R Analysis

Dina Sami - Heba Amer

2025-02-20

Contents

Simplifying the Table

```r
gene_counts <- read.table("gene_counts.txt", header = TRUE, sep = "\t", row.names = 1)

colnames(gene_counts)
```

```
##  [1] "Chr"
##  [2] "Start"
##  [3] "End"
##  [4] "Strand"
##  [5] "Length"
##  [6] "aligned_data.LUT.105_R_S25_Aligned.sortedByCoord.out.bam"
##  [7] "aligned_data.LUT.114_R_S26_Aligned.sortedByCoord.out.bam"
##  [8] "aligned_data.LUT.19_R_S27_Aligned.sortedByCoord.out.bam"
##  [9] "aligned_data.UTL.101_R_S30_Aligned.sortedByCoord.out.bam"
## [10] "aligned_data.UTL.103_R_S31_Aligned.sortedByCoord.out.bam"
## [11] "aligned_data.UTL.10_R_cont__S28_Aligned.sortedByCoord.out.bam"
## [12] "aligned_data.UTL.10_R_nash__S29_Aligned.sortedByCoord.out.bam"
## [13] "aligned_data.UTL.110_R_S33_Aligned.sortedByCoord.out.bam"
## [14] "aligned_data.UTL.118_R_S34_Aligned.sortedByCoord.out.bam"
## [15] "aligned_data.UTL.122_R_S35_Aligned.sortedByCoord.out.bam"
## [16] "aligned_data.UTL.129_R_S36_Aligned.sortedByCoord.out.bam"
## [17] "aligned_data.UTL.131_R_S37_Aligned.sortedByCoord.out.bam"
## [18] "aligned_data.UTL.132_R_S38_Aligned.sortedByCoord.out.bam"
## [19] "aligned_data.UTL.137_R_S39_Aligned.sortedByCoord.out.bam"
## [20] "aligned_data.UTL.16_R_S40_Aligned.sortedByCoord.out.bam"
## [21] "aligned_data.UTL.29_R_S41_Aligned.sortedByCoord.out.bam"
## [22] "aligned_data.UTL.39_R_S42_Aligned.sortedByCoord.out.bam"
## [23] "aligned_data.UTL.44_R_S43_Aligned.sortedByCoord.out.bam"
## [24] "aligned_data.UTL.54_R_S44_Aligned.sortedByCoord.out.bam"
## [25] "aligned_data.UTL.55_R_S45_Aligned.sortedByCoord.out.bam"
## [26] "aligned_data.UTL.57_R_S46_Aligned.sortedByCoord.out.bam"
## [27] "aligned_data.UTL.68_R_S48_Aligned.sortedByCoord.out.bam"
## [28] "aligned_data.UTL.6_R_S47_Aligned.sortedByCoord.out.bam"
## [29] "aligned_data.UTL.71_R_S49_Aligned.sortedByCoord.out.bam"
## [30] "aligned_data.UTL.78_R_S50_Aligned.sortedByCoord.out.bam"
```

```r
gene_counts <- gene_counts[, -c(1:5)]

colnames(gene_counts)
```

```
##  [1] "aligned_data.LUT.105_R_S25_Aligned.sortedByCoord.out.bam"
##  [2] "aligned_data.LUT.114_R_S26_Aligned.sortedByCoord.out.bam"
##  [3] "aligned_data.LUT.19_R_S27_Aligned.sortedByCoord.out.bam"
##  [4] "aligned_data.UTL.101_R_S30_Aligned.sortedByCoord.out.bam"
##  [5] "aligned_data.UTL.103_R_S31_Aligned.sortedByCoord.out.bam"
##  [6] "aligned_data.UTL.10_R_cont__S28_Aligned.sortedByCoord.out.bam"
##  [7] "aligned_data.UTL.10_R_nash__S29_Aligned.sortedByCoord.out.bam"
##  [8] "aligned_data.UTL.110_R_S33_Aligned.sortedByCoord.out.bam"
##  [9] "aligned_data.UTL.118_R_S34_Aligned.sortedByCoord.out.bam"
## [10] "aligned_data.UTL.122_R_S35_Aligned.sortedByCoord.out.bam"
## [11] "aligned_data.UTL.129_R_S36_Aligned.sortedByCoord.out.bam"
## [12] "aligned_data.UTL.131_R_S37_Aligned.sortedByCoord.out.bam"
## [13] "aligned_data.UTL.132_R_S38_Aligned.sortedByCoord.out.bam"
## [14] "aligned_data.UTL.137_R_S39_Aligned.sortedByCoord.out.bam"
## [15] "aligned_data.UTL.16_R_S40_Aligned.sortedByCoord.out.bam"
## [16] "aligned_data.UTL.29_R_S41_Aligned.sortedByCoord.out.bam"
## [17] "aligned_data.UTL.39_R_S42_Aligned.sortedByCoord.out.bam"
## [18] "aligned_data.UTL.44_R_S43_Aligned.sortedByCoord.out.bam"
## [19] "aligned_data.UTL.54_R_S44_Aligned.sortedByCoord.out.bam"
## [20] "aligned_data.UTL.55_R_S45_Aligned.sortedByCoord.out.bam"
## [21] "aligned_data.UTL.57_R_S46_Aligned.sortedByCoord.out.bam"
## [22] "aligned_data.UTL.68_R_S48_Aligned.sortedByCoord.out.bam"
## [23] "aligned_data.UTL.6_R_S47_Aligned.sortedByCoord.out.bam"
## [24] "aligned_data.UTL.71_R_S49_Aligned.sortedByCoord.out.bam"
## [25] "aligned_data.UTL.78_R_S50_Aligned.sortedByCoord.out.bam"
```

```
new_colnames <- sub(".*_(S[0-9]+)_.*", "\\1", colnames(gene_counts))

colnames(gene_counts) <- new_colnames

colnames(gene_counts)
```

```
##  [1] "S25" "S26" "S27" "S30" "S31" "S28" "S29" "S33" "S34" "S35" "S36" "S37"
## [13] "S38" "S39" "S40" "S41" "S42" "S43" "S44" "S45" "S46" "S48" "S47" "S49"
## [25] "S50"
```

```
# Create a mapping of sample IDs to conditions with "Control"
sample_conditions <- data.frame(
  Sample = c("S25", "S26", "S27", "S28", "S29", "S30", "S31", "S33", "S34", "S35", "S36", "S37", "S38",
  Condition = c("EtOH", "Control", "Control", "Control", "NASH", "NASH", "Control", "NASH", "NASH", "Et
)

# Subset the gene_counts data frame based on conditions
EtOH_samples <- sample_conditions$Sample[sample_conditions$Condition == "EtOH"]
Control_samples <- sample_conditions$Sample[sample_conditions$Condition == "Control"]
NASH_samples <- sample_conditions$Sample[sample_conditions$Condition == "NASH"]

gene_counts_EtOH <- gene_counts[, EtOH_samples]
gene_counts_Control <- gene_counts[, Control_samples]
gene_counts_NASH <- gene_counts[, NASH_samples]

# Display the first few rows of each subset
list(EtOH = head(gene_counts_EtOH), Control = head(gene_counts_Control), NASH = head(gene_counts_NASH))
```

```
## $EtOH
##                 S25 S35 S36 S39 S40 S41 S42 S47 S49 S50
## ENSG00000228037   0   0   0   7   0   1   3   0   7   3
## ENSG00000142611  11   6  19  18  19  12  13  41  22  16
## ENSG00000284616   0   0   0   1   0   1   0   0   1   0
## ENSG00000157911  54  60  61  32  40 100  41  60 111 124
## ENSG00000260972   0   0   0   0   0   0   2   0   0   0
## ENSG00000224340   0   0   0   2   0   0   1   0   0   0
##
## $Control
##                 S26 S27 S28 S31 S43 S48
## ENSG00000228037   6   0  21   0   0   2
## ENSG00000142611   8   8  88   0   4   7
## ENSG00000284616   0   0   0   0   0   0
## ENSG00000157911 141 146 213 108  98  79
## ENSG00000260972   0   0   0   0   0   0
## ENSG00000224340   0   0   0   1   0   1
##
## $NASH
##                 S29 S30 S33 S34 S37 S38 S44 S45 S46
## ENSG00000228037   0   1   0   2   0   2   0   0   1
## ENSG00000142611  20   4  64   5   5  10  23   5  10
## ENSG00000284616   0   0   0   0   0   0   0   0   0
## ENSG00000157911  87 145 291  99  36  32  55  20  76
## ENSG00000260972   1   0   0   0   0   0   0   0   0
## ENSG00000224340   0   0   0   1   0   0   0  15   0
```

```
# Rename columns for each condition
colnames(gene_counts_Control) <- paste0("C_", colnames(gene_counts_Control))
colnames(gene_counts_EtOH) <- paste0("E_", colnames(gene_counts_EtOH))
colnames(gene_counts_NASH) <- paste0("N_", colnames(gene_counts_NASH))

# Display the column names of each subset
list(Control = colnames(gene_counts_Control), EtOH = colnames(gene_counts_EtOH), NASH = colnames(gene_c
```

```
## $Control
## [1] "C_S26" "C_S27" "C_S28" "C_S31" "C_S43" "C_S48"
##
## $EtOH
##  [1] "E_S25" "E_S35" "E_S36" "E_S39" "E_S40" "E_S41" "E_S42" "E_S47" "E_S49"
## [10] "E_S50"
##
## $NASH
## [1] "N_S29" "N_S30" "N_S33" "N_S34" "N_S37" "N_S38" "N_S44" "N_S45" "N_S46"
```

```
# Merge Control, EtOH, and NASH data frames
gene_counts_all <- cbind(gene_counts_Control, gene_counts_EtOH, gene_counts_NASH)

# Display the first few rows of the merged data frame
head(gene_counts_all)
```

```
##                 C_S26 C_S27 C_S28 C_S31 C_S43 C_S48 E_S25 E_S35 E_S36 E_S39
## ENSG00000228037     6     0    21     0     0     2     0     0     0     7
## ENSG00000142611     8     8    88     0     4     7    11     6    19    18
## ENSG00000284616     0     0     0     0     0     0     0     0     0     1
## ENSG00000157911   141   146   213   108    98    79    54    60    61    32
## ENSG00000260972     0     0     0     0     0     0     0     0     0     0
## ENSG00000224340     0     0     0     1     0     1     0     0     0     2
##                 E_S40 E_S41 E_S42 E_S47 E_S49 E_S50 N_S29 N_S30 N_S33 N_S34
## ENSG00000228037     0     1     3     0     7     3     0     1     0     2
## ENSG00000142611    19    12    13    41    22    16    20     4    64     5
## ENSG00000284616     0     1     0     0     1     0     0     0     0     0
## ENSG00000157911    40   100    41    60   111   124    87   145   291    99
## ENSG00000260972     0     0     2     0     0     0     1     0     0     0
## ENSG00000224340     0     0     1     0     0     0     0     0     0     1
##                 N_S37 N_S38 N_S44 N_S45 N_S46
## ENSG00000228037     0     2     0     0     1
## ENSG00000142611     5    10    23     5    10
## ENSG00000284616     0     0     0     0     0
## ENSG00000157911    36    32    55    20    76
## ENSG00000260972     0     0     0     0     0
## ENSG00000224340     0     0     0    15     0
```

```
# Save the merged data frame to a file
write.table(gene_counts_all, "raw_gene_counts_all.txt", sep = "\t", quote = FALSE, col.names = NA)
```

```
# Define the columns you want to keep
columns_to_keep <- c("C_S27", "C_S28", "C_S31", "C_S43", "C_S48",
                     "E_S36", "E_S41", "E_S42", "E_S49", "E_S50",
                     "N_S29", "N_S30", "N_S37", "N_S38", "N_S46")
```

```r
# Subset the dataframe to keep only the specified columns
gene_counts_subset <- gene_counts_all[, columns_to_keep]


# Save the merged data frame to a file
write.table(gene_counts_subset, "gene_counts_all.txt", sep = "\t", quote = FALSE, col.names = NA)
```

