```
Differential expression analysis using DESeq2. This script performs differential gene
expression analysis on RNA-seq data quantified with RSEM

# ------------------------------
# 1) Load required libraries
# ------------------------------
library(DESeq2) # differential expression analysis
library(dplyr) #data manipulation (not strictly required here, but commonly used)
library(ggplot2) #plotting (used later if you want to visualize)
library(biomaRt) #for gene annotation
library(EnhancedVolcano) # for volcano plots

# ------------------------------
# 2) Create a metadata of samples
# ------------------------------
# This block constructs a small metadata table (sample names, corresponding files, and
condition of the sample).
# 'sample' should match the column names you want in the final count matrix.
# 'file' should be the filenames of RSEM or other gene-level results files.
# 'condition' is the experimental factor; convert to factor before DESeq2.
# set working directory to where your genes.results files are
samples <- data.frame(
sample = c("BY4272_R1","BY4272_R2","BY4272_R3",
             "SY14_R1","SY14_R2","SY14_R3"),
  file = c("BY4272_R1.genes.results","BY4272_R2.genes.results",
           "BY4272_R3.genes.results","SY14_R1.genes.results",
           "SY14_R2.genes.results","SY14_R3.genes.results"),
  condition = c("WT","WT","WT","SY14","SY14","SY14"),
  stringsAsFactors = FALSE
)

# Save metadata to CSV for record-keeping.
write.csv(samples, "metadata.csv", row.names = FALSE)

# ------------------------------
# 3) Read gene.results files & build count matrix
# ------------------------------
# This function reads one file and extracts gene_id and expected_count columns.
# read table: header = TRUE (first row column names), sep = "\t" for tab-delimited
# keep only gene identifier and the expected_count column (RSEM output).

read_rsem <- function(file, sample_name){
  df <- read.table(file, header = TRUE, sep = "\t", stringsAsFactors = FALSE)
  df <- df[, c("gene_id", "expected_count")]
  # rename the expected_count column to the sample name
  colnames(df)[2] <- sample_name
  return(df)
}
# Read the sample file list from the metadata CSV we just wrote
samples <- read.csv("metadata.csv", stringsAsFactors = FALSE)
# mapply is used to iterate over all input files and read them into a list of data frames
count_list <- mapply(
  FUN = read_rsem,
  file = samples$file,
  sample_name = samples$sample,
  SIMPLIFY = FALSE
)

# Merge the list of data.frames by the gene_id column to produce a combined counts table.
# Reduce with merge will progressively join all frames; this assumes all files share the same
gene_id set.
counts <- Reduce(function(x, y) merge(x, y, by = "gene_id", all = TRUE), count_list)

# Replace NA values in numeric columns (sample columns) with 0 -1 tells it to exclude gene_id
from this function because it has mix characters
```

```
counts[, -1][is.na(counts[, -1])] <- 0

# Round numeric counts to integers
counts[, -1] <- round(as.matrix(counts[, -1]))

# reorder columns to match metadata order
counts <- counts[, c("gene_id", samples$sample)]

# Save the counts matrix to CSV for record-keeping
write.csv(counts, "counts_matrix.csv", quote = FALSE, row.names = FALSE)

# ------------------------------
# 4) Create DESeq2 dataset and run analysis
# ------------------------------
# Convert the condition column to a factor so R understands it as categorical
metadata <- samples
rownames(metadata) <- metadata$sample
metadata$condition <- factor(metadata$condition)
str(metadata$condition) #shows the structure

# Create DESeqDataSetFromMatrix:
# countData: integer count matrix (rows = genes, columns = samples)
# colData: metadata dataframe (must have rownames matching colnames(countData) or contain a
'sample' column)
# design: formula specifying the model; here ~ condition
# ensure rownames(samples) match column names of counts or provide colData with rownames set.
rownames(metadata) <- metadata$sample

# Prepare numeric count matrix for DESeq2 (essentially removes the gene_id column for DESeq
from the count matrix because it is not numerical )
count_matrix <- counts[, -1]         # remove gene_id column
rownames(count_matrix) <- counts$gene_id

# Ensure metadata rownames match count columns
rownames(metadata) <- metadata$sample
metadata$condition <- factor(metadata$condition)

# Verify match
all(colnames(count_matrix) == rownames(metadata))  # should be TRUE as they all need to match

# Create DESeq2 object
library(DESeq2)
dds <- DESeqDataSetFromMatrix(
    countData = count_matrix,
    colData = metadata,
    design = ~ condition
)

# Run DESeq2
dds <- DESeq(dds)


# ------------------------------
# 5) Extract results
# ------------------------------
# res() extracts the differential expression results.
# Here we want WT vs SY14 (log2FoldChange positive means SY14 has higher expression than WT)
# Get DESeq2 results: SY14 vs WT
res <- results(dds, contrast = c("condition","SY14","WT"))

# Sort by adjusted p-value (put NA padj at the bottom)
res <- res[order(res$padj, na.last = TRUE), ]

# Convert to data frame and add gene_id column
res_df <- as.data.frame(res)
res_df$gene_id <- rownames(res_df)
```

```
# adding the biotype column to filter protein coding genes

library(rtracklayer)
gtf <- import("GCF_000146045.2_R64_genomic.gtf")
gtf_gene <- gtf[gtf$type == "gene"]
annotated <- as.data.frame(mcols(gtf_gene))[, c("gene_id", "gene", "gene_biotype")]
colnames(annotated) <- c("gene_id", "gene_name", "gene_biotype")
table(annotated$gene_biotype)
res_annotated <- merge(res_df, annotated, by = "gene_id", all.x = TRUE)


# Saving the results with protein coding genes

write.csv(res_annotated, "genes_with_biotype.csv", row.names = FALSE)

# ------------------------------
# 6) Filter significant genes
# ------------------------------

sig_final <- res[which(res$padj < 0.001 & abs(res$log2FoldChange) > 1), ]
nrow(sig_final)

# adding the biotype column to filter protein coding significant genes

sig_final_df <- as.data.frame(sig_final)
sig_final_df$gene_id <- rownames(sig_final_df)
library(rtracklayer)
gtf <- import("GCF_000146045.2_R64_genomic.gtf")
gtf_gene <- gtf[gtf$type == "gene"]
annot_df <- as.data.frame(mcols(gtf_gene))[, c("gene_id", "gene", "gene_biotype")]
colnames(annot_df) <- c("gene_id", "gene_name", "gene_biotype")
annot_df <- as.data.frame(annot_df)
sig_final_with_biotype <- base::merge (sig_final_df, annot_df, by = "gene_id", all.x = TRUE)


# saving significant genes with biotype

write.csv(sig_final_with_biotype,file = "sig_DESeq_results_SY14_vs_WT_with_biotype.csv",quote
= FALSE,row.names = FALSE)

# to filter out all genes except protein coding

sig_protein <- subset(sig_final_with_biotype, gene_biotype == "protein_coding")
nrow(sig_protein)

#save file with only protein coding DEG

write.csv(sig_protein,file = "sig_DESeq_with_protein_coding.csv",quote = FALSE,row.names =
FALSE)

## optional this helps separate genes that are over and under expressed by adding a column
sig_protein$DEG[sig_protein$padj < 0.001 & sig_protein$log2FoldChange > 1]  <- "Up"
sig_protein$DEG[sig_protein$padj < 0.001 & sig_protein$log2FoldChange < -1] <- "Down"

# save if created
```