This documentation outlines the steps for performing differential gene expression analysis between SARS and mock conditions using both **edgeR** and **DESeq2** libraries in R. The code also includes visualization techniques such as **PCA plots** and **heatmaps** to analyze sample clustering.

1. Loading Necessary Libraries

We begin by loading the essential libraries for the analysis: `DESeq2`, `edgeR`, `tidyverse`, `pheatmap`, and `ashr`. These libraries help perform differential expression analysis, normalization, statistical testing, and data visualization.

```r
# Load necessary libraries
library(DESeq2)
library(edgeR)
library(tidyverse)
library(pheatmap)
library(ggplot2)
library(ashr)
```

2. Loading the Data and Defining Threshold

We load the raw count data (`GSE159717_rnaseq_deseq_5dpi_counts_raw.tsv`) and create a count matrix and condition vector for the analysis. The conditions include three groups: `SARS`, `mock`, and `Rem`.

```r
# Significance threshold for adjusted p-values (FDR)
signif_threshold <- 0.2

# Load data - counts and metadata
data <- read.delim("GSE159717_rnaseq_deseq_5dpi_counts_raw.tsv", header = TRUE, sep = "\t")

# Prepare count matrix (expression data)
count_data <- as.matrix(data[, grepl("^S_", colnames(data))])
rownames(count_data) <- data$gene_name

# Define conditions for the analysis (COVID-19 vs mock)
condition <- factor(c("Rem", "SARS", "mock", "Rem", "SARS", "mock"))
condition <- relevel(condition, ref = "SARS")  # Set "SARS" as the reference
```

3. Differential Expression Analysis Using edgeR

We proceed to the differential expression analysis using the **edgeR** package. The steps include normalizing the counts using the TMM method, fitting a GLM, and performing a likelihood ratio test (LRT) for the comparison between **mock** and **SARS**.

```r
# Create DGEList object for edgeR
dge <- DGEList(counts = count_data)

# Filter low-expression genes: keep genes with CPM > 1 in at least two samples
keep <- rowSums(cpm(dge) > 1) >= 2
dge <- dge[keep, , keep.lib.sizes = FALSE]  # Filtered DGEList object

# Normalize counts using TMM (to account for library size differences)
dge <- calcNormFactors(dge)

# Estimate dispersions
dge <- estimateDisp(dge, design)

# Fit generalized linear model (GLM) to normalized data
fit <- glmFit(dge, design)

# Likelihood ratio test for "mock" vs "SARS"
lrt_mock_vs_sars <- glmLRT(fit, coef = 2)

# Retrieve table of top genes by p-value
res_edgeR_mock_vs_sars <- topTags(lrt_mock_vs_sars, n = nrow(dge))$table

# Adjust p-values (Benjamini-Hochberg method)
res_edgeR_mock_vs_sars$FDR <- p.adjust(res_edgeR_mock_vs_sars$PValue, method = "BH")
```

4. Differential Expression Analysis Using DESeq2

The next step is performing differential gene expression analysis using **DESeq2**. This includes running the differential expression pipeline and performing **log fold change shrinkage** for small sample sizes

```r
# Create DESeq2 dataset
dds <- DESeqDataSetFromMatrix(countData = count_data, colData = data.frame(condition), design = ~ condition)

# Run DESeq2 pipeline (normalization, dispersion estimation, and differential expression)
dds <- DESeq(dds)

# Get results from DESeq2 for "mock" vs "SARS"
res_DESeq2_mock_vs_sars <- results(dds, contrast = c("condition", "mock", "SARS"), alpha = 0.3)

# Perform log fold change shrinkage (recommended for small sample sizes)
res_DESeq2_mock_vs_sars <- lfcShrink(dds, contrast = c("condition", "mock", "SARS"), res = res_DESeq2_mock_v
```

5. Combining Results from edgeR and DESeq2

To ensure consistency, the downregulated genes from both **edgeR** and **DESeq2** are intersected, and the top common genes are listed.

```r
# Combine results: Intersection of downregulated genes from both tools for mock vs SARS
common_downregulated_mock_vs_sars <- intersect(downregulated_DESeq2_mock_vs_sars, downregulated_edgeR_mock_

write.table(common_downregulated_mock_vs_sars, "common_downregulated_mock_vs_sars.txt", quote = FALSE, row.

print(paste("Common downregulated genes (mock vs SARS):", length(common_downregulated_mock_vs_sars)))


# ---------------------- Merge Results and Visualization ----------------------
# Pull log2 fold change and padj values for common downregulated genes in DESeq2
common_DESeq2_mock_vs_sars <- res_DESeq2_mock_vs_sars[common_downregulated_mock_vs_sars, c("log2FoldChange"
common_DESeq2_mock_vs_sars <- as.data.frame(common_DESeq2_mock_vs_sars)
common_DESeq2_mock_vs_sars$Gene <- rownames(common_DESeq2_mock_vs_sars)
```

6. PCA Plot for Sample Clustering

We generate a **PCA plot** to visualize sample clustering based on the gene expression data.

```r
# Perform PCA using variance stabilizing transformation (VST)
rld <- rlog(dds, blind = TRUE)

# PCA plot grouped by condition
plotPCA(rld, intgroup = "condition")
```

7. Heatmap for Hierarchical Clustering

We create a **heatmap** to visualize the hierarchical clustering of samples based on their gene expression profiles. The heatmap allows us to see how similar or distinct the samples are.

```r
# Compute sample distances based on the rlog-transformed data
sample_dists <- dist(t(assay(rld)))
sample_dist_matrix <- as.matrix(sample_dists)

annotation_col <- data.frame(condition = condition)
rownames(annotation_col) <- colnames(count_data)  # Assign sample names

# Heatmap of sample distances with hierarchical clustering
pheatmap(sample_dist_matrix,
         clustering_distance_rows = sample_dists,
         clustering_distance_cols = sample_dists,
         annotation_col = annotation_col)
```