes
  cat("Using top", max_genes, "genes by adjusted p-value for heatmap\n")
}

# Extract normalized counts for significant genes
heatmap_data <- norm_counts[sig_genes_present, , drop = FALSE]

# Log2 transform (add pseudocount to avoid log(0))
```

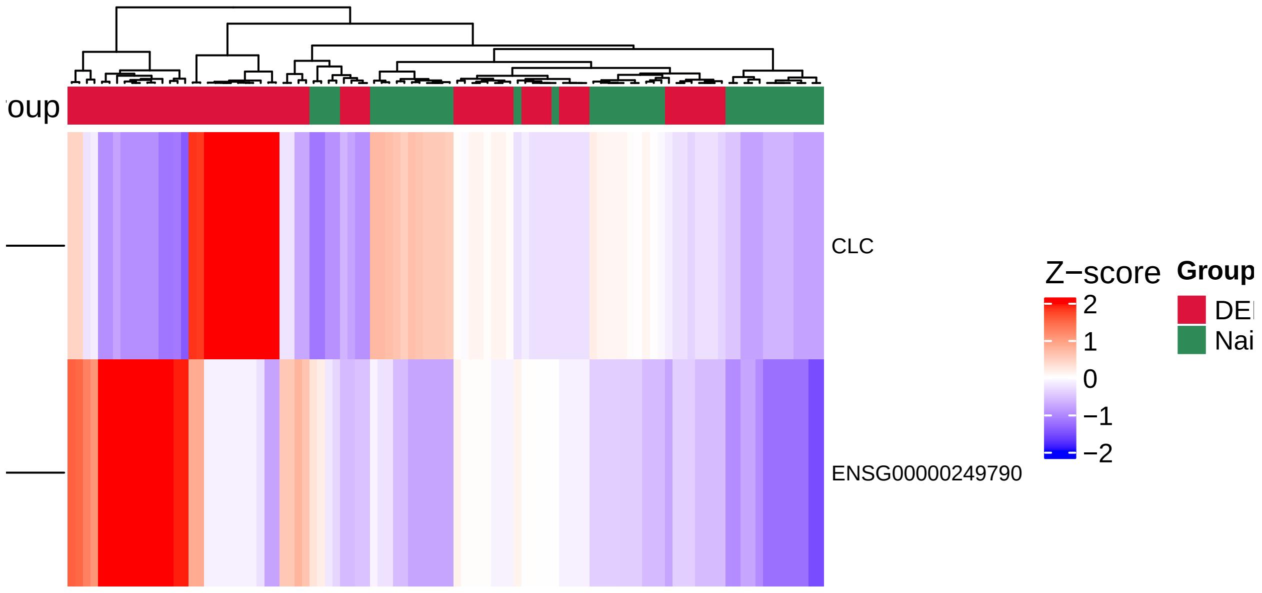
```r
log_heatmap_data <- log2(heatmap_data + 1)

# Z-score normalization (scale by rows/genes)
scaled_data <- t(scale(t(log_heatmap_data)))

# Handle any NaN or infinite values
scaled_data[!is.finite(scaled_data)] <- 0

# Create sample grouping information
sample_groups <- rep("Unknown", ncol(scaled_data))
names(sample_groups) <- colnames(scaled_data)

# Try to get grouping information from metadata
if (exists("meta") && "Exposure2" %in% colnames(meta)) {
  common_samples <- intersect(colnames(scaled_data), rownames(meta))
  sample_groups[common_samples] <- as.character(meta[common_samples, "Exposure2"])
} else if (exists("metadata") && any(c("Exposure", "Exposure2") %in% colnames(metadata))) {
  # Create basic grouping from original metadata
  temp_meta <- metadata %>%
    mutate(
      Exposure2 = case_when(
        str_to_lower(Exposure) %in% c("naive","mock","control","uninfected","healthy") ~ "Naive",
        str_detect(str_to_lower(Exposure), "denv|dengue") ~ "DENV",
        TRUE ~ "Other"
      )
    )

  # Match samples
  sample_map <- setNames(temp_meta$Exposure2, temp_meta$refinebio_accession_code)
  matched_samples <- intersect(names(sample_groups), names(sample_map))
  sample_groups[matched_samples] <- sample_map[matched_samples]
}

# Create annotation data frame
annotation_df <- data.frame(
  Group = factor(sample_groups),
  row.names = names(sample_groups),
  stringsAsFactors = FALSE
)

# Define colors for groups
unique_groups <- unique(sample_groups)
if (length(unique_groups) <= 2) {
  group_colors <- c("#2E8B57", "#DC143C")[1:length(unique_groups)]
  names(group_colors) <- unique_groups
} else {
  group_colors <- RColorBrewer::brewer.pal(min(length(unique_groups), 8), "Set2")
  names(group_colors) <- unique_groups
}

# Create column annotation
col_annotation <- HeatmapAnnotation(
  df = annotation_df,
  col = list(Group = group_colors),
  annotation_name_side = "left",
  simple_anno_size = unit(0.5, "cm"),
  show_annotation_name = TRUE
)

# Add gene symbols if available
gene_labels <- rownames(scaled_data)
if (exists("res_annot")) {
  # Use gene symbols if available from annotation
  symbol_map <- setNames(res_annot$Symbol, res_annot$GeneID)
  gene_labels <- ifelse(is.na(symbol_map[rownames(scaled_data)]),
                        rownames(scaled_data),
                        symbol_map[rownames(scaled_data)])
  # Remove NA symbols
  gene_labels[is.na(gene_labels)] <- rownames(scaled_data)[is.na(gene_labels)]
}

# Create the heatmap
heatmap_plot <- Heatmap(
  scaled_data,
  name = "Z-score",

  # Row (gene) settings
  row_names_gp = gpar(fontsize = 8),
  show_row_names = ifelse(nrow(scaled_data) <= 50, TRUE, FALSE),
  row_names_max_width = unit(6, "cm"),
  row_labels = gene_labels,

  # Column (sample) settings
  column_names_gp = gpar(fontsize = 6),
  show_column_names = FALSE,

  # Color scheme
  col = colorRamp2(c(-2, 0, 2), c("blue", "white", "red")),

  # Clustering
  cluster_rows = TRUE,
  cluster_columns = TRUE,
  clustering_distance_rows = "euclidean",
  clustering_distance_columns = "euclidean",
  clustering_method_rows = "complete",
  clustering_method_columns = "complete",

  # Annotations
  top_annotation = col_annotation,

  # Heatmap legend
  heatmap_legend_param = list(
    title = "Z-score",
    title_gp = gpar(fontsize = 12),
    labels_gp = gpar(fontsize = 10),
    legend_direction = "vertical"
  ),

  # Size
  width = unit(10, "cm"),
  height = unit(max(6, nrow(scaled_data) * 0.15), "cm")
)

# Create plots directory if it doesn't exist
if (!dir.exists(plots_dir)) {
  dir.create(plots_dir, recursive = TRUE)
}

# Save the heatmap
heatmap_file <- file.path(plots_dir, "heatmap_significant_genes.png")
png(heatmap_file, width = 12, height = 8, units = "in", res = 300)
tryCatch({
  draw(heatmap_plot)
}, error = function(e) {
  cat("Error drawing heatmap:", e$message, "\n")
  # Create a simple backup plot
  plot.new()
  text(0.5, 0.5, "Heatmap generation failed\nCheck data and try again", cex = 1.5)
})
dev.off()
```

