"RNAseq Multiple Volcano Plots Comparison"

"Mohammad Reza Mohajeri"

2023-09-16

- This Rmd file provides a collection of scripts to generate volcano plots for visualizing RNA-Seq analysis results.
- Here, we employ three distinct methods for creating these plots to offer various perspectives and customizability.

1. ggplot2 Volcano Plot (Method 1): Leveraging the powerful ggplot2 package, this section offers a detailed and customizable volcano plot.

2. ggplot2 Volcano Plot (Method 2): An alternative rendition using ggplot2, providing a different visualization approach.

3. EnhancedVolcano Plot: Making use of the EnhancedVolcano package in R, this section is tailored for generating informative volcano plots in a genomics context.

- Prior to creating the plots, we undergo a comprehensive "PREPARATION AND ANALYSIS OF RNA-SEQ DATA" phase to ensure the data is well-prepared for visualization.

1. PREPARATION AND ANALYSIS OF RNA-SEQ DATA

This section focuses on importing, preparing, and analyzing RNA-seq data to detect differentially expressed genes.

```
require(readr)
require(tidyr)
require(gridExtra)
require(reshape2)
require(viridis)
require(ggplot2)
require(DESeq2)
require(biomaRt)
require(genefilter)
require(org.Hs.eg.db)
```

```r
  require(ComplexHeatmap)
  require(clusterProfiler)
  require(readr)
  require(knitr)
  require(genefilter)
  require(ggtree)
# if (!requireNamespace("BiocManager", quietly = TRUE))
#     install.packages("BiocManager")
#
# BiocManager::install("ggtree", dependencies = TRUE)

# ============================================
# Importing_Counts_Matrix
# ============================================
# Set the random seed for reproducible results
set.seed(1234)

# Reading and preparing files:
counts <- data.frame(read_tsv("countMatrix.tsv"))

# Set the row names of the 'counts' data frame to be the first column
# (usually gene IDs or similar identifiers)
rownames(counts) <- counts[,1]

# Remove the first column since its information is now in row names
counts <- counts[,-1]

# Round the values in the 'counts' data frame to the nearest integer
# This may be important for some statistical analyses that require integer count data
counts <- round(counts)

# Check the number of missing values (NAs) in the 'counts' data frame
# 'which' returns the positions of NAs, and 'length' counts them
length(which(is.na(counts)))
```

```
## [1] 0
```

```r
# ============================================
# Importing_Metadata
# ============================================
# Read metadata from a tab-separated file "metadata.tsv" into a data frame
metadata <- read_tsv("metadata.tsv")

# Create a new column 'groups' in the metadata data frame.
metadata$groups <- ifelse(
  metadata$Response == "Progressive Disease" |
```

```
    metadata$Response ==  "Stable Disease",
                          "NonResponder", "Responder")

# Sort the rows of the metadata data frame by 'Sample.name' in ascending order
metadata <- metadata[order(metadata$Sample.name), ]

# Convert the 'groups' column to a factor data type,
# useful for statistical modeling:
metadata$groups <- factor(metadata$groups)

# ==========================================
# replace_dots_in_colnames_for_merge
# ==========================================
colnames(counts) <- gsub("\\.", "-", colnames(counts))

# ==========================================
# order_counts_columns_alphabetically
# ==========================================
# Sort the columns of the 'counts' data frame based on their names
counts <- counts[, order(colnames(counts))]

# ==========================================
# Replacement-of-variance_filtering_and_comparison
# ==========================================
# Make a copy of the original 'counts' matrix
original_counts <- counts

# Apply the varFilter to the original_counts and store in a new variable
filtered_counts <- varFilter(as.matrix(original_counts))

# Assigning the filtered data to the 'counts' variable for further analysis
counts <- filtered_counts