Introduction

This analysis focuses on gene expression differences between Control, EtOH, and NASH samples using DESeq2. We will load the gene count data, perform filtering, and run the DESeq2 analysis step by step.

Gene Counts Analysis

Library Installation and Loading

The following block installs necessary libraries (if not already installed) and loads them for subsequent use. It also includes the installation of annotation databases for gene mapping.

```r
# Install necessary libraries if not already installed
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(c("DESeq2", "pheatmap", "ggplot2", "EnhancedVolcano","AnnotationDbi","org.Hs.eg.db

# Load necessary libraries
library(DESeq2)
library(pheatmap)
library(ggplot2)
library(EnhancedVolcano)
library("org.Hs.eg.db")
library(clusterProfiler)
```

Load the Gene Count Data

In this step, we will load the gene count data from a tab-separated file called `gene_counts_all.tsv`. The data frame contains gene counts for each sample, with genes as rows and samples as columns.

```r
# Load the gene counts data
gene_counts <- read.table("gene_counts_all.txt", header = TRUE, sep = "\t", row.names = 1)

# Display the first few rows of the data frame
colnames(gene_counts)
```

```
##  [1] "C_S27" "C_S28" "C_S31" "C_S43" "C_S48" "E_S36" "E_S41" "E_S42" "E_S49"
## [10] "E_S50" "N_S29" "N_S30" "N_S37" "N_S38" "N_S46"
```

Prepare Metadata

This step involves creating a metadata (colData) data frame that describes the condition for each sample. The conditions include Control, EtOH, and NASH samples.

```r
# Create a metadata (colData) data frame that describes the conditions for each sample
sample_conditions <- data.frame(
  row.names = colnames(gene_counts),
  condition = c(rep("Control", 5), rep("EtOH", 5), rep("NASH", 5))
)

# Display the metadata
sample_conditions
```

```
##       condition
## C_S27   Control
## C_S28   Control
## C_S31   Control
## C_S43   Control
## C_S48   Control
## E_S36      EtOH
## E_S41      EtOH
## E_S42      EtOH
## E_S49      EtOH
## E_S50      EtOH
## N_S29      NASH
## N_S30      NASH
## N_S37      NASH
## N_S38      NASH
## N_S46      NASH
```

Filter Out Zero Counts

Here, we filter out genes with all zero counts across all samples, as they provide no information for analysis.

```
# Filter out genes with all zero counts
gene_counts_filtered <- gene_counts[rowSums(gene_counts != 0) > 0, ]

# Display the first few rows of the filtered data frame
dim(gene_counts_filtered)
```

```
## [1] 48403    15
```

Filter by Mean Expression

After removing genes with zero counts, we further filter genes based on their mean expression across samples. Only genes with a mean expression greater than a specified threshold 10 are kept for analysis.

```
# Calculate the mean expression for each gene
gene_means <- rowMeans(gene_counts_filtered)

# Filter out genes with mean expression below a certain threshold (e.g., mean < 10)
mean_threshold <- 10
gene_counts_filtered <- gene_counts_filtered[gene_means >= mean_threshold, ]

# Display the first few rows of the filtered data frame
dim(gene_counts_filtered)
```

```
## [1] 16018    15
```

Create DESeq2 Dataset

DESeq2 requires a DESeqDataSet object, which is created by providing the count data and sample information (metadata). Here, we create the dataset using the filtered gene count data.

```r
# Create the DESeqDataSet with counts
dds_filtered <- DESeqDataSetFromMatrix(countData = gene_counts_filtered,
                                       colData = sample_conditions,
                                       design = ~ condition)


# Display the DESeqDataSet object
dds_filtered
```

```
## class: DESeqDataSet
## dim: 16018 15
## metadata(1): version
## assays(1): counts
## rownames(16018): ENSG00000142611 ENSG00000157911 ... ENSG00000276345
##   ENSG00000271254
## rowData names(0):
## colnames(15): C_S27 C_S28 ... N_S38 N_S46
## colData names(1): condition
```

Run DESeq2

Now that we have the dataset ready, we run the DESeq2 analysis to identify differentially expressed genes between the conditions.

```r
# Run the DESeq2 differential expression analysis
dds_filtered <- DESeq(dds_filtered)

dds_filtered
```

```
## class: DESeqDataSet
## dim: 16018 15
## metadata(1): version
## assays(4): counts mu H cooks
## rownames(16018): ENSG00000142611 ENSG00000157911 ... ENSG00000276345
##   ENSG00000271254
## rowData names(26): baseMean baseVar ... deviance maxCooks
## colnames(15): C_S27 C_S28 ... N_S38 N_S46
## colData names(2): condition sizeFactor
```

Extract Results for Control vs EtOH

We are extracting the results of the differential expression analysis for the Control vs EtOH comparison.

```r
# Extract the results for EtOH vs Control
results_etoh <- results(dds_filtered, contrast = c("condition", "EtOH", "Control"))

# Display the first few rows of the results
summary(results_etoh)
```

```
##
## out of 16018 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 1409, 8.8%
## LFC < 0 (down)     : 1283, 8%
## outliers [1]       : 138, 0.86%
## low counts [2]     : 0, 0%
## (mean count < 7)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Extract Results for NASH vs Control

We are extracting the results of the differential expression analysis for the NASH vs Control comparison.