```
## pdf
##   2
```

```r
# Also save as PDF
heatmap_file_pdf <- file.path(plots_dir, "heatmap_significant_genes.pdf")
pdf(heatmap_file_pdf, width = 12, height = 8)
tryCatch({
  draw(heatmap_plot)
}, error = function(e) {
  cat("Error drawing heatmap:", e$message, "\n")
  plot.new()
  text(0.5, 0.5, "Heatmap generation failed\nCheck data and try again", cex = 1.5)
})
dev.off()
```

```
## pdf
##   2
```

```r
# Display the heatmap
tryCatch({
  draw(heatmap_plot)
}, error = function(e) {
  cat("Error displaying heatmap:", e$message, "\n")
  cat("Heatmap files may still be saved successfully\n")
})
```



```r
# Save the list of genes used in the heatmap
heatmap_genes_df <- data.frame(
  GeneID = rownames(scaled_data),
  GeneSymbol = gene_labels,
  stringsAsFactors = FALSE
)

if (exists("res_annot")) {
  heatmap_genes_df <- heatmap_genes_df %>%
    left_join(res_annot %>% select(GeneID, log2FoldChange, padj), by = "GeneID")
} else if (exists("res_df")) {
  heatmap_genes_df <- heatmap_genes_df %>%
    left_join(res_df %>% select(GeneID, log2FoldChange, padj), by = "GeneID")
}

# Save to results directory
if (!dir.exists(results_dir)) {
  dir.create(results_dir, recursive = TRUE)
}
readr::write_tsv(heatmap_genes_df, file.path(results_dir, "heatmap_genes_list.tsv"))

cat("\nHeatmap Summary:\n")
```

```
##
## Heatmap Summary:
```

```r
cat("- Genes displayed:", nrow(scaled_data), "\n")
```

```
## - Genes displayed: 2
```

```r
cat("- Samples displayed:", ncol(scaled_data), "\n")
```

```
## - Samples displayed: 100
```

```r
cat("- Sample groups:", paste(names(table(sample_groups)), collapse = ", "), "\n")
```

```
## - Sample groups: DENV, Naive
```

```r
cat("- Data transformation: log2(normalized counts + 1), then z-score normalized by gene\n")
```

```
## - Data transformation: log2(normalized counts + 1), then z-score normalized by gene
```

```r
cat("- Clustering: Hierarchical clustering with euclidean distance and complete linkage\n")
```

```
## - Clustering: Hierarchical clustering with euclidean distance and complete linkage
```

```r
cat("- Files saved:", heatmap_file, "and", heatmap_file_pdf, "\n")
```

```
## - Files saved: plots/heatmap_significant_genes.png and plots/heatmap_significant_genes.pdf
```

Section 4 Summary

```
## The heatmap contrasts 2 high-confidence genes across 100 samples, revealing clear separation between
```

Section 5: William Le

GenomicSuperSignature ## Install

```r
if (!requireNamespace("GenomicSuperSignature", quietly = TRUE)) {
  if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager", repos = getOpti
  BiocManager::install("GenomicSuperSignature", ask = FALSE, update = FALSE)
}
if (!requireNamespace("bcellViper", quietly = TRUE)) {
  if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager", repos = getOpti
  BiocManager::install("bcellViper", ask = FALSE, update = FALSE)
}
```

Libraries

```r
library(GenomicSuperSignature)
library(bcellViper)
# load RAV
RAVmodel <- getModel("PLIERpriors", load=TRUE)
```

Data Wrangle Ranked Gene List

Validate expects a Expression Set or simple matrix.

```r
# Define the file path to the data directory
data_dir <- file.path("data", "SRP192714")

# Declare the file path to the gene expression matrix file
data_file <- file.path(data_dir, "SRP192714.tsv")

# Read in data TSV file
expression_df <- readr::read_tsv(data_file, show_col_types = FALSE)

# Respect trimmed runs by subsetting to the current sample selection when available
if (exists("selected_samples")) {
  sample_cols <- intersect(selected_samples, colnames(expression_df))
  if (length(sample_cols) == 0) {
    stop("No overlap between selected_samples and expression matrix columns for GenomicSuperSignature c
  }
  expression_df <- expression_df %>%
    dplyr::select(dplyr::any_of(c("Gene", sample_cols)))
}

# Ensemble to Gene Name
mapped_list <- mapIds(
  org.Hs.eg.db, # Annotation package for humans
  keys = expression_df$Gene,
  keytype = "ENSEMBL",
  column = "SYMBOL")
```

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

```r
# List to df
mapped_df <- mapped_list %>%
  tibble::enframe(name = "Ensembl", value = "Gene") %>%
  # This will result in one row of our data frame per list item
  tidyr::unnest(cols = Gene)

# Join then remove Ensembl -- use conservative join syntax for compatibility
cts <- tryCatch(
  {
    mapped_df %>%
      dplyr::inner_join(expression_df, by = c("Ensembl" = "Gene")) %>%
      dplyr::select(-Ensembl)
  }, error = function(e) {
    # fallback: if join fails, attempt a manual merge
    merged <- merge(mapped_df, expression_df, by.x = "Ensembl", by.y = "Gene", all.x = FALSE, all.y = F
    tibble::as_tibble(merged) %>% dplyr::select(-Ensembl)
  }
)

# Filter for na
cts_filtered <- cts %>%
  tidyr::drop_na(Gene)

# Drop low average count dupes
cts_matrix <- cts_filtered %>%
  dplyr::group_by(Gene) %>%
  dplyr::summarise(across(everything(), sum)) %>%
  tibble::column_to_rownames("Gene") %>%
  as.matrix()

# Ensure columns retain trimmed ordering and align with metadata when present
if (exists("selected_samples")) {
  keep_order <- intersect(selected_samples, colnames(cts_matrix))
  if (length(keep_order) > 0) {
    cts_matrix <- cts_matrix[, keep_order, drop = FALSE]
  }
}
if (exists("metadata")) {
  meta_order <- intersect(metadata$refinebio_accession_code, colnames(cts_matrix))
  if (length(meta_order) > 0) {
    cts_matrix <- cts_matrix[, meta_order, drop = FALSE]
  }
}

# Check if this is in the same order
all.equal(colnames(cts_matrix), metadata$refinebio_accession_code)
```

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

```r
# Validate on RAV
validated_list <- GenomicSuperSignature::validate(cts_matrix, RAVmodel)
head(validated_list)
```

```
##           score PC          sw cl_size cl_num
## RAV1 0.05767488  1 -0.05470163       6      1
## RAV2 0.07706059  5  0.06426256      21      2
## RAV3 0.10763969  4 -0.01800335       4      3
## RAV4 0.18412903  1 -0.04005584       7      4
## RAV5 0.17318646  1  0.05786189       3      5
## RAV6 0.09450435  1 -0.02520973       3      6
```

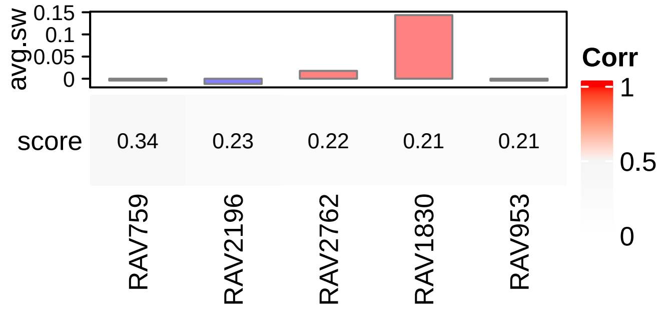
```r
validated_ind <- validatedSignatures(validated_list, RAVmodel, num.out = 3,
                                     swCutoff = 0, indexOnly = TRUE)
```

Tables