# ==========================================
# EnsemblID_to_GeneSymbol_Mapping
# ==========================================

# Connect to Ensembl to fetch gene information
ensembl <- useMart(biomart = "ensembl",
                   dataset = "hsapiens_gene_ensembl",
                   host = "https://www.ensembl.org")

# Retrieve gene symbols corresponding to the Ensembl IDs
symbols <- getBM(attributes = c("ensembl_gene_id", "external_gene_name"),
                 filters = "ensembl_gene_id",
                 values = rownames(counts),
```

```
                  mart = ensembl)

mapping_dict <- setNames(symbols$external_gene_name,
                         symbols$ensembl_gene_id)

# ==========================================
# Differential Expression Analysis
# ==========================================
# Prepare the data set for DESeq2 analysis
deseq <- DESeqDataSetFromMatrix(countData = counts,
                                colData = metadata,
                                design = ~ groups)

deseq <- deseq[rowSums(counts(deseq)) >= 10, ]

deseq <- DESeq(deseq)

deseq_result <- function(deseq,
                         FC,
                         p_adj,
                         mapping,
                         ntop) {

  dir.create("DESeq2", showWarnings = F)

  res <- results(deseq)
  res <- data.frame(res)

  if(mapping) {
    res$Symbol <- mapping_dict[rownames(res)]
    res <- res[which(res$Symbol != ""), c(7, 2, 5:6)]
    rownames(res) <- NULL

  } else {
    res <- cbind(rownames(res), res)
    colnames(res)[1] <- "Symbol"
    res <- res[, c(1,3,6:7)]
  }

  res <- na.omit(res)

  # this up_down
  up_down <- res[which(abs(res$log2FoldChange) > FC & res$padj < p_adj), ]
  up <- up_down[which(up_down$log2FoldChange > FC), ]
  down <- up_down[which(!up_down$log2FoldChange > FC), ]
```

```
  up <- up[order(up$log2FoldChange, decreasing = TRUE),]
  down <- down[order(down$log2FoldChange, decreasing = FALSE),]

  up_down <- rbind(up, down)

  # another version of the above
  message(cat("Up-regulate: ", nrow(up), "|", "Down-rgulated: ", nrow(down)))

  # Prepare the top 10 up and down regulated genes
  top_up = head(up, 10)
  top_down = head(down, 10)

  # Display tables
message("Top 10 Up-regulated genes:")
print(kable(top_up[, c("Symbol", "log2FoldChange", "padj")], digits = 2))

message("Top 10 Down-regulated genes:")
# Using the `kable` function from the `knitr` package to create a well-formatted table
print(kable(top_down[, c("Symbol", "log2FoldChange", "padj")], digits = 2))

  write_csv(up_down, "DESeq2/DESeq2_Sorted_Up_Down_Genes.csv")
  write_csv(res, "DESeq2/DESeq2_Unsorted_All_Genes.csv")

  # return(list(UP = top_up, DOWN = top_down))
  # to return only specific columns (Symbol, log2FoldChange, and padj)
  return(list(UP = top_up[, c("Symbol", "log2FoldChange", "padj")],
              DOWN = top_down[, c("Symbol", "log2FoldChange", "padj")]))

}

results <- deseq_result(deseq,
             FC = 1,
             p_adj = 0.05,
             mapping = TRUE,
             ntop = 10)
```

```
## Up-regulate:  382 | Down-rgulated:  903
##
## |      |Symbol  | log2FoldChange| padj|
## |:-----|:-------|--------------:|----:|
## |17903 |UBD     |           8.20|    0|
## |16033 |UTS2B   |           7.49|    0|
## |16143 |MYBPC1  |           7.05|    0|
## |4307  |CCL20   |           6.88|    0|
## |7645  |PIK3C2G |           6.32|    0|
## |7826  |BCL2A1  |           6.10|    0|
```

```
## |17610 |IGKC    |           5.92|    0|
## |17613 |IGLC3   |           5.89|    0|
## |6484  |JCHAIN  |           5.13|    0|
## |17620 |IGHG1   |           5.06|    0|
##
##
## |      |Symbol    | log2FoldChange| padj|
## |:-----|:---------|--------------:|----:|
## |19168 |PRAMEF36P |         -24.88| 0.00|
## |22124 |TBC1D3    |         -11.32| 0.01|
## |21512 |TBC1D3G   |         -10.72| 0.02|
## |8106  |KLK3      |         -10.18| 0.03|
## |14592 |ZNF716    |         -10.13| 0.03|
## |13915 |MAGEB10   |          -9.97| 0.00|
## |18460 |TRIM49D1  |          -9.97| 0.03|
## |16462 |MAGEA6    |          -9.64| 0.00|
## |11144 |TBXT      |          -9.53| 0.00|
## |15289 |OR2V1     |          -9.46| 0.05|
```