```r
#Extract results for NASH:

results_nash <- results(dds_filtered, contrast = c("condition", "NASH", "Control"))


summary(results_nash)
```

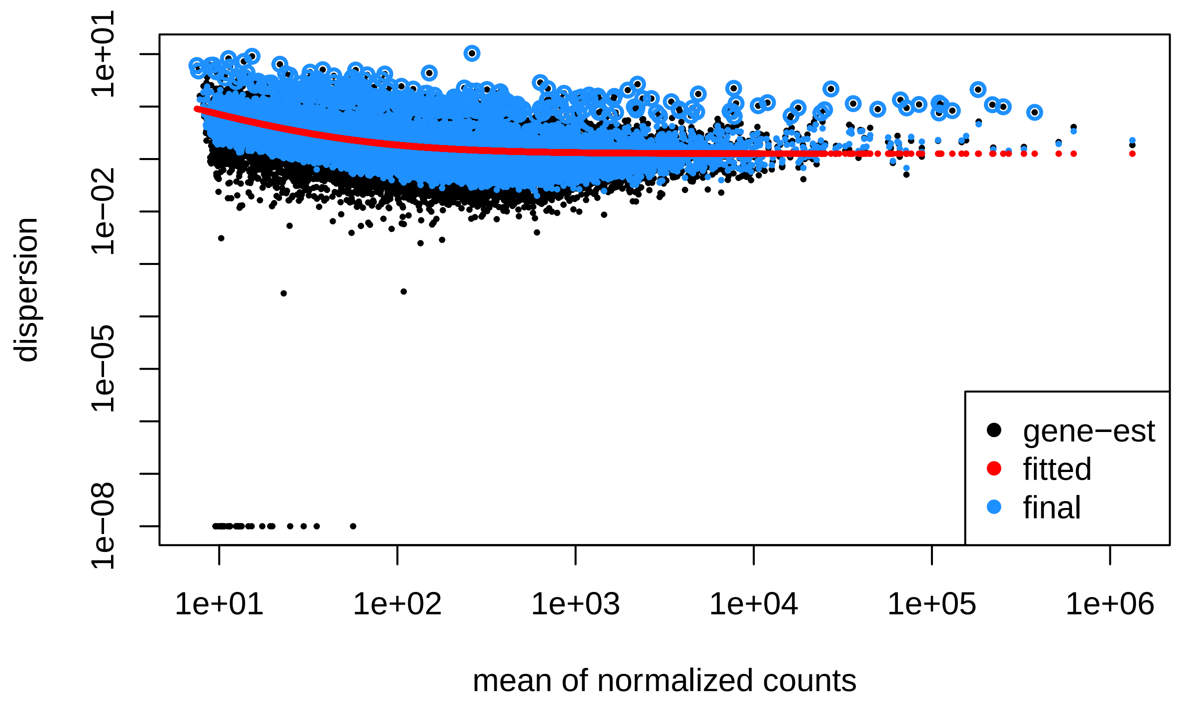
```
##
## out of 16018 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 62, 0.39%
## LFC < 0 (down)     : 75, 0.47%
## outliers [1]       : 138, 0.86%
## low counts [2]     : 0, 0%
## (mean count < 7)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Quality Check for Data and Analysis

Plot Dispersion Estimates

We are plotting the dispersion estimates to visualize gene-wise dispersion across samples.

```r
#Plot the dispersion estimates:

plotDispEsts(dds_filtered)
```



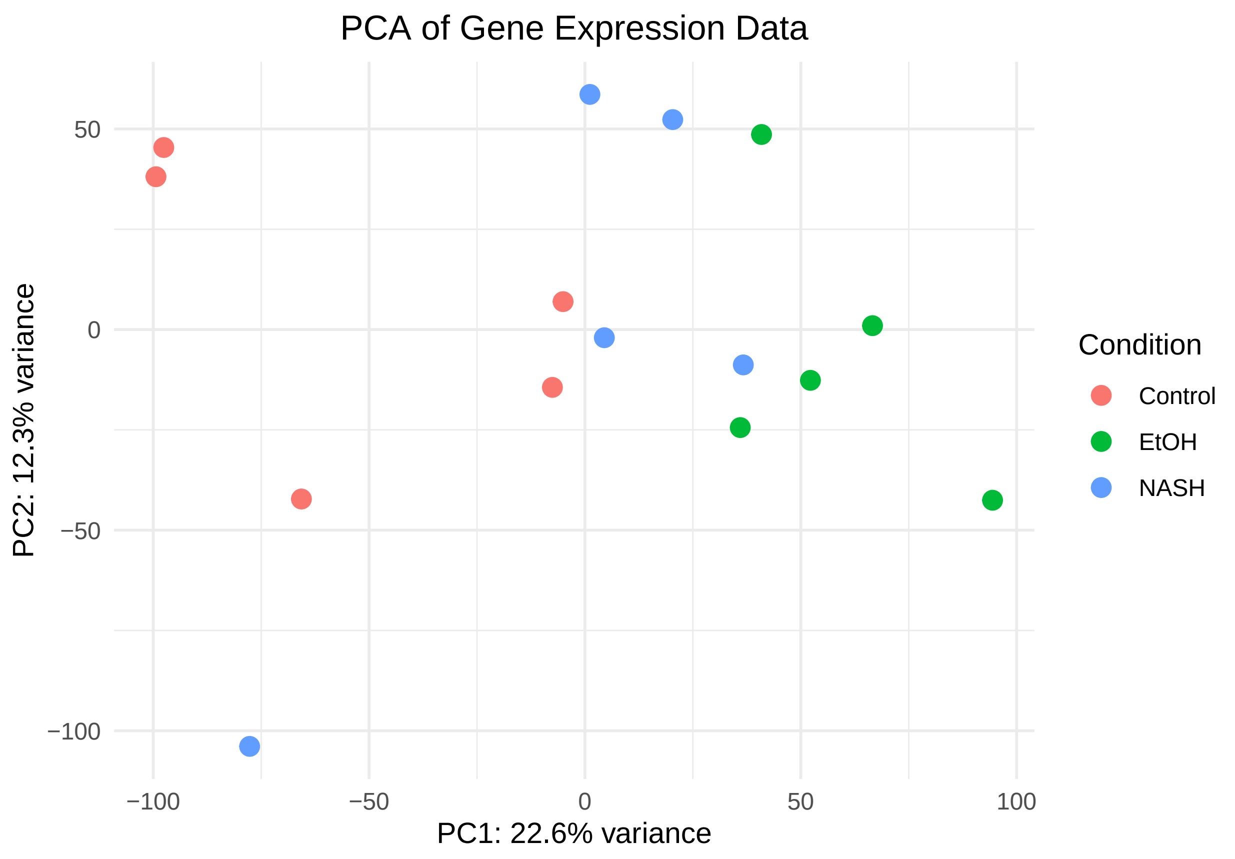
```r
pdf("DispersionPlot.pdf")
plotDispEsts(dds_filtered)
dev.off()
```

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

PCA Plot

```r
# Perform variance-stabilizing transformation (VST) on dds_filtered
vsd <- vst(dds_filtered, blind = FALSE)  # or use rlog for rlog-transformed data

# Extract the transformed expression data
pca_data <- assay(vsd)  # This will be used for PCA


# Perform PCA on the transposed matrix (samples as rows)
pca <- prcomp(t(pca_data), scale. = TRUE)

# Extract the sample metadata (conditions)
sample_conditions <- colData(dds_filtered)$condition  # Assuming this contains 'Control', 'EtOH', 'NASH

# Create a data frame for PCA plotting
pca_df <- data.frame(PC1 = pca$x[,1], PC2 = pca$x[,2], Condition = sample_conditions)

# Load ggplot2 for plotting
```

```r
library(ggplot2)

# Plot PCA with points colored by Condition (Control, EtOH, NASH)
pca_plot <- ggplot(pca_df, aes(x = PC1, y = PC2, color = Condition)) +
    geom_point(size = 3) +
    theme_minimal() +
    labs(title = "PCA of Gene Expression Data",
         x = paste0("PC1: ", round(100 * summary(pca)$importance[2,1], 1), "% variance"),
         y = paste0("PC2: ", round(100 * summary(pca)$importance[2,2], 1), "% variance")) +
    theme(plot.title = element_text(hjust = 0.5))

# Display the plot
print(pca_plot)
```



```r
# Save the plot as a PDF
ggsave("PCA_plot_Control_EtOH_NASH.pdf", plot = pca_plot, width = 8, height = 6)
```