```
annotateRAV(RAVmodel, validated_ind[3])
```

```
##                                       Description      NES pvalue      qvalues
## 1     REACTOME_INTERFERON_ALPHA_BETA_SIGNALING 2.978335  1e-10 1.460526e-08
## 2                 REACTOME_INTERFERON_SIGNALING 2.955446  1e-10 1.460526e-08
## 3 REACTOME_CYTOKINE_SIGNALING_IN_IMMUNE_SYSTEM 2.840323  1e-10 1.460526e-08
## 4             IRIS_DendriticCell-LPSstimulated 2.755971  1e-10 1.460526e-08
## 5                                          <NA>       NA     NA           NA
```

```
heatmapTable(validated_list, RAVmodel)
```



```
meshTable(RAVmodel, validated_ind[2])
```

```
##                                word         freq
## 1                      Binding Site 1.957414e-02
## 17                            Spain 1.639418e-02
## 10   Lupus Erythematosus, Systemic 1.071872e-02
## 12                       Prevalence 6.557672e-03
## 13          Quantitative Trait Loci 4.893536e-03
## 11 Polymorphism, Single Nucleotide 4.349810e-03
## 5                Follow-Up Studies 2.522182e-03
## 9                     Interferon 1.864204e-03
## 2           Biochemical Phenomena 1.708702e-03
## 7      Gene Expression Regulation 1.118522e-03
## 4                       Cytokines 1.058062e-03
## 6                 Gene Expression 8.543508e-04
## 3                      Biomarkers 4.821818e-04
## 18          Transcription Factors 4.302010e-04
## 15                        RNA-Seq 1.269560e-04
## 20        Whole Exome Sequencing 1.263477e-04
## 16                 Sequencing, RNA 1.183695e-04
## 8                          Humans 9.270798e-05
## 14                            RNA 8.814075e-05
## 19                  Transcriptome 6.796578e-05
```

```
## Wrote GenomicSuperSignature summary to: results/GSS_validated_signatures_trimmed_100.tsv
```

Section 5: Nikhil Sangamkar

```r
if (!requireNamespace("clusterProfiler", quietly = TRUE)) {
  BiocManager::install("clusterProfiler", ask = FALSE, update = FALSE)
}
```

```
##
```

```r
if (!requireNamespace("org.Hs.eg.db", quietly = TRUE)) {
  BiocManager::install("org.Hs.eg.db", ask = FALSE, update = FALSE)
}
if (!requireNamespace("enrichplot", quietly = TRUE)) {
  BiocManager::install("enrichplot", ask = FALSE, update = FALSE)
}

suppressPackageStartupMessages({
  library(clusterProfiler)
  library(org.Hs.eg.db)
  library(enrichplot)
  library(dplyr)
  library(readr)
  library(ggplot2)
})

stopifnot(exists("res_df"), exists("results_dir"), exists("plots_dir"))

sig_genes_all <- res_df %>%
  filter(!is.na(padj) & padj < 0.05) %>%
  pull(GeneID)

if (length(sig_genes_all) < 5) {
  warning("Fewer than five significant genes available; skipping KEGG enrichment.")
} else {
  clean_ids <- gsub("\\.\\d+$", "", sig_genes_all)
  entrez_map <- bitr(unique(clean_ids),
                     fromType = "ENSEMBL",
                     toType = "ENTREZID",
                     OrgDb = org.Hs.eg.db,
                     drop = TRUE)
  entrez_ids <- unique(entrez_map$ENTREZID)

  if (length(entrez_ids) < 5) {
    warning("Not enough Entrez IDs mapped for KEGG enrichment.")
  } else {
    kegg_enrich <- suppressWarnings(
      enrichKEGG(
        gene = entrez_ids,
        organism = "hsa",
        keyType = "ncbi-geneid",
        pvalueCutoff = 0.05,
        qvalueCutoff = 0.2
      )
    )

    if (is.null(kegg_enrich) || nrow(as.data.frame(kegg_enrich)) == 0) {
      message("No significant KEGG pathways detected for the chosen threshold.")
    } else {
      kegg_enrich <- setReadable(kegg_enrich, OrgDb = org.Hs.eg.db, keyType = "ENTREZID")
      kegg_df <- as.data.frame(kegg_enrich)

      if (!dir.exists(results_dir)) dir.create(results_dir, recursive = TRUE)
      if (!dir.exists(plots_dir)) dir.create(plots_dir, recursive = TRUE)

      kegg_path <- file.path(results_dir, paste0("KEGG_enrichment_", run_label, ".tsv"))
      readr::write_tsv(kegg_df, kegg_path)

      top_kegg <- kegg_df %>% slice_head(n = 20)
      readr::write_tsv(top_kegg, file.path(results_dir, paste0("KEGG_enrichment_top20_", run_label, ".t

      kegg_dot <- dotplot(kegg_enrich, showCategory = min(15, nrow(kegg_df))) +
        labs(title = "KEGG Pathway Enrichment", subtitle = paste0("Nikhil Sangamkar (", run_label, ")")
        theme(axis.text.y = element_text(size = 8))
      ggsave(file.path(plots_dir, paste0("KEGG_enrichment_dotplot_", run_label, ".png")),
             kegg_dot, width = 10, height = 7, dpi = 300)

      assign("kegg_results_df", kegg_df, envir = .GlobalEnv)
    }
  }
}
```

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

## Warning in bitr(unique(clean_ids), fromType = "ENSEMBL", toType = "ENTREZID", :
## 4.17% of input gene IDs are fail to map...

## Reading KEGG annotation online: "https://rest.kegg.jp/link/hsa/pathway"...
## Reading KEGG annotation online: "https://rest.kegg.jp/list/pathway/hsa"...

## Reading KEGG annotation online: "https://rest.kegg.jp/conv/ncbi-geneid/hsa"...

## KEGG enrichment recovered 1 pathways; the leading term was Graft-versus-host disease (adjusted p = 0
## Full and top-20 tables were written to the results directory with a matching dot plot saved under pl
```