2. COMPARATIVE VISUALIZATION OF RNA-SEQ RESULTS USING VOLCANO PLOTS

1.CUSTOM GGPLOT2 VOLCANO PLOT (METHOD 1)

- A flexible approach leveraging ggplot2 to create a volcano plot from a CSV data file.
- This method classifies genes into categories:

  - Upregulated

  - Downregulated

  - NotSignificant

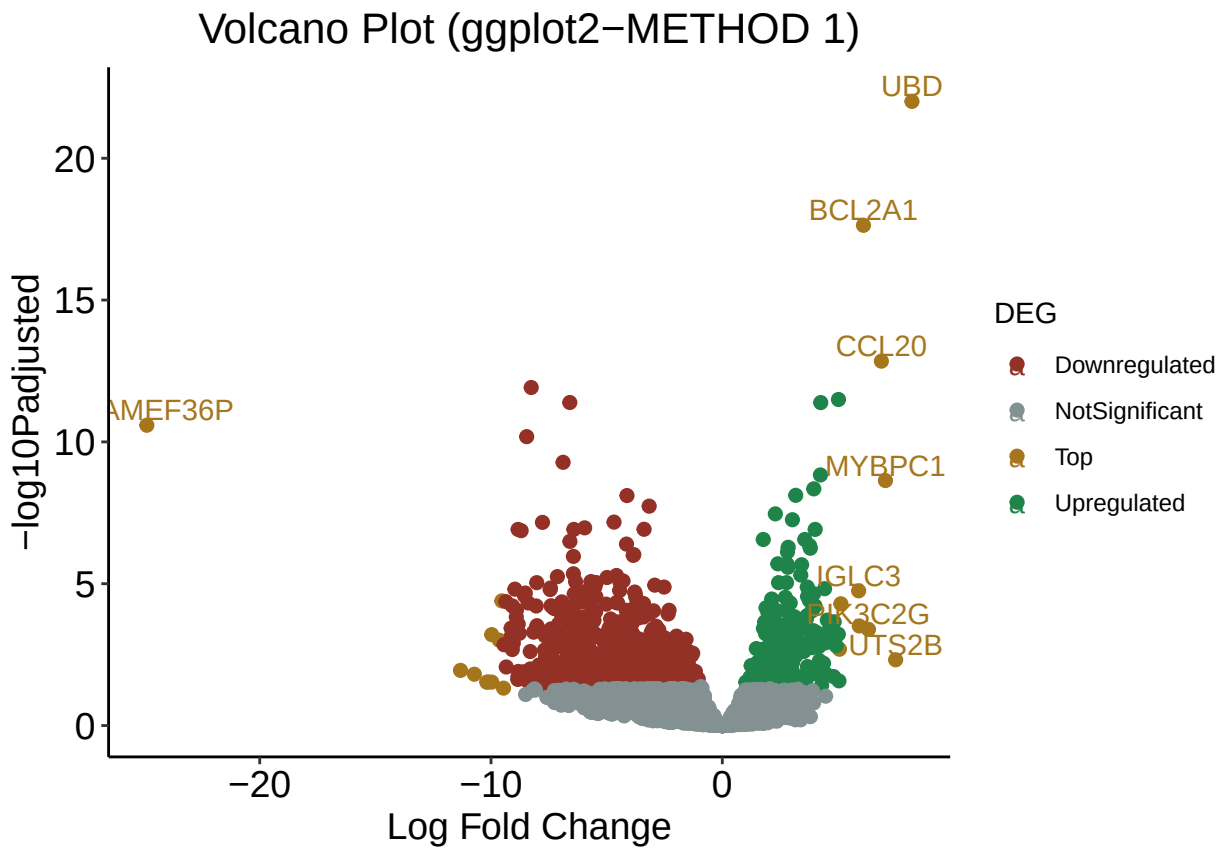
- Visualization is enhanced using a custom color palette.
- The generated plot is saved as a PDF for future reference.