Obtaining the Results

Extract Significant Results for EtOH

We are extracting the significantly differentially expressed genes for EtOH vs Control (adjusted p-value < 0.05).

```r
#Extract significant results:
```

```r
sigs_etoh <- na.omit(results_etoh)
sigs_etoh <- sigs_etoh[sigs_etoh$padj < 0.05, ]


# Convert sigs_control_etoh (DESeqResults object) to a data frame
sigs_etoh_df <- as.data.frame(sigs_etoh)

# Load biomaRt package
library(biomaRt)

# Use the ENSEMBL database to retrieve gene symbols
mart <- useEnsembl(biomart = "ensembl", dataset = "hsapiens_gene_ensembl", mirror = "us")

# Get the gene symbols for the ENSEMBL IDs in sigs_control_etoh_df
genes <- rownames(sigs_etoh_df)
gene_info <- getBM(filters = "ensembl_gene_id",
                   attributes = c("ensembl_gene_id", "hgnc_symbol"),
                   values = genes,
                   mart = mart)

# Add the ensembl_gene_id as a new column in sigs_control_nash_df
sigs_etoh_df$ensembl_gene_id <- rownames(sigs_etoh_df)

# Merge the gene symbols into sigs_control_nash_df
sigs_etoh_annotated <- merge(sigs_etoh_df, gene_info, by = "ensembl_gene_id", all.x = TRUE)

# Reorder columns to make gene symbols appear first if desired
sigs_etoh_annotated <- sigs_etoh_annotated[, c("hgnc_symbol", "ensembl_gene_id", colnames(sigs_etoh_ann


# Save the annotated results as an Excel file
library(writexl)

write_xlsx(sigs_etoh_annotated, "sigs_etoh_with_gene_symbols.xlsx")
```

```r
# Install readxl if you haven't already
if (!requireNamespace("readxl", quietly = TRUE)) {
  install.packages("readxl")
}

library(readxl)

# Read your Excel file
sigs_e <- "sigs_etoh_with_gene_symbols.xlsx"
Sigs_e_1 <- read_excel(sigs_e)

# Create a preview
head(Sigs_e_1, 10)
```

```
## # A tibble: 10 x 8
##    hgnc_symbol ensembl_gene_id baseMean log2FoldChange lfcSE  stat   pvalue
##    <chr>       <chr>              <dbl>          <dbl> <dbl> <dbl>    <dbl>
##  1 GCLC        ENSG00000001084   1109.          -0.755 0.247 -3.06 0.00223
##  2 CFTR        ENSG00000001626    371.           2.56  0.886  2.89 0.00384
##  3 POLDIP2     ENSG00000004142   2088.          -0.777 0.221 -3.52 0.000427
##  4 SLC25A13    ENSG00000004864   1477.          -0.889 0.282 -3.15 0.00161
##  5 ST7         ENSG00000004866    274.          -0.849 0.259 -3.28 0.00105
##  6 SLC25A5     ENSG00000005022   2528.          -0.710 0.247 -2.87 0.00412
##  7 WDR54       ENSG00000005448     36.1          1.28  0.432  2.96 0.00310
##  8 CROT        ENSG00000005469    388.          -0.928 0.315 -2.95 0.00321
##  9 GDE1        ENSG00000006007    313.          -0.850 0.227 -3.75 0.000176
## 10 TMEM132A    ENSG00000006118     58.1          2.35  0.690  3.40 0.000676
## # i 1 more variable: padj <dbl>
```

Extract Significant Results for NASH

We are extracting the significantly differentially expressed genes for NASH vs Control (adjusted p-value < 0.05).

```r
#Extract significant results:

sigs_nash <- na.omit(results_nash)
sigs_nash <- sigs_nash[sigs_nash$padj < 0.05, ]


# Convert sigs_control_nash (DESeqResults object) to a data frame
sigs_nash_df <- as.data.frame(sigs_nash)

# Load biomaRt package
library(biomaRt)

# Use the ENSEMBL database to retrieve gene symbols
mart <- useEnsembl(biomart = "ensembl", dataset = "hsapiens_gene_ensembl", mirror = "us")

# Get the gene symbols for the ENSEMBL IDs in sigs_control_nash_df
genes <- rownames(sigs_nash_df)
gene_info <- getBM(filters = "ensembl_gene_id",
                   attributes = c("ensembl_gene_id", "hgnc_symbol"),
                   values = genes,
                   mart = mart)

# Add the ensembl_gene_id as a new column in sigs_control_nash_df
sigs_nash_df$ensembl_gene_id <- rownames(sigs_nash_df)

# Merge the gene symbols into sigs_control_nash_df
sigs_nash_annotated <- merge(sigs_nash_df, gene_info, by = "ensembl_gene_id", all.x = TRUE)

# Reorder columns to make gene symbols appear first if desired
sigs_nash_annotated <- sigs_nash_annotated[, c("hgnc_symbol", "ensembl_gene_id", colnames(sigs_nash_anno


# Save the annotated results as an Excel file
library(writexl)
write_xlsx(sigs_nash_annotated, "sigs_nash_with_gene_symbols.xlsx")
```

```r
# Install readxl if you haven't already
if (!requireNamespace("readxl", quietly = TRUE)) {
  install.packages("readxl")
}
```

```r
library(readxl)

# Read your Excel file
sigs_n <- "sigs_nash_with_gene_symbols.xlsx"
Sigs_n_1 <- read_excel(sigs_n)

# Create a preview
head(Sigs_n_1, 10)
```

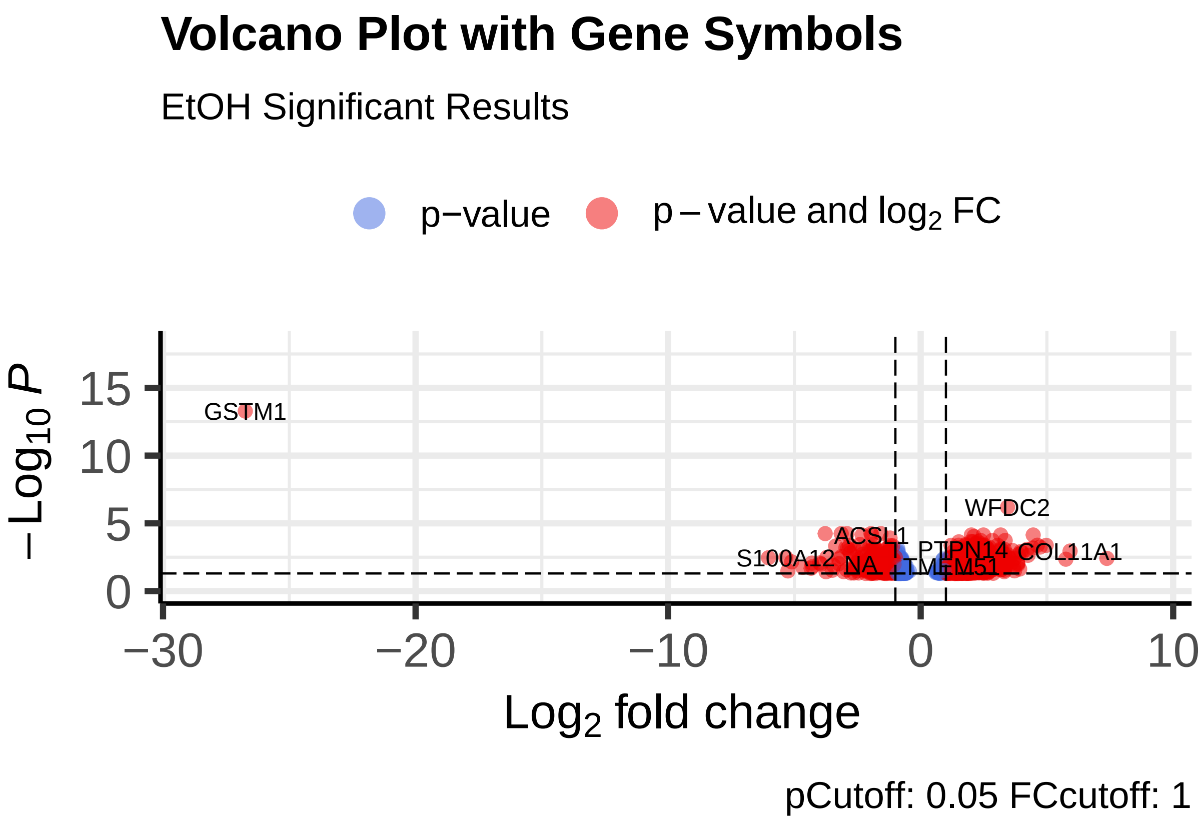
```
## # A tibble: 10 x 8
##    hgnc_symbol ensembl_gene_id baseMean log2FoldChange lfcSE  stat      pvalue
##    <chr>       <chr>              <dbl>          <dbl> <dbl> <dbl>       <dbl>
##  1 ME1         ENSG00000065833    169.          -1.75  0.354 -4.95 0.000000729
##  2 TRO         ENSG00000067445    138.           1.53  0.292  5.24 0.000000161
##  3 DERL2       ENSG00000072849    416.          -0.986 0.250 -3.95 0.0000780
##  4 SLCO1A2     ENSG00000084453     99.7          3.12  0.662  4.72 0.00000237
##  5 AURKA       ENSG00000087586     59.0         -2.56  0.621 -4.12 0.0000380
##  6 CAPN3       ENSG00000092529     50.5          1.81  0.442  4.09 0.0000428
##  7 NANS        ENSG00000095380    248.          -1.17  0.296 -3.96 0.0000763
##  8 TRIM9       ENSG00000100505     31.5         -3.81  0.955 -3.99 0.0000669
##  9 SRP54       ENSG00000100883    719.          -0.977 0.227 -4.30 0.0000171
## 10 MEFV        ENSG00000103313     29.4         -2.68  0.672 -3.99 0.0000670
## # i 1 more variable: padj <dbl>
```