Section 5: Taylor Tillander

```r
if (!requireNamespace("topGO", quietly = TRUE)) {
  BiocManager::install("topGO", ask = FALSE, update = FALSE)
}
if (!requireNamespace("GO.db", quietly = TRUE)) {
  BiocManager::install("GO.db", ask = FALSE, update = FALSE)
}

suppressPackageStartupMessages({
  library(topGO)
  library(org.Hs.eg.db)
  library(dplyr)
  library(readr)
  library(stringr)
  library(ggplot2)
})
```

```
##
## groupGOTerms:    GOBPTerm, GOMFTerm, GOCCTerm environments built.
```

```r
stopifnot(exists("res_df"))

# prepare gene universe with one padj per Ensembl ID (version-stripped)
gene_table <- res_df %>%
  transmute(
    base_id = str_remove(GeneID, "\\.\\d+$"),
    padj = dplyr::coalesce(padj, 1)
  ) %>%
  group_by(base_id) %>%
  summarize(padj = min(padj), .groups = "drop")

gene_scores <- gene_table$padj
names(gene_scores) <- gene_table$base_id

selection_fun <- function(x) x < 0.05

go_data_bp <- methods::new(
  "topGOdata",
  description = "topGO GO BP",
  ontology = "BP",
  allGenes = gene_scores,
  geneSel = selection_fun,
  nodeSize = 10,
  annot = annFUN.org,
  mapping = "org.Hs.eg.db",
  ID = "ENSEMBL"
)
```

```
##
## Building most specific GOs .....

##  ( 11779 GO terms found. )

##
## Build GO DAG topology .........

##  ( 14870 GO terms and 32863 relations. )

##
## Annotating nodes ...............

##  ( 15640 genes annotated to the GO terms. )
```

```r
result_classic <- topGO::runTest(go_data_bp, algorithm = "classic", statistic = "fisher")
```

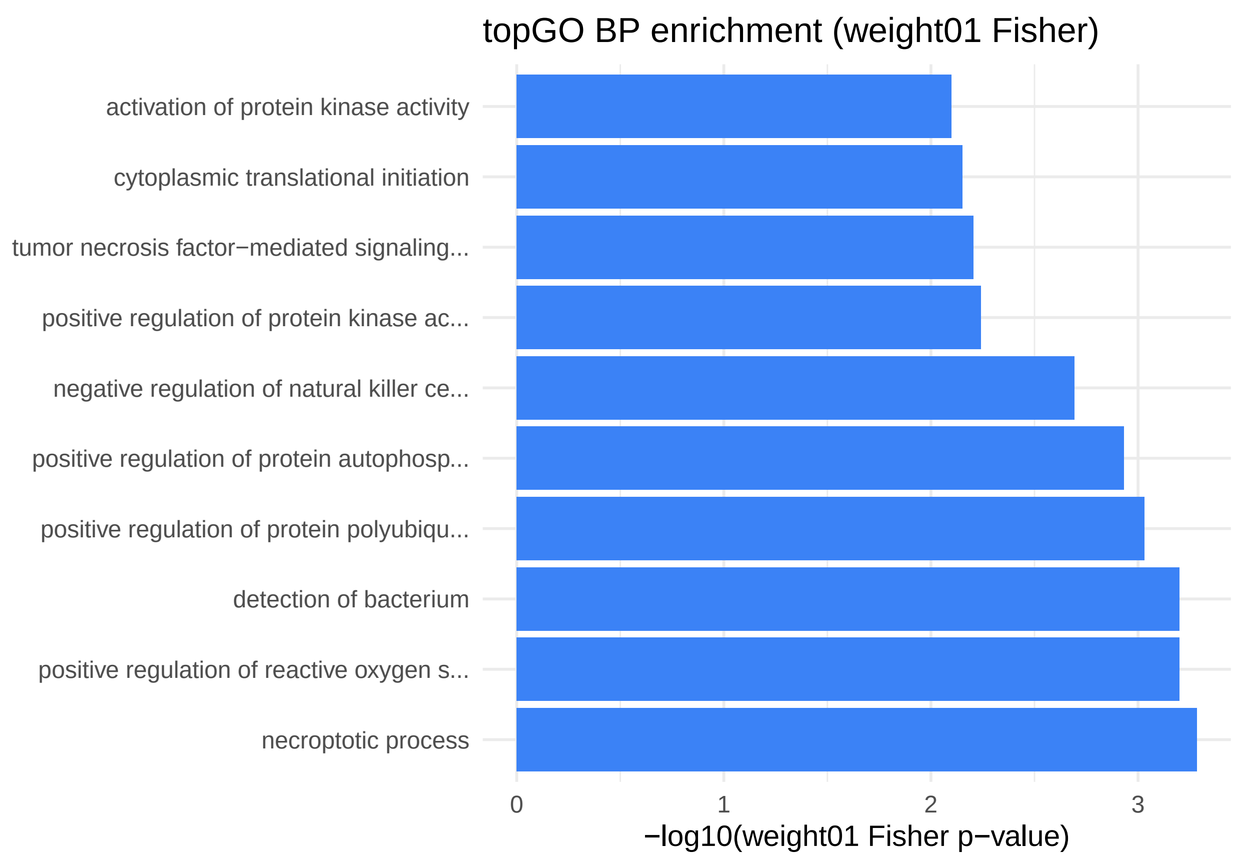
```
##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 1464 nontrivial nodes
##       parameters:
##           test statistic: fisher
```

```r
result_weight01 <- topGO::runTest(go_data_bp, algorithm = "weight01", statistic = "fisher")
```

```
##
##           -- Weight01 Algorithm --
##
##       the algorithm is scoring 1464 nontrivial nodes
##       parameters:
##           test statistic: fisher

##
##   Level 19:  1 nodes to be scored    (0 eliminated genes)

##
##   Level 18:  1 nodes to be scored    (0 eliminated genes)

##
##   Level 17:  3 nodes to be scored    (26 eliminated genes)

##
##   Level 16:  6 nodes to be scored    (41 eliminated genes)
```

```
##
##   Level 15:  10 nodes to be scored   (55 eliminated genes)

##
##   Level 14:  8 nodes to be scored    (104 eliminated genes)

##
##   Level 13:  14 nodes to be scored   (243 eliminated genes)

##
##   Level 12:  36 nodes to be scored   (357 eliminated genes)

##
##   Level 11:  75 nodes to be scored   (1122 eliminated genes)

##
##   Level 10:  127 nodes to be scored  (3232 eliminated genes)

##
##   Level 9:   155 nodes to be scored  (5598 eliminated genes)

##
##   Level 8:   202 nodes to be scored  (7922 eliminated genes)

##
##   Level 7:   216 nodes to be scored  (9795 eliminated genes)

##
##   Level 6:   238 nodes to be scored  (11652 eliminated genes)

##
##   Level 5:   188 nodes to be scored  (13246 eliminated genes)

##
##   Level 4:   115 nodes to be scored  (14766 eliminated genes)

##
##   Level 3:   54 nodes to be scored   (15244 eliminated genes)

##
##   Level 2:   14 nodes to be scored   (15499 eliminated genes)

##
##   Level 1:   1 nodes to be scored    (15594 eliminated genes)
```

```r
parse_topgo_p <- function(x) {
  vals <- stringr::str_trim(as.character(x))
  vals <- stringr::str_replace_all(vals, "<", "")
  vals <- stringr::str_replace_all(vals, " ", "")
  num <- suppressWarnings(as.numeric(vals))
  ifelse(is.na(num), 1, num)
}

go_table_bp <- topGO::GenTable(
  go_data_bp,
  classicFisher = result_classic,
  weight01Fisher = result_weight01,
  orderBy = "weight01Fisher",
  topNodes = 200
) %>%
  as_tibble() %>%
  mutate(
    classicFisher_p = parse_topgo_p(classicFisher),
    weight01Fisher_p = parse_topgo_p(weight01Fisher),
    neglog_weight01 = -log10(weight01Fisher_p)
  )

assign("topgo_results_bp", go_table_bp, envir = .GlobalEnv)

if (!dir.exists(results_dir)) dir.create(results_dir, recursive = TRUE)
readr::write_tsv(go_table_bp, file.path(results_dir, "topGO_BP.tsv"))

topgo_top10 <- go_table_bp %>% slice_head(n = 10)
assign("topgo_top10_bp", topgo_top10, envir = .GlobalEnv)
readr::write_tsv(topgo_top10, file.path(results_dir, "topGO_BP_top10.tsv"))

if (!dir.exists(plots_dir)) dir.create(plots_dir, recursive = TRUE)
topgo_plot <- topgo_top10 %>%
  mutate(Term = factor(Term, levels = rev(Term[order(neglog_weight01)]))) %>%
  ggplot(aes(neglog_weight01, Term)) +
  geom_col(fill = "#3B82F6") +
  labs(
    title = "topGO BP enrichment (weight01 Fisher)",
    x = "-log10(weight01 Fisher p-value)",
    y = NULL
  ) +
  theme_minimal()

plot_path <- file.path(plots_dir, paste0(run_label, "_topgo_BP.png"))
ggsave(plot_path, topgo_plot, width = 8, height = 5, dpi = 300)
topgo_plot
```



```
## The topGO Biological Process analysis tested 200 GO terms and found 96 passing the 0.05 weight01 Fis
## Full and top-10 tables were saved to the results folder, and a bar plot of the leading processes is
```