1.CUSTOM GGPLOT2 VOLCANO PLOT (METHOD 1)

```r
# ------------------------------------------------------------
# Section 1: Define the volcano_plot Function
# ------------------------------------------------------------
volcano_plot <- function(padjlevel,
                         Up_FC,
                         Down_FC,
                         ntop) {

# ------------------------------------------------------------
# Section 2: Create Output Directory for Plots
# ------------------------------------------------------------
  dir.create("plots", showWarnings = FALSE)


# ------------------------------------------------------------
# Section 3: Read the Data
# ------------------------------------------------------------
  volcano <- data.frame(read_csv(paste0("DESeq2/DESeq2_Unsorted_All_Genes.csv")))


# ------------------------------------------------------------
# Section 4: Preprocess the Data
# ------------------------------------------------------------
  colnames(volcano)
  volcano <- volcano %>%
    mutate( neg_log10padj = -log10(padj)) %>%
    mutate(DEG = "NotSignificant" ) %>%
    mutate(DEG = ifelse( neg_log10padj > -log10(padjlevel) &
                           log2FoldChange > Up_FC, "Upregulated", DEG)) %>%
    mutate(DEG = ifelse( neg_log10padj > -log10(padjlevel) &
                           log2FoldChange < Down_FC, "Downregulated", DEG))


# ------------------------------------------------------------
# Section 5: Define Color Palette
# ------------------------------------------------------------
my_pal <- c("#943126", "#839192", "#A6761D", "#1E8449")


# ------------------------------------------------------------
# Section 6: Create Data Subsets
# ------------------------------------------------------------
  volcano_up <- volcano[which(volcano$DEG == "Upregulated"),]
  volcano_up <- volcano_up[order(volcano_up$log2FoldChange, decreasing = TRUE), ]

  volcano_down <- volcano[which(volcano$DEG == "Downregulated"),]
```

```r
  volcano_down <- volcano_down[order(volcano_down$log2FoldChange, decreasing = F), ]

  volcano_NotSignificant <- volcano[which(volcano$DEG == "NotSignificant" ),]

  volcano_all <- rbind(volcano_up, volcano_down, volcano_NotSignificant)


# ------------------------------------------------------------
# Section 7: Modify Data Frame for Plotting
# ------------------------------------------------------------
  colnames(volcano_all)

  ntop_rep <- rep("Top", ntop)
  volcano_all$DEG <- c(ntop_rep, volcano_up$DEG[-c(1:ntop)],
                       ntop_rep, volcano_down$DEG[-c(1:ntop)],
                       volcano_NotSignificant$DEG)

  volcano_all$Top <- ifelse(volcano_all$DEG == "Top",
                            volcano_all$Symbol, "")

# ------------------------------------------------------------
# Section 8: Create the Volcano Plot
# ------------------------------------------------------------
  g <- ggplot(data = volcano_all,
              aes(
                x = log2FoldChange,
                y =  neg_log10padj,
                color = DEG,
                fill = DEG,
                label = Top)
              ) +

      geom_point(size = 2, shape = 21) +
      geom_text(check_overlap = T, vjust = -0.1, nudge_y = 0.1) +
      scale_color_manual(values = my_pal) +
      scale_fill_manual(values = my_pal) +

      theme_classic() +
      labs(x = "Log Fold Change", y = "-log10Padjusted", title = "Volcano PLot")


# ------------------------------------------------------------
# Section 9: Customize the Plot and Save
# ------------------------------------------------------------
  g <- g + theme(axis.line = element_line(linetype = "solid"),
                 axis.title = element_text(size = 14),
```

```
                axis.text = element_text(size = 14, colour = "black"),
                plot.title = element_text(size = 16, hjust = 0.5)) +
      labs(title = "Volcano Plot (ggplot2-METHOD 1)")

  pdf("plots/volcanoplot.pdf", width = 10, height = 8)
  print(g)
  dev.off()

# ------------------------------------------------------------
# Section 10: Return the Plot
# ------------------------------------------------------------
  return(g)

}

# ------------------------------------------------------------
# Section 11: Call the volcano_plot Function
# ------------------------------------------------------------
volcano_plot(padjlevel = 0.05,
             Up_FC = 1,
             Down_FC = -1,
             ntop = 10)
```