Visualise the Results

Final Volcano Plot for EtOH with Gene Symbols

We are generating a Volcano plot using gene symbols for labeling the significant results in EtOH vs Control.

```r
#Map ENSEMBL IDs to gene symbols and generate the Volcano plot:

sigs.df_etoh <- as.data.frame(sigs_etoh)

sigs.df_etoh$symbol <- mapIds(org.Hs.eg.db,
                         keys = rownames(sigs.df_etoh),
                         keytype = "ENSEMBL",
                         column = "SYMBOL",
                         multiVals = function(x) paste(x, collapse = ";"))


# Define the cutoffs
pCutoffValue <- 0.05
FCcutoffValue <- 1.0


EnhancedVolcano(sigs.df_etoh,
                lab = sigs.df_etoh$symbol,
                x = 'log2FoldChange',
                y = 'padj',
                pCutoff = pCutoffValue,
                FCcutoff = FCcutoffValue,
                pointSize = 2.0,
                labSize = 3.0,
                title = 'Volcano Plot with Gene Symbols',
                subtitle = 'EtOH Significant Results',
                caption = paste('pCutoff:', pCutoffValue, 'FCcutoff:', FCcutoffValue))
```



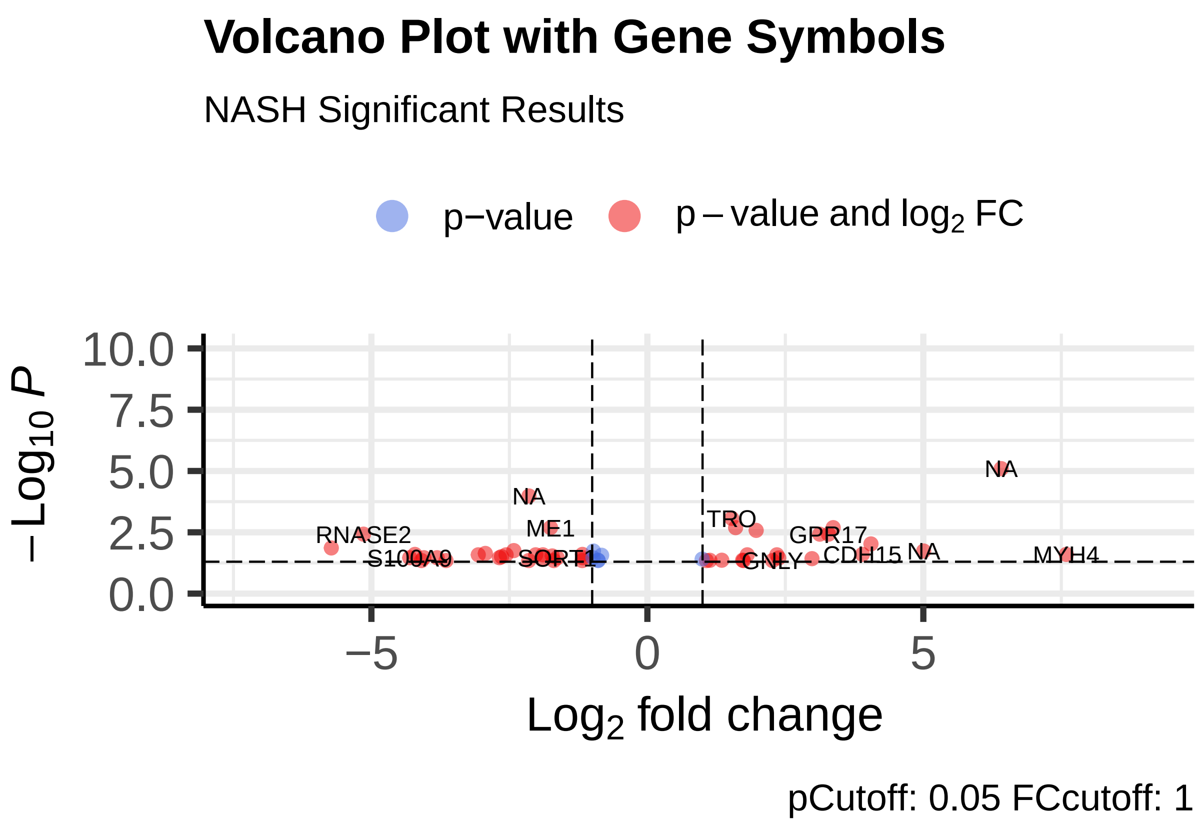
```
# Save the EnhancedVolcano plot as a PDF
pdf("EtOH_EnhancedVolcanoPlot_with_GeneSymbols.pdf")
EnhancedVolcano(sigs.df_etoh,
                lab = sigs.df_etoh$symbol,  # Use the gene symbols for labeling
                x = 'log2FoldChange',
                y = 'padj',
                pCutoff = pCutoffValue,
                FCcutoff = FCcutoffValue,
                pointSize = 2.0,
                labSize = 3.0,
                title = 'Volcano Plot with Gene Symbols',
                subtitle = 'EtOH Significant Results',
                caption = paste('pCutoff:', pCutoffValue, 'FCcutoff:', FCcutoffValue))
dev.off()
```

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

Final Volcano Plot for NASH with Gene Symbols

We are generating a Volcano plot using gene symbols for labeling the significant results in NASH vs Control.

```r
# Map ENSEMBL IDs to gene symbols and generate the Volcano plot:

sigs.df_nash <- as.data.frame(sigs_nash)

sigs.df_nash$symbol <- mapIds(org.Hs.eg.db,
                             keys = rownames(sigs.df_nash),
                             keytype = "ENSEMBL",
                             column = "SYMBOL",
                             multiVals = function(x) paste(x, collapse = ";"))


# Define the cutoffs
pCutoffValue <- 0.05
FCcutoffValue <- 1.0

EnhancedVolcano(sigs.df_nash,
                lab = sigs.df_nash$symbol,
                x = 'log2FoldChange',
                y = 'padj',
                pCutoff = pCutoffValue,
                FCcutoff = FCcutoffValue,
                pointSize = 2.0,
                labSize = 3.0,
                title = 'Volcano Plot with Gene Symbols',
                subtitle = 'NASH Significant Results',
                caption = paste('pCutoff:', pCutoffValue, 'FCcutoff:', FCcutoffValue))
```