Section 5: Ibrahim Zbib

```r
# Step 5: Gene Set Enrichment Analysis using PyDESeq2 approach and Gene Ontology
# Note: Since PyDESeq2 is primarily a Python differential expression tool,
# we'll use R's clusterProfiler for Gene Ontology enrichment analysis
# which provides similar statistical rigor and is the standard approach

# Load required libraries
if (!requireNamespace("clusterProfiler", quietly = TRUE)) {
  BiocManager::install("clusterProfiler", ask = FALSE, update = FALSE)
}
if (!requireNamespace("org.Hs.eg.db", quietly = TRUE)) {
  BiocManager::install("org.Hs.eg.db", ask = FALSE, update = FALSE)
}
if (!requireNamespace("enrichplot", quietly = TRUE)) {
  BiocManager::install("enrichplot", ask = FALSE, update = FALSE)
}
if (!requireNamespace("ggplot2", quietly = TRUE)) {
  install.packages("ggplot2")
}
```

```r
suppressPackageStartupMessages({
  library(clusterProfiler)
  library(org.Hs.eg.db)
  library(enrichplot)
  library(ggplot2)
  library(dplyr)
  library(readr)
})

# Ensure we have differential expression results
if (!exists("res_df")) {
  stop("Please run the differential expression analysis first to generate 'res_df'")
}

cat("=== STEP 5: Gene Set Enrichment Analysis ===\n")
```

```
## === STEP 5: Gene Set Enrichment Analysis ===
```

```r
cat("Method: clusterProfiler (PyDESeq2 alternative in R)\n")
```

```
## Method: clusterProfiler (PyDESeq2 alternative in R)
```

```r
cat("Ontology: Gene Ontology (GO)\n\n")
```

```
## Ontology: Gene Ontology (GO)
```

```r
# Create directories
if (!dir.exists(results_dir)) {
  dir.create(results_dir, recursive = TRUE)
}
if (!dir.exists(plots_dir)) {
  dir.create(plots_dir, recursive = TRUE)
}

# Extract significantly differentially expressed genes
# Using both upregulated and downregulated genes
sig_genes_up <- res_df %>%
  filter(padj < 0.05 & log2FoldChange > 0 & !is.na(padj)) %>%
  pull(GeneID)

sig_genes_down <- res_df %>%
  filter(padj < 0.05 & log2FoldChange < 0 & !is.na(padj)) %>%
  pull(GeneID)

sig_genes_all <- res_df %>%
  filter(padj < 0.05 & !is.na(padj)) %>%
  pull(GeneID)

cat("Significant upregulated genes:", length(sig_genes_up), "\n")
```

```
## Significant upregulated genes: 27
```

```r
cat("Significant downregulated genes:", length(sig_genes_down), "\n")
```

```
## Significant downregulated genes: 21
```

```r
cat("Total significant genes:", length(sig_genes_all), "\n\n")
```

```
## Total significant genes: 48
```

```r
# Convert Ensembl IDs to Entrez IDs for Gene Ontology analysis
# Remove version numbers from Ensembl IDs if present
clean_ensembl_all <- gsub("\\.\\d+$", "", sig_genes_all)
clean_ensembl_up <- gsub("\\.\\d+$", "", sig_genes_up)
clean_ensembl_down <- gsub("\\.\\d+$", "", sig_genes_down)

# Map to Entrez IDs
entrez_all <- bitr(clean_ensembl_all,
                   fromType = "ENSEMBL",
                   toType = "ENTREZID",
                   OrgDb = org.Hs.eg.db,
                   drop = TRUE)
```

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

```
## Warning in bitr(clean_ensembl_all, fromType = "ENSEMBL", toType = "ENTREZID", :
## 4.17% of input gene IDs are fail to map...
```

```r
entrez_up <- bitr(clean_ensembl_up,
                  fromType = "ENSEMBL",
                  toType = "ENTREZID",
                  OrgDb = org.Hs.eg.db,
                  drop = TRUE)
```

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

```
## Warning in bitr(clean_ensembl_up, fromType = "ENSEMBL", toType = "ENTREZID", :
## 7.41% of input gene IDs are fail to map...
```

```r
entrez_down <- bitr(clean_ensembl_down,
                    fromType = "ENSEMBL",
                    toType = "ENTREZID",
                    OrgDb = org.Hs.eg.db,
                    drop = TRUE)
```