A fully-optimized version that utilizes geom text to label the top 5 genes. The position of each gene label can be individually adjusted for the best visual representation. The fold change (FC) is included next to each gene name, like PRAMEF36P (FC:25).

This is an advanced rendition using ggplot2 focusing on visual clarity.

-Key features include:

- Identification and individual labeling of the top 5 upregulated and downregulated genes along with their fold change.

- Styling the plot with a minimalistic theme.

- Customized axes and titles for better clarity.

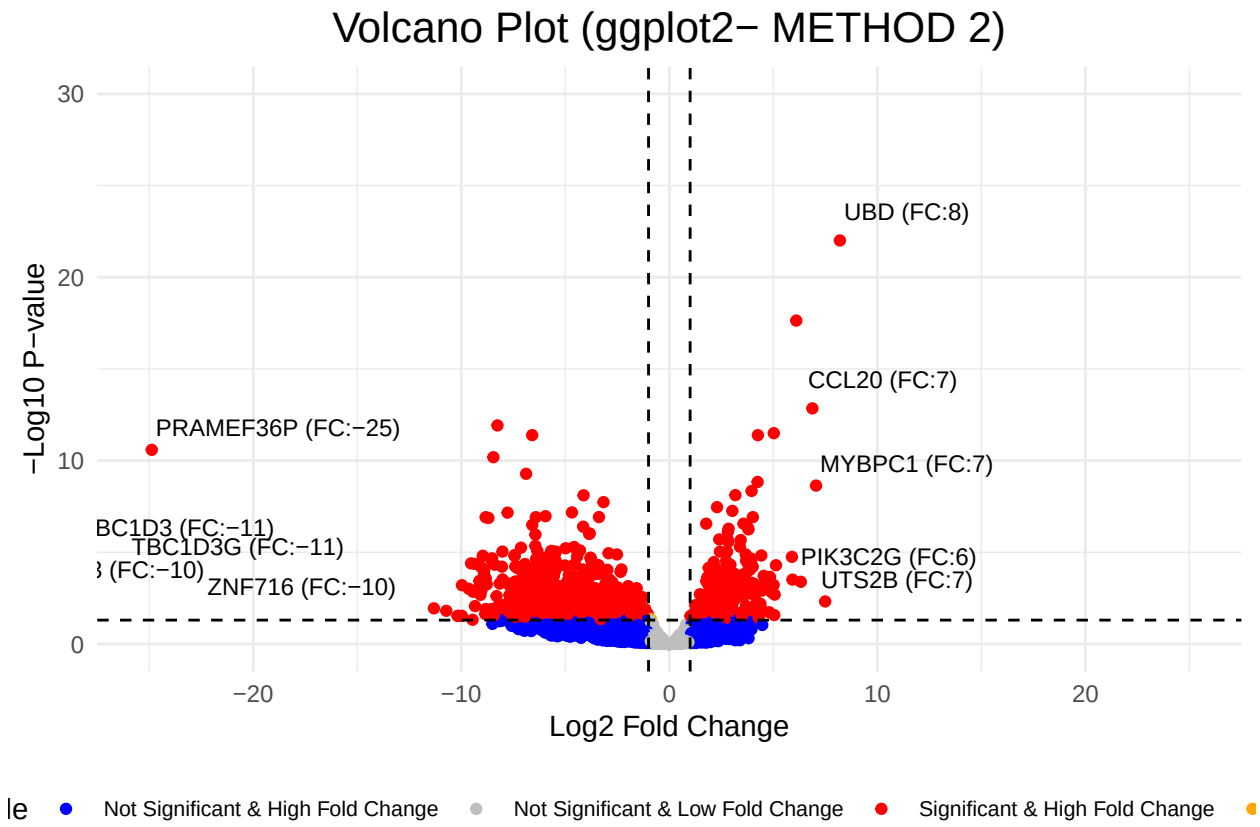
2. OPTIMIZED GGPLOT2 VOLCANO PLOT (METHOD 2)

```
# ------------------------------------------------------------
# Section 1: Load Required Packages
# ------------------------------------------------------------
# Load ggplot2 for plotting and dplyr for data manipulation
library(ggplot2)
library(dplyr)

# ------------------------------------------------------------
# Section 2: Read and Preprocess Data
# ------------------------------------------------------------
# Read the CSV file into a data frame
data <- read.csv("DESeq2/DESeq2_Unsorted_All_Genes.csv")

# ------------------------------------------------------------
# Section 3: Identify Top Genes
# ------------------------------------------------------------
# Filter the top 5 upregulated genes based on adjusted p-value and log2FoldChange
top_5_up <- data %>%
  filter(padj <= 0.05) %>%
  arrange(desc(log2FoldChange)) %>%
  head(5)


# Filter the top 5 downregulated genes based on adjusted p-value and log2FoldChange
top_5_down <- data %>%
  filter(padj <= 0.05) %>%
  arrange(log2FoldChange) %>%
```

```r
  head(5)

# ------------------------------------------------------------
# Section 4: Create Separate Data Subsets for Top 5 Genes
# ------------------------------------------------------------
# Create separate data subsets for each of the Top 5 genes
up_1 <- top_5_up[1, ]
up_2 <- top_5_up[2, ]
up_3 <- top_5_up[3, ]
up_4 <- top_5_up[4, ]
up_5 <- top_5_up[5, ]

down_1 <- top_5_down[1, ]
down_2 <- top_5_down[2, ]