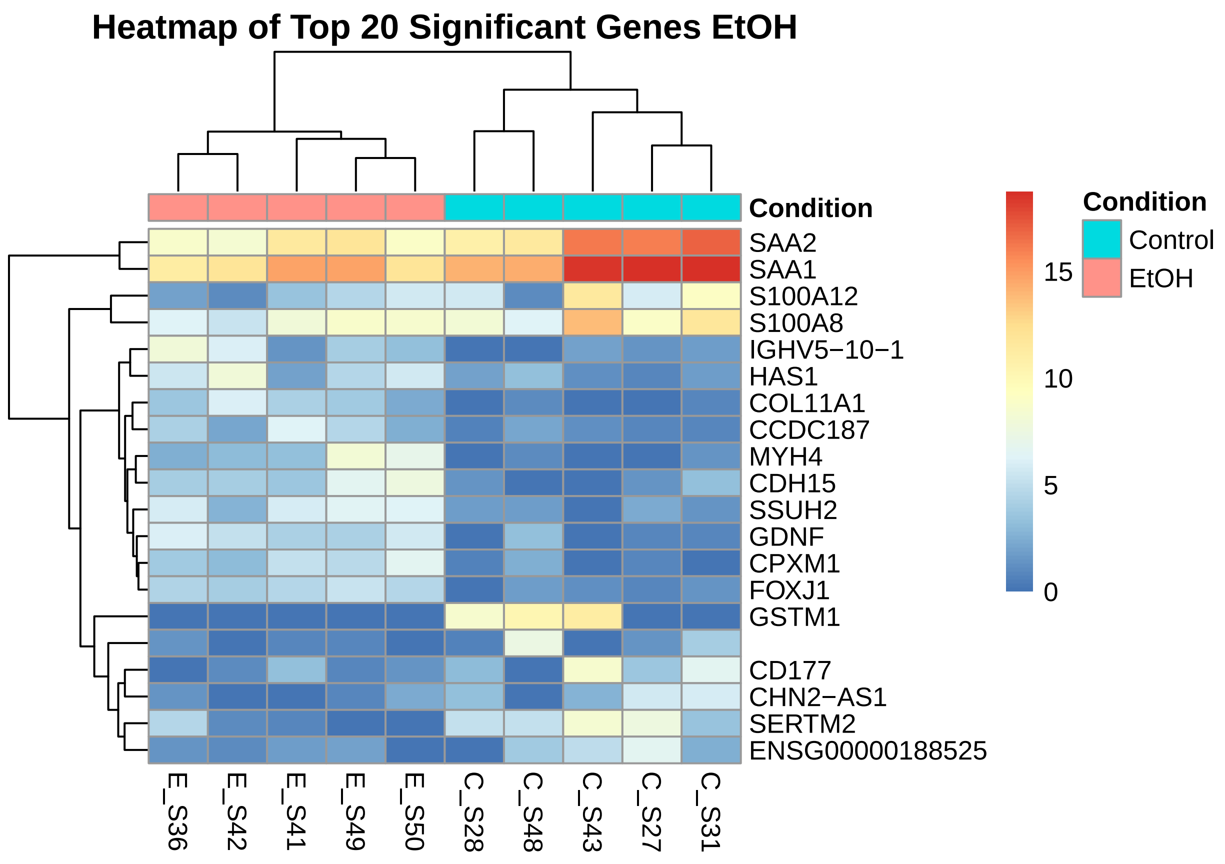
```r
# Save the EnhancedVolcano plot as a PDF
pdf("NASH_EnhancedVolcanoPlot_with_GeneSymbols.pdf")
EnhancedVolcano(sigs.df_nash,
                lab = sigs.df_nash$symbol,  # Use the gene symbols for labeling
                x = 'log2FoldChange',
                y = 'padj',
                pCutoff = pCutoffValue,
                FCcutoff = FCcutoffValue,
                pointSize = 2.0,
                labSize = 3.0,
                title = 'Volcano Plot with Gene Symbols',
                subtitle = 'NASH Significant Results',
                caption = paste('pCutoff:', pCutoffValue, 'FCcutoff:', FCcutoffValue))
dev.off()
```

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

Top Genes Heatmap Control_EtOH

```r
# Assuming you already have `sigs_control_etoh` with gene symbols loaded
# Load the DESeq2 object that contains the count data (dds_filtered object)
# Filter the DESeq2 object to only keep Control and EtOH samples
dds_filtered_control_etoh <- dds_filtered[, colData(dds_filtered)$condition %in% c("Control", "EtOH")]

# Get the normalized counts for the significant genes from Control vs. EtOH
# Extract only the genes that are significant and present in your filtered data
significant_genes_e <- rownames(sigs_etoh)  # Assuming rownames are gene symbols or ENSEMBL IDs
normalized_counts <- counts(dds_filtered_control_etoh, normalized = TRUE)
sig_gene_counts_e <- normalized_counts[significant_genes_e, ]


# Step 1: Sort the significant genes from EtOH by log2FoldChange and select the top 20

# Select top 10 upregulated genes (highest log2FoldChange)
top_up_e <- sigs_etoh_df[order(-sigs_etoh_df$log2FoldChange), ][1:10, ]

# Select top 10 downregulated genes (lowest log2FoldChange)
top_down_e <- sigs_etoh_df[order(sigs_etoh_df$log2FoldChange), ][1:10, ]

# Combine top up- and downregulated genes
top_genes_e <- rbind(top_up_e, top_down_e)

# Step 2: Get the normalized counts for these top genes
top_gene_ensembl_ids_e <- rownames(top_genes_e)  # These are ENSEMBL IDs
top_gene_counts_e <- normalized_counts[top_gene_ensembl_ids_e, ]

# Step 3: Use biomaRt to convert ENSEMBL IDs to gene symbols for EtOH
# Connect to the Ensembl database

mart <- useEnsembl(biomart = "ensembl", dataset = "hsapiens_gene_ensembl", mirror = "us")
```

```r
# Query to get gene symbols for the given ENSEMBL IDs
gene_mapping_e <- getBM(
    filters = "ensembl_gene_id",
    attributes = c("ensembl_gene_id", "hgnc_symbol"),
    values = top_gene_ensembl_ids_e,
    mart = mart
)

# Merge the gene symbols with your top gene data
# Ensure the order matches the original ENSEMBL IDs
top_gene_symbols_e <- gene_mapping_e$hgnc_symbol[match(top_gene_ensembl_ids_e, gene_mapping_e$ensembl_g

# Replace any missing symbols with the ENSEMBL IDs (in case symbols are missing)
top_gene_symbols_e[is.na(top_gene_symbols_e)] <- top_gene_ensembl_ids_e[is.na(top_gene_symbols_e)]

# Optional: Log transform the counts to reduce the dynamic range
log_top_counts_e <- log2(top_gene_counts_e + 1)

# Step 4: Ensure the annotation_col is a data frame with proper column names (for Control vs. EtOH)
annotation_col_e <- data.frame(Condition = colData(dds_filtered)$condition)
rownames(annotation_col_e) <- colnames(dds_filtered)

# Step 5: Save the heatmap as a PDF with gene symbols as row names for EtOH
pdf("heatmap_etoh_top_genes.pdf", width = 10, height = 8)

# Generate the heatmap with gene symbols as row names
pheatmap(log_top_counts_e,
         cluster_rows = TRUE,
         cluster_cols = TRUE,
         show_rownames = TRUE,    # Show row names for the top genes (gene symbols)
         labels_row = top_gene_symbols_e,  # Use gene symbols as row labels
         show_colnames = TRUE,    # Keep column names
         annotation_col = annotation_col_e,   # Correctly formatted annotation data
         main = "Heatmap of Top 20 Significant Genes EtOH")
```