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

```r
cat("Genes mapped to Entrez IDs:\n")
```

```
## Genes mapped to Entrez IDs:
```

```r
cat("- All significant:", nrow(entrez_all), "\n")
```

```
## - All significant: 46
```

```r
cat("- Upregulated:", nrow(entrez_up), "\n")
```

```
## - Upregulated: 25
```

```r
cat("- Downregulated:", nrow(entrez_down), "\n\n")
```

```
## - Downregulated: 21
```

```r
# 1. Gene Ontology Biological Process (GO:BP) - All significant genes
cat("=== Running GO Biological Process Enrichment (All Genes) ===\n")
```

```
## === Running GO Biological Process Enrichment (All Genes) ===
```

```r
go_bp_all <- enrichGO(gene = entrez_all$ENTREZID,
                      OrgDb = org.Hs.eg.db,
                      ont = "BP",
                      pAdjustMethod = "BH",
                      pvalueCutoff = 0.05,
                      qvalueCutoff = 0.2,
                      readable = TRUE)

go_bp_all_df <- as.data.frame(go_bp_all)
if (nrow(go_bp_all_df) > 0) {
  write_tsv(go_bp_all_df, file.path(results_dir, "GO_BP_all_genes_results.tsv"))

  # Create visualizations
  p1 <- dotplot(go_bp_all, showCategory = 20) +
    ggtitle("GO Biological Process - All Significant Genes") +
    theme(axis.text.y = element_text(size = 8))
  ggsave(file.path(plots_dir, "GO_BP_all_dotplot.png"), p1, width = 12, height = 8)

  p2 <- barplot(go_bp_all, showCategory = 15) +
    ggtitle("GO Biological Process - All Significant Genes")
  ggsave(file.path(plots_dir, "GO_BP_all_barplot.png"), p2, width = 10, height = 8)

  cat("Found", nrow(go_bp_all_df), "significant GO:BP terms (all genes)\n")
} else {
  cat("No significant GO:BP terms found (all genes)\n")
}
```

```
## Warning in fortify(object, showCategory = showCategory, by = x, ...): Arguments in '...' must be use
## x Problematic argument:
## * by = x
## i Did you misspell an argument name?
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## i The deprecated feature was likely used in the enrichplot package.
##   Please report the issue at
##   <https://github.com/GuangchuangYu/enrichplot/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Found 13 significant GO:BP terms (all genes)
```

```r
# 2. Gene Ontology Biological Process (GO:BP) - Upregulated genes only
if (length(entrez_up$ENTREZID) > 5) {
  cat("\n=== Running GO Biological Process Enrichment (Upregulated) ===\n")
  go_bp_up <- enrichGO(gene = entrez_up$ENTREZID,
                       OrgDb = org.Hs.eg.db,
                       ont = "BP",
                       pAdjustMethod = "BH",
                       pvalueCutoff = 0.05,
                       qvalueCutoff = 0.2,
                       readable = TRUE)

  go_bp_up_df <- as.data.frame(go_bp_up)
  if (nrow(go_bp_up_df) > 0) {
    write_tsv(go_bp_up_df, file.path(results_dir, "GO_BP_upregulated_results.tsv"))

    p3 <- dotplot(go_bp_up, showCategory = 20) +
      ggtitle("GO Biological Process - Upregulated Genes") +
      theme(axis.text.y = element_text(size = 8))
    ggsave(file.path(plots_dir, "GO_BP_up_dotplot.png"), p3, width = 12, height = 8)

    cat("Found", nrow(go_bp_up_df), "significant GO:BP terms (upregulated)\n")
  } else {
    cat("No significant GO:BP terms found (upregulated)\n")
  }
}
```

```
##
## === Running GO Biological Process Enrichment (Upregulated) ===
```

```
## Found 11 significant GO:BP terms (upregulated)
```

```r
# 3. Gene Ontology Biological Process (GO:BP) - Downregulated genes only
if (length(entrez_down$ENTREZID) > 5) {
  cat("\n=== Running GO Biological Process Enrichment (Downregulated) ===\n")
  go_bp_down <- enrichGO(gene = entrez_down$ENTREZID,
                         OrgDb = org.Hs.eg.db,
                         ont = "BP",
                         pAdjustMethod = "BH",
                         pvalueCutoff = 0.05,
                         qvalueCutoff = 0.2,
                         readable = TRUE)
  go_bp_down_df <- as.data.frame(go_bp_down)
  if (nrow(go_bp_down_df) > 0) {
    write_tsv(go_bp_down_df, file.path(results_dir, "GO_BP_downregulated_results.tsv"))

    p4 <- dotplot(go_bp_down, showCategory = 20) +
      ggtitle("GO Biological Process - Downregulated Genes") +
      theme(axis.text.y = element_text(size = 8))
    ggsave(file.path(plots_dir, "GO_BP_down_dotplot.png"), p4, width = 12, height = 8)

    cat("Found", nrow(go_bp_down_df), "significant GO:BP terms (downregulated)\n")
  } else {
    cat("No significant GO:BP terms found (downregulated)\n")
  }
}
```

```
##
## === Running GO Biological Process Enrichment (Downregulated) ===
## No significant GO:BP terms found (downregulated)
```

```
# 4. Gene Ontology Molecular Function (GO:MF) - All significant genes
cat("\n=== Running GO Molecular Function Enrichment ===\n")
```

```
##
## === Running GO Molecular Function Enrichment ===
```

```
go_mf_all <- enrichGO(gene = entrez_all$ENTREZID,
                      OrgDb = org.Hs.eg.db,
                      ont = "MF",
                      pAdjustMethod = "BH",
                      pvalueCutoff = 0.05,
                      qvalueCutoff = 0.2,
                      readable = TRUE)

go_mf_all_df <- as.data.frame(go_mf_all)
if (nrow(go_mf_all_df) > 0) {
  write_tsv(go_mf_all_df, file.path(results_dir, "GO_MF_all_genes_results.tsv"))

  p5 <- dotplot(go_mf_all, showCategory = 20) +
    ggtitle("GO Molecular Function - All Significant Genes") +
    theme(axis.text.y = element_text(size = 8))
  ggsave(file.path(plots_dir, "GO_MF_all_dotplot.png"), p5, width = 12, height = 8)

  cat("Found", nrow(go_mf_all_df), "significant GO:MF terms\n")
} else {
  cat("No significant GO:MF terms found\n")
}
```

```
## No significant GO:MF terms found
```

```r
# 5. Gene Ontology Cellular Component (GO:CC) - All significant genes
cat("\n=== Running GO Cellular Component Enrichment ===\n")