down_3 <- top_5_down[3, ]
down_4 <- top_5_down[4, ]
down_5 <- top_5_down[5, ]

# ------------------------------------------------------------
# Section 5: Create Volcano Plot with ggplot2
# ------------------------------------------------------------
# Create Volcano Plot with ggplot2
ggplot(data, aes(x=log2FoldChange, y=-log10(padj))) +
  # Add points based on significance and fold change
  geom_point(aes(
    color =
      case_when(
        padj > 0.05 & abs(log2FoldChange) < 1  ~ "Not Significant & Low Fold Change",
        padj > 0.05 & abs(log2FoldChange) > 1  ~ "Not Significant & High Fold Change",
        padj <= 0.05 & abs(log2FoldChange) < 1 ~ "Significant & Low Fold Change",
        padj <= 0.05 & abs(log2FoldChange) > 1 ~ "Significant & High Fold Change"
      )
            ), alpha = 1) +

  # Add labels for top 5 upregulated genes
  geom_text(data = up_1, aes(
    label = sprintf("%s (FC:%d)", Symbol, round(log2FoldChange))),
    vjust = -1, hjust = 0, nudge_x = 0.2, nudge_y = 0.2, size = 3) +

  geom_text(data = up_2, aes(
    label = sprintf("%s (FC:%d)", Symbol, round(log2FoldChange))),
    vjust = -1, hjust = 0, nudge_x = -0.2, nudge_y = -0.2, size = 3) +

  geom_text(data = up_3, aes(
    label = sprintf("%s (FC:%d)", Symbol, round(log2FoldChange))),
    vjust = -1, hjust = 0, nudge_x = 0.2, nudge_y = -0.2, size = 3) +
```

```
geom_text(data = up_4, aes(
  label = sprintf("%s (FC:%d)", Symbol, round(log2FoldChange))),
  vjust = -1, hjust = 0, nudge_x = -0.2, nudge_y = 0.2, size = 3) +

geom_text(data = up_5, aes(
  label = sprintf("%s (FC:%d)", Symbol, round(log2FoldChange))),
  vjust = -1, hjust = 0, nudge_x = 0, nudge_y = 0, size = 3) +


 # Add labels for top 5 downregulated genes
geom_text(data = down_1, aes(
  label = sprintf("%s (FC:%d)", Symbol, round(log2FoldChange))),
  vjust = -1, hjust = 0, nudge_x = 0.2, nudge_y = -0.2, size = 3) + # PRAMEF36P

geom_text(data = down_2, aes(
  label = sprintf("%s (FC:%d)", Symbol, round(log2FoldChange))),
  vjust = -4, hjust = 1.5, nudge_x = -3, nudge_y = 0.2, size = 3) + # TBC1D3