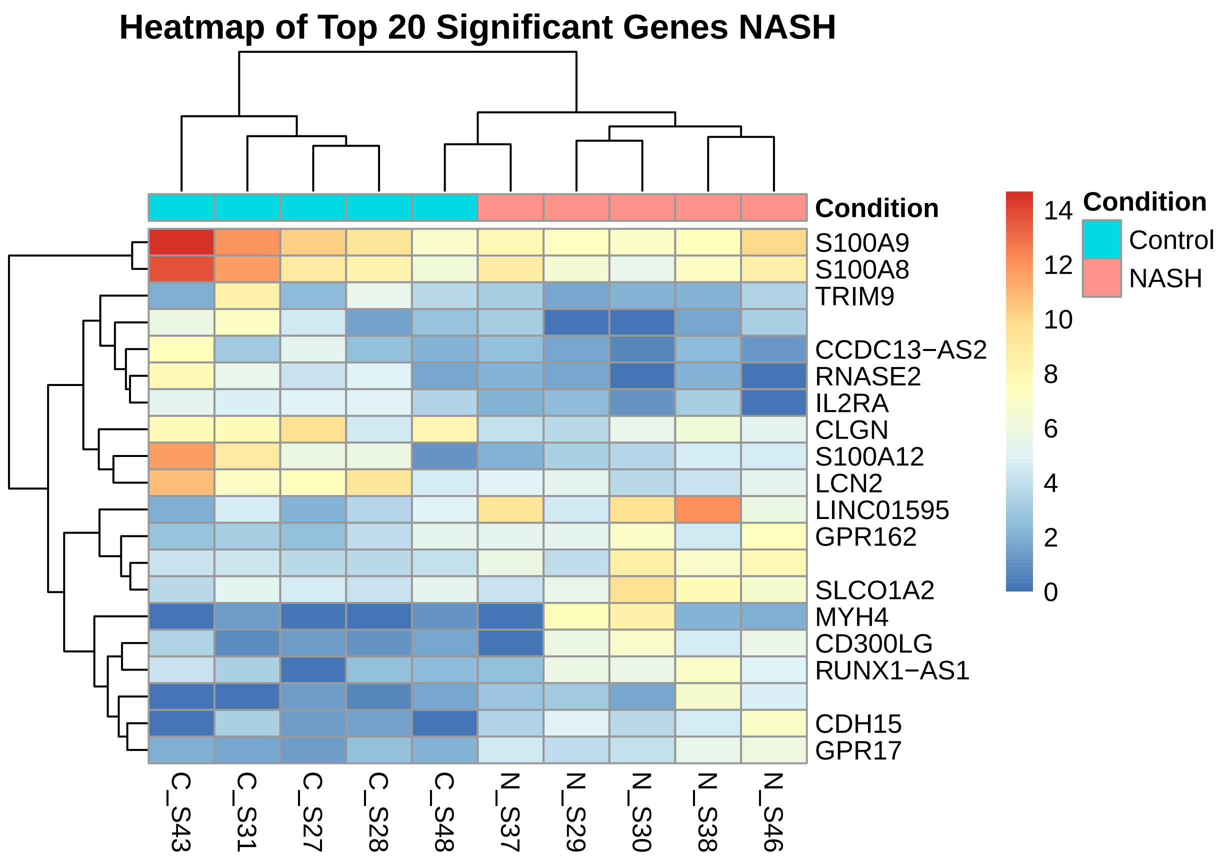
```r
dev.off()  # Close the PDF device
```

```
## pdf
##   3
```

Top Genes Heatmap Control_NASH

```r
# Assuming you already have `sigs_control_nash` with gene symbols loaded
# Load the DESeq2 object that contains the count data (dds_filtered object)
# Filter the DESeq2 object to only keep Control and NASH samples
dds_filtered_nash <- dds_filtered[, colData(dds_filtered)$condition %in% c("Control", "NASH")]

# Get the normalized counts for the significant genes from Control vs. NASH
# Extract only the genes that are significant and present in your filtered data
significant_genes <- rownames(sigs_nash)  # Assuming rownames are gene symbols or ENSEMBL IDs
normalized_counts <- counts(dds_filtered_nash, normalized = TRUE)
sig_gene_counts <- normalized_counts[significant_genes, ]


# Step 1: Sort the significant genes from NASH by log2FoldChange and select the top 20

# Select top 10 upregulated genes (highest log2FoldChange)
top_up_n <- sigs_nash_df[order(-sigs_nash_df$log2FoldChange), ][1:10, ]

# Select top 10 downregulated genes (lowest log2FoldChange)
top_down_n <- sigs_nash_df[order(sigs_nash_df$log2FoldChange), ][1:10, ]
```

```r
# Combine top up- and downregulated genes
top_genes_nash <- rbind(top_up_n, top_down_n)


# Step 2: Get the normalized counts for these top genes
top_gene_ensembl_ids_nash <- rownames(top_genes_nash)  # These are ENSEMBL IDs
top_gene_counts_nash <- normalized_counts[top_gene_ensembl_ids_nash, ]

# Step 3: Use biomaRt to convert ENSEMBL IDs to gene symbols for NASH
# Connect to the Ensembl database
mart <- useEnsembl(biomart = "ensembl", dataset = "hsapiens_gene_ensembl", mirror = "us")

# Query to get gene symbols for the given ENSEMBL IDs
gene_mapping_nash <- getBM(
    filters = "ensembl_gene_id",
    attributes = c("ensembl_gene_id", "hgnc_symbol"),
    values = top_gene_ensembl_ids_nash,
    mart = mart
)

# Merge the gene symbols with your top gene data
# Ensure the order matches the original ENSEMBL IDs
top_gene_symbols_nash <- gene_mapping_nash$hgnc_symbol[match(top_gene_ensembl_ids_nash, gene_mapping_nas

# Replace any missing symbols with the ENSEMBL IDs (in case symbols are missing)
top_gene_symbols_nash[is.na(top_gene_symbols_nash)] <- top_gene_ensembl_ids_nash[is.na(top_gene_symbols_

# Optional: Log transform the counts to reduce the dynamic range
log_top_counts_nash <- log2(top_gene_counts_nash + 1)

# Step 4: Ensure the annotation_col is a data frame with proper column names (for Control vs. NASH)
annotation_col_nash <- data.frame(Condition = colData(dds_filtered)$condition)
rownames(annotation_col_nash) <- colnames(dds_filtered)

# Step 5: Save the heatmap as a PDF with gene symbols as row names for NASH
pdf("heatmap_nash_top_genes.pdf", width = 10, height = 8)

# Generate the heatmap with gene symbols as row names
pheatmap(log_top_counts_nash,
         cluster_rows = TRUE,
         cluster_cols = TRUE,
         show_rownames = TRUE,    # Show row names for the top genes (gene symbols)
         labels_row = top_gene_symbols_nash,  # Use gene symbols as row labels
         show_colnames = TRUE,    # Keep column names
         annotation_col = annotation_col_nash,   # Correctly formatted annotation data
         main = "Heatmap of Top 20 Significant Genes NASH")
```



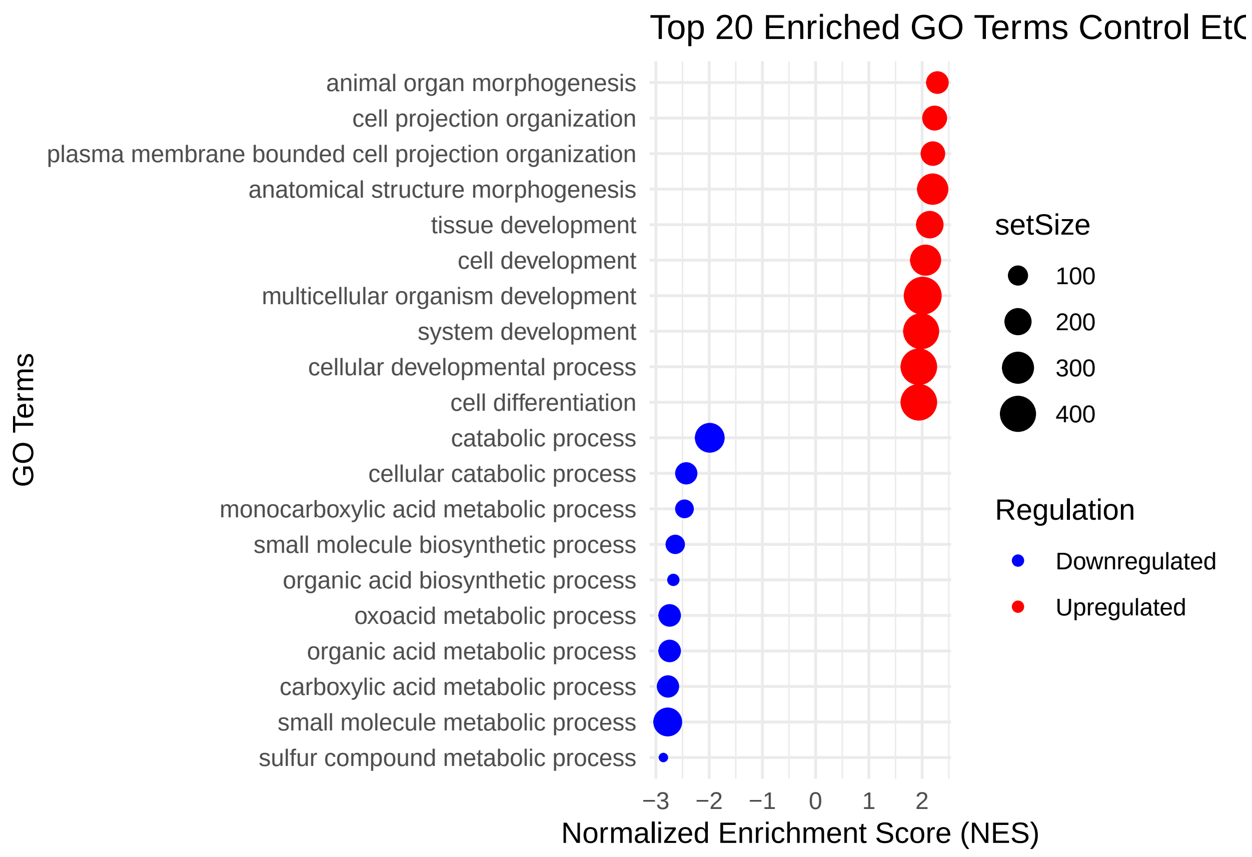
```r
dev.off()  # Close the PDF device
```

```
## pdf
##   3
```