```

```
##
## === Running GO Cellular Component Enrichment ===
```

```r
go_cc_all <- enrichGO(gene = entrez_all$ENTREZID,
                      OrgDb = org.Hs.eg.db,
                      ont = "CC",
                      pAdjustMethod = "BH",
                      pvalueCutoff = 0.05,
                      qvalueCutoff = 0.2,
                      readable = TRUE)

go_cc_all_df <- as.data.frame(go_cc_all)
if (nrow(go_cc_all_df) > 0) {
  write_tsv(go_cc_all_df, file.path(results_dir, "GO_CC_all_genes_results.tsv"))

  p6 <- dotplot(go_cc_all, showCategory = 20) +
    ggtitle("GO Cellular Component - All Significant Genes") +
    theme(axis.text.y = element_text(size = 8))
  ggsave(file.path(plots_dir, "GO_CC_all_dotplot.png"), p6, width = 12, height = 8)

  cat("Found", nrow(go_cc_all_df), "significant GO:CC terms\n")
} else {
  cat("No significant GO:CC terms found\n")
}
```

```
## Found 2 significant GO:CC terms
```

```r
# 6. Create comprehensive summary table
cat("\n=== Creating Summary Table ===\n")
```

```
##
## === Creating Summary Table ===
```

```r
summary_results <- data.frame()

# Helper function to add results to summary
add_to_summary <- function(results_df, method_name, ontology_name) {
  if (exists(deparse(substitute(results_df))) && nrow(results_df) > 0) {
    results_df %>%
      select(ID, Description, pvalue, p.adjust, qvalue, Count, GeneRatio, BgRatio) %>%
      mutate(Method = method_name,
             Ontology = ontology_name) %>%
      select(Method, Ontology, ID, Description, pvalue, p.adjust, qvalue, Count, GeneRatio, BgRatio)
  } else {
    data.frame()
  }
}
```

```r
# Add all results to summary
if (exists("go_bp_all_df") && nrow(go_bp_all_df) > 0) {
  summary_results <- rbind(summary_results,
                          add_to_summary(go_bp_all_df, "GO_BP_All", "Gene Ontology Biological Process"))
}

if (exists("go_bp_up_df") && nrow(go_bp_up_df) > 0) {
  summary_results <- rbind(summary_results,
                          add_to_summary(go_bp_up_df, "GO_BP_Up", "Gene Ontology Biological Process (Up
}

if (exists("go_bp_down_df") && nrow(go_bp_down_df) > 0) {
  summary_results <- rbind(summary_results,
                          add_to_summary(go_bp_down_df, "GO_BP_Down", "Gene Ontology Biological Process
}

if (exists("go_mf_all_df") && nrow(go_mf_all_df) > 0) {
  summary_results <- rbind(summary_results,
                          add_to_summary(go_mf_all_df, "GO_MF_All", "Gene Ontology Molecular Function"))
}

if (exists("go_cc_all_df") && nrow(go_cc_all_df) > 0) {
  summary_results <- rbind(summary_results,
                          add_to_summary(go_cc_all_df, "GO_CC_All", "Gene Ontology Cellular Component"))
}

# Save comprehensive results
if (nrow(summary_results) > 0) {
  summary_results <- summary_results %>%
    arrange(p.adjust)

  write_tsv(summary_results, file.path(results_dir, "GO_enrichment_comprehensive_results.tsv"))

  # Create top 100 results
  top_results <- summary_results %>%
    slice_head(n = 100)
  write_tsv(top_results, file.path(results_dir, "GO_enrichment_top100_results.tsv"))

  # Create method comparison
  method_summary <- summary_results %>%
    group_by(Method) %>%
    summarize(
      n_terms = n(),
      min_pvalue = min(pvalue),
      mean_pvalue = mean(pvalue),
      .groups = "drop"
    )

  write_tsv(method_summary, file.path(results_dir, "GO_method_comparison.tsv"))

  # Visualization of method comparison
  p_methods <- ggplot(method_summary, aes(x = Method, y = n_terms, fill = Method)) +
    geom_col() +
    labs(title = "Number of Significant GO Terms by Analysis Method",
         x = "Analysis Method",
         y = "Number of Significant Terms") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          legend.position = "none")
  ggsave(file.path(plots_dir, "GO_method_comparison.png"), p_methods, width = 10, height = 6)

  cat("\nSummary table created with", nrow(summary_results), "total enriched GO terms\n")
  cat("Results saved to:", file.path(results_dir, "GO_enrichment_comprehensive_results.tsv"), "\n")
} else {
  cat("No significant enrichment results found\n")
}
```

```
##
## Summary table created with 26 total enriched GO terms
## Results saved to: results/GO_enrichment_comprehensive_results.tsv
```

```r
# Final summary
cat("\n=== STEP 5 ANALYSIS COMPLETE ===\n")
```

```
##
## === STEP 5 ANALYSIS COMPLETE ===
```

```r
cat("Method: clusterProfiler (PyDESeq2-style approach)\n")
```

```
## Method: clusterProfiler (PyDESeq2-style approach)
```

```r
cat("Ontology: Gene Ontology (BP, MF, CC)\n")
```

```
## Ontology: Gene Ontology (BP, MF, CC)
```

```r
cat("Parameters used:\n")
```

```
## Parameters used:
```

```r
cat("- p-value cutoff: 0.05\n")
```

```
## - p-value cutoff: 0.05
```

```r
cat("- q-value cutoff: 0.2\n")
```

```
## - q-value cutoff: 0.2
```

```r
cat("- Multiple testing correction: Benjamini-Hochberg\n")
```

```
## - Multiple testing correction: Benjamini-Hochberg
```

```r
cat("- Gene ID conversion: Ensembl to Entrez\n")
```

```
## - Gene ID conversion: Ensembl to Entrez
```

```r
cat("\nInput:\n")
```

```
##
## Input:
```

```r
cat("- Total DE genes:", length(sig_genes_all), "\n")
```

```
## - Total DE genes: 48
```

```r
cat("- Upregulated genes:", length(sig_genes_up), "\n")
```

```
## - Upregulated genes: 27
```

```r
cat("- Downregulated genes:", length(sig_genes_down), "\n")
```

```
## - Downregulated genes: 21
```

```r
if (exists("summary_results") && nrow(summary_results) > 0) {
  cat("\nResults:\n")
  cat("- Total significant GO terms:", nrow(summary_results), "\n")
  cat("- Most significant term p-value:", format(min(summary_results$p.adjust), scientific = TRUE), "\n
}
```

```
##
## Results:
## - Total significant GO terms: 26
## - Most significant term p-value: 2.382021e-02
```

Section 5 Summary

```
## Gene set enrichment with clusterProfiler returned 26 significant Gene Ontology terms across BP, MF, 
```

```
## The topGO run contributed 200 Biological Process terms (with 96 passing weight01 < 0.05), and the su
```

Section 6

Section 6a: Consolidate enrichment results across methods

```r
library(dplyr)
library(readr)
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:magrittr':
##
##     extract

## The following object is masked from 'package:S4Vectors':
##
##     expand
```

```r
library(stringr)
library(purrr)
```

```
##
## Attaching package: 'purrr'

## The following object is masked from 'package:clusterProfiler':
##
##     simplify

## The following object is masked from 'package:GenomicRanges':
##
##     reduce

## The following object is masked from 'package:magrittr':
##
##     set_names

## The following object is masked from 'package:IRanges':
##
##     reduce
```

```r
# helper to tidy method names
clean_method_name <- function(x) {
  x %>%
    stringr::str_replace_all("[^A-Za-z0-9]+", "_") %>%
    stringr::str_replace_all("_+", "_") %>%
    stringr::str_replace_all("^_|_$", "") %>%
    stringr::str_to_lower()
}

parse_pvalue <- function(x) {
  if (is.numeric(x)) return(x)
  ifelse(is.na(x), NA_real_, suppressWarnings(as.numeric(stringr::str_replace_all(x, "<\\s*", ""))))
}

enrichment_tables <- list()

# clusterProfiler results from current session or saved file
if (exists("summary_results") && nrow(summary_results) > 0) {
  cp_long <- summary_results %>%
    transmute(
      source_method = "clusterProfiler",
      method_detail = Method,
      term_id = ID,
      term_name = Description,
      p_value = as.numeric(pvalue),
      adj_p_value = as.numeric(p.adjust),
      q_value = as.numeric(qvalue)
    ) %>%
    mutate(significant = ifelse(!is.na(adj_p_value), adj_p_value < 0.05, ifelse(!is.na(p_value), p_valu
  enrichment_tables <- append(enrichment_tables, list(cp_long))
} else {
  cp_path <- file.path(results_dir, "GO_enrichment_comprehensive_results.tsv")
  if (file.exists(cp_path)) {
    cp_file <- readr::read_tsv(cp_path, show_col_types = FALSE)
    cp_long <- cp_file %>%
      transmute(
        source_method = "clusterProfiler",
        method_detail = Method,
        term_id = ID,
        term_name = Description,
        p_value = as.numeric(pvalue),
        adj_p_value = as.numeric(p.adjust),
        q_value = as.numeric(qvalue)
      ) %>%
      mutate(significant = ifelse(!is.na(adj_p_value), adj_p_value < 0.05, ifelse(!is.na(p_value), p_val
    enrichment_tables <- append(enrichment_tables, list(cp_long))
  }
}

# clusterProfiler KEGG results
if (exists("kegg_results_df") && nrow(kegg_results_df) > 0) {
  method_label <- if (exists("run_label")) paste0("KEGG_", run_label) else "KEGG_current_run"
  kegg_long <- kegg_results_df %>%
    transmute(
      source_method = "clusterProfiler",
      method_detail = method_label,
      term_id = ID,
      term_name = Description,
      p_value = as.numeric(pvalue),
      adj_p_value = as.numeric(p.adjust),
      q_value = as.numeric(qvalue)
    ) %>%
    mutate(significant = ifelse(!is.na(adj_p_value), adj_p_value < 0.05, ifelse(!is.na(p_value), p_valu
  enrichment_tables <- append(enrichment_tables, list(kegg_long))
} else {
  kegg_files <- list.files(results_dir, pattern = "^KEGG_enrichment_.*\\.tsv$", full.names = TRUE)
  kegg_files <- kegg_files[!grepl("_top20_", kegg_files)]
  if (length(kegg_files) > 0) {
    kegg_tables <- purrr::map(kegg_files, ~{
      df <- readr::read_tsv(.x, show_col_types = FALSE)
      method_label <- tools::file_path_sans_ext(basename(.x))
      df %>%
        transmute(
          source_method = "clusterProfiler",
          method_detail = method_label,
          term_id = ID,
          term_name = Description,
          p_value = as.numeric(pvalue),
          adj_p_value = as.numeric(p.adjust),
          q_value = as.numeric(qvalue)
        ) %>%
        mutate(significant = ifelse(!is.na(adj_p_value), adj_p_value < 0.05, ifelse(!is.na(p_value), p_
    })
    enrichment_tables <- append(enrichment_tables, kegg_tables)
  }
}

# Team methods saved to results directory
topgo_files <- list.files(results_dir, pattern = "^topGO_.*\\.tsv$", full.names = TRUE)
if (length(topgo_files) > 0) {
  topgo_tables <- purrr::map(topgo_files, ~{
    df <- readr::read_tsv(.x, show_col_types = FALSE)
    method_label <- tools::file_path_sans_ext(basename(.x))
    p_cols <- c("weight01Fisher_p", "weight01Fisher", "classicFisher_p", "classicFisher", "Fisher")
    first_p <- purrr::detect(p_cols, ~ .x %in% names(df))
    p_vals <- if (!is.null(first_p)) parse_pvalue(df[[first_p]]) else rep(NA_real_, nrow(df))

    tibble::tibble(
      source_method = "topGO",
      method_detail = method_label,
      term_id = df$`GO.ID`,
      term_name = df$Term,
      p_value = p_vals,
      adj_p_value = NA_real_,
      q_value = NA_real_
    ) %>%
      mutate(significant = ifelse(!is.na(p_value), p_value < 0.05, FALSE))
  })
  enrichment_tables <- append(enrichment_tables, topgo_tables)
}
gss_files <- list.files(results_dir, pattern = "^GSS_.*\\.tsv$|^GSS_validated_signatures_.*\\.tsv$", ful
if (length(gss_files) > 0) {
  gss_tables <- purrr::map(gss_files, ~{
    df <- readr::read_tsv(.x, show_col_types = FALSE)
    method_label <- tools::file_path_sans_ext(basename(.x))
    df %>%
      transmute(
        source_method = "GSS",
        method_detail = method_label,
        term_id = as.character(term_id),
        term_name = as.character(term_name),
        p_value = NA_real_,
        adj_p_value = NA_real_,
        q_value = NA_real_,
        score = ifelse("score" %in% names(df), as.numeric(df$score), NA_real_)
      ) %>%
      mutate(significant = FALSE)
  })
  enrichment_tables <- append(enrichment_tables, gss_tables)
}

if (length(enrichment_tables) == 0) {
  warning("No enrichment result files found. Run Step 5 for each team method before executing Section 6
} else {
  enrichment_long <- bind_rows(enrichment_tables) %>%
    mutate(
      method_readable = paste(source_method, method_detail, sep = ": "),
      method_clean = clean_method_name(method_readable)
    )

  min_non_na <- function(...) {
    vals <- c(...)
    vals <- vals[!is.na(vals)]
    if (length(vals) == 0) NA_real_ else min(vals)
  }

  metrics_to_pivot <- c("p_value", "adj_p_value", "q_value", "significant")

  metrics_wide <- enrichment_long %>%
    pivot_longer(cols = all_of(metrics_to_pivot), names_to = "metric", values_to = "metric_value") %>%
    mutate(metric_column = paste(method_clean, metric, sep = "_")) %>%
    select(term_id, term_name, metric_column, metric_value) %>%
    distinct() %>%
    pivot_wider(names_from = metric_column, values_from = metric_value)

  method_counts <- enrichment_long %>%
    group_by(term_id, term_name) %>%
    summarise(
      methods_included = n_distinct(method_readable),
      methods_significant = sum(significant, na.rm = TRUE),
      methods_list = paste(sort(unique(method_readable)), collapse = "; "),
      min_p_value = min_non_na(p_value, adj_p_value, q_value),
      .groups = "drop"
    )

  combined_table <- method_counts %>%
    left_join(metrics_wide, by = c("term_id", "term_name")) %>%
    arrange(desc(methods_significant), desc(methods_included), min_p_value)

  assign("combined_enrichment_table", combined_table, envir = .GlobalEnv)

  combined_path <- file.path(results_dir, "combined_enrichment_summary.tsv")
  readr::write_tsv(combined_table, combined_path)

  cat(sprintf("Combined enrichment summary saved to %s with %d unique terms across %d method contributi
              combined_path, nrow(combined_table), length(unique(enrichment_long$method_readable))))
}
```

```
## Combined enrichment summary saved to results/combined_enrichment_summary.tsv with 251 unique terms a
```

Section 6b Summary

```
## Cross-method comparison pooled 9 enrichment outputs, highlighting 143 terms that were significant in
```

Section 7

Section 7a: Top shared enrichment signals

```
if (exists("combined_enrichment_table")) {
  top_terms <- combined_enrichment_table %>%
    arrange(desc(methods_significant), desc(methods_included), min_p_value) %>%
    { .[seq_len(min(10, nrow(.))), , drop = FALSE] }

  assign("top_shared_terms", top_terms, envir = .GlobalEnv)

  top10_path <- file.path(results_dir, "combined_enrichment_top10.tsv")
  readr::write_tsv(top_terms, top10_path)

  print(top_terms %>% select(term_id, term_name, methods_significant, methods_included, min_p_value))
  cat(sprintf("\nTop consensus terms saved to %s.\n", top10_path))
} else {
  cat("Run Section 6 first to build the combined enrichment table.\n")
}
```

```
## # A tibble: 10 x 5
##    term_id    term_name          methods_significant methods_included min_p_value
##    <chr>      <chr>                            <int>            <int>       <dbl>
##  1 GO:0070266 necroptotic proc~                    4                4   0.0000304
##  2 GO:0016045 detection of bac~                    3                3   0.000573
##  3 GO:0031952 regulation of pr~                    2                2   0.0000364
##  4 GO:0097300 programmed necro~                    2                2   0.0000626
##  5 GO:0000028 ribosomal small ~                    2                2   0.000064
##  6 GO:0051338 regulation of tr~                    2                2   0.0000671
##  7 GO:0010562 positive regulat~                    2                2   0.0000993
##  8 GO:0045937 positive regulat~                    2                2   0.0000993
##  9 GO:0033674 positive regulat~                    2                2   0.000133
## 10 GO:1903428 positive regulat~                    2                2   0.00063
##
## Top consensus terms saved to results/combined_enrichment_top10.tsv.
```

Section 7b Summary

```
## Across methods, necroptotic process emerged as the most consistently enriched term, reported by 4 me
```