GSEA for Control_EtOH

```r
# Create a ranked gene list based on log2FoldChange
# Remove any NA values and sort by log2FoldChange
gene_list <- sigs_etoh_annotated$log2FoldChange
names(gene_list) <- sigs_etoh_annotated$ensembl_gene_id  # Use gene symbols or ENSEMBL IDs

# Remove NA values and sort the list in decreasing order (for GSEA)
gene_list <- sort(na.omit(gene_list), decreasing = TRUE)

# Load necessary packages
library(clusterProfiler)
library(org.Hs.eg.db)  # Load human gene annotation

# Perform GSEA using GO terms (Biological Process, Molecular Function, and Cellular Component)
gsea_results <- gseGO(geneList = gene_list,
                      OrgDb = org.Hs.eg.db,     # Use human gene annotation
                      ont = "BP",              # Perform GSEA for all GO categories (BP, MF, CC)
                      pvalueCutoff = 0.05,     # p-value cutoff for enrichment
                      keyType = "ENSEMBL",     # Use ENSEMBL IDs for genes
                      minGSSize = 15,          # Minimum size of gene sets
```

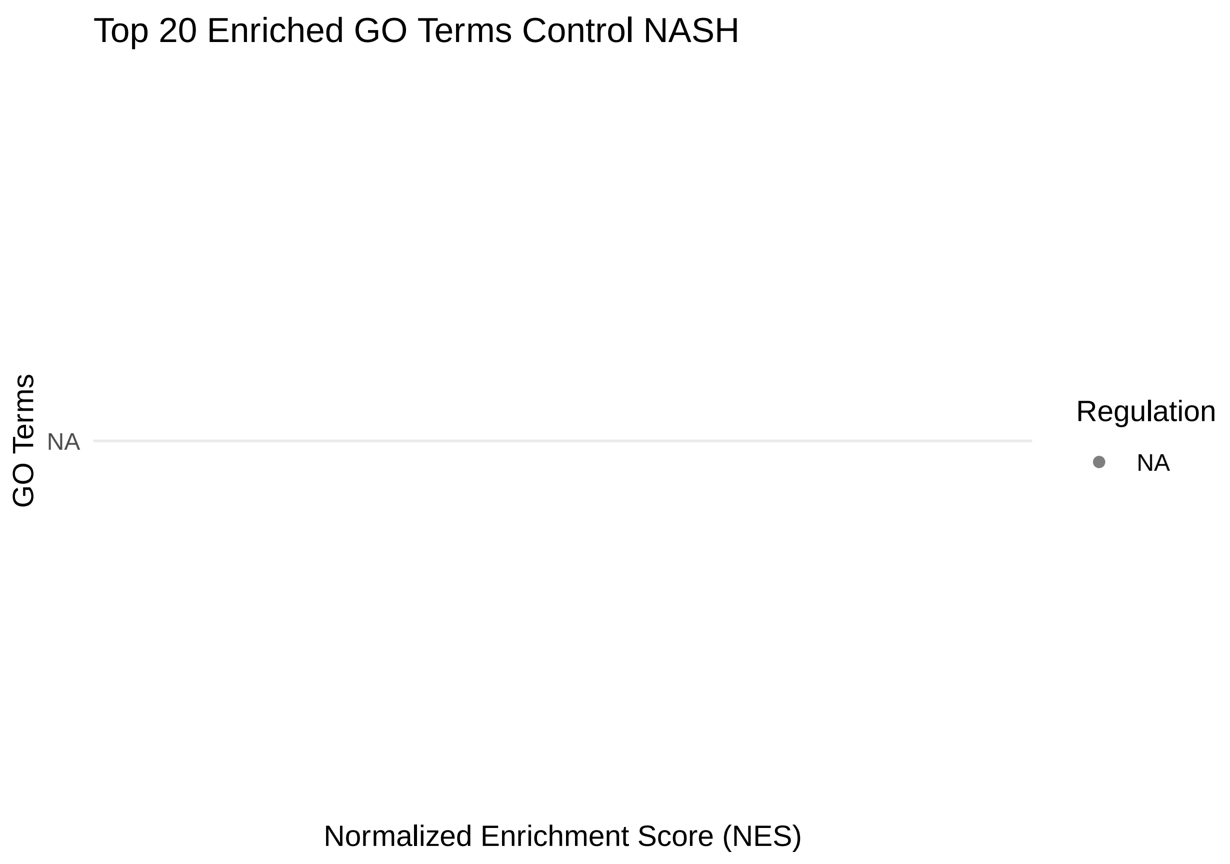
```
                                   maxGSSize = 500,          # Maximum size of gene sets
                                   eps = 0)


# Convert GSEA results to a data frame
gsea_df <- as.data.frame(gsea_results)

# Add a Regulation column indicating up- or down-regulation
gsea_df$Regulation <- ifelse(gsea_df$NES > 0, "Upregulated", "Downregulated")


# Modify the dot plot to color by Regulation (Upregulated or Downregulated)
go_gsea_plot <- ggplot(gsea_df[1:20, ], aes(x = NES, y = reorder(Description, NES), color = Regulation,
  geom_point() +
  scale_color_manual(values = c("Upregulated" = "red", "Downregulated" = "blue")) +
  labs(title = "Top 20 Enriched GO Terms Control EtOH",
       x = "Normalized Enrichment Score (NES)",
       y = "GO Terms") +
  theme_minimal()

# Print the plot
print(go_gsea_plot)
```



```
# Save the plot as a PDF file
ggsave("go_gsea_plot_etoh.pdf", plot = go_gsea_plot, width = 10, height = 7)
```

GSEA for Control_NASH

```r
# Create a ranked gene list based on log2FoldChange
# Remove any NA values and sort by log2FoldChange
gene_list <- sigs_nash_annotated$log2FoldChange
names(gene_list) <- sigs_nash_annotated$ensembl_gene_id  # Use gene symbols or ENSEMBL IDs

# Remove NA values and sort the list in decreasing order (for GSEA)
gene_list <- sort(na.omit(gene_list), decreasing = TRUE)

# Load necessary packages
library(clusterProfiler)
library(org.Hs.eg.db)  # Load human gene annotation

# Perform GSEA using GO terms (Biological Process, Molecular Function, and Cellular Component)
gsea_results <- gseGO(geneList = gene_list,
                        OrgDb = org.Hs.eg.db,    # Use human gene annotation
                        ont = "BP",             # Perform GSEA for all GO categories (BP, MF, CC)
                        pvalueCutoff = 0.05,     # p-value cutoff for enrichment
                        keyType = "ENSEMBL",     # Use ENSEMBL IDs for genes
                        minGSSize = 15,          # Minimum size of gene sets
                        maxGSSize = 500,         # Maximum size of gene sets
                        eps = 0)


# Convert GSEA results to a data frame
gsea_df <- as.data.frame(gsea_results)

# Add a Regulation column indicating up- or down-regulation
gsea_df$Regulation <- ifelse(gsea_df$NES > 0, "Upregulated", "Downregulated")


# Modify the dot plot to color by Regulation (Upregulated or Downregulated)
go_gsea_plot_nash <- ggplot(gsea_df[1:20, ], aes(x = NES, y = reorder(Description, NES), color = Regula
  geom_point() +
  scale_color_manual(values = c("Upregulated" = "red", "Downregulated" = "blue")) +
  labs(title = "Top 20 Enriched GO Terms Control NASH",
       x = "Normalized Enrichment Score (NES)",
       y = "GO Terms") +
  theme_minimal()

# Print the plot
print(go_gsea_plot_nash)
```



```r
# Save the plot as a PDF file
ggsave("go_gsea_plot_control_nash.pdf", plot = go_gsea_plot_nash, width = 10, height = 7)
```