geom_text(data = down_3, aes(
  label = sprintf("%s (FC:%d)", Symbol, round(log2FoldChange))),
  vjust = -3, hjust = 1.5, nudge_x = 0.2, nudge_y = 0.2, size = 3) + # TBC1D3G

geom_text(data = down_4, aes(
  label = sprintf("%s (FC:%d)", Symbol, round(log2FoldChange))),
  vjust = -2, hjust = 2.6, nudge_x = 0.2, nudge_y = 0.2, size = 3) + # KLK3

geom_text(data = down_5, aes(
  label = sprintf("%s (FC:%d)", Symbol, round(log2FoldChange))),
  vjust = -1, hjust = 1, nudge_x = -3, nudge_y = 0.2, size = 3) + # ZNF716

 # Custom scale, theme, and labels
scale_shape_manual(values = c("Top 5 Genes" = 3)) +
scale_color_manual(values = c("Not Significant & Low Fold Change" = "grey",
                              "Not Significant & High Fold Change" = "blue",
                              "Significant & Low Fold Change" = "orange",
                              "Significant & High Fold Change" = "red")) +
xlim(c(-25, 25)) +
ylim(c(0, 30)) +
geom_hline(yintercept = -log10(0.05), linetype = "dashed", color = "black") +
geom_vline(xintercept = c(-1, 1), linetype = "dashed", color = "black") +
theme_minimal() +
labs(
  title = "Volcano Plot (ggplot2- METHOD 2)",
  x = "Log2 Fold Change",
  y = "-Log10 P-value",
```

```
    color = "Custom Legend Title",
    shape = "Special Points"
  ) +

  theme(
    legend.position = "bottom",
    legend.text = element_text(size = 8),
    # 0 aligns the text to the left;
    # 0.5 centers the text;
    # 1 aligns the text to the right.
    plot.title = element_text(hjust = 0.5, size = 16)

  )
```



3. ENHANCED VOLCANO PLOT GENERATION

Purpose:

- This code chunk is responsible for generating the final version of the Enhanced Volcano plot.
- The plot visualizes the differentially expressed genes based on their

log2FoldChange and adjusted p-values (padj).

Key Features:
- Custom coloring schemes based on significance and fold change.
- Highlights and labels the top 5 upregulated and downregulated genes.
- Axes and labels are appropriately scaled and styled.
- Y-axis is transformed using -log10 for better visual discrimination of smaller p-values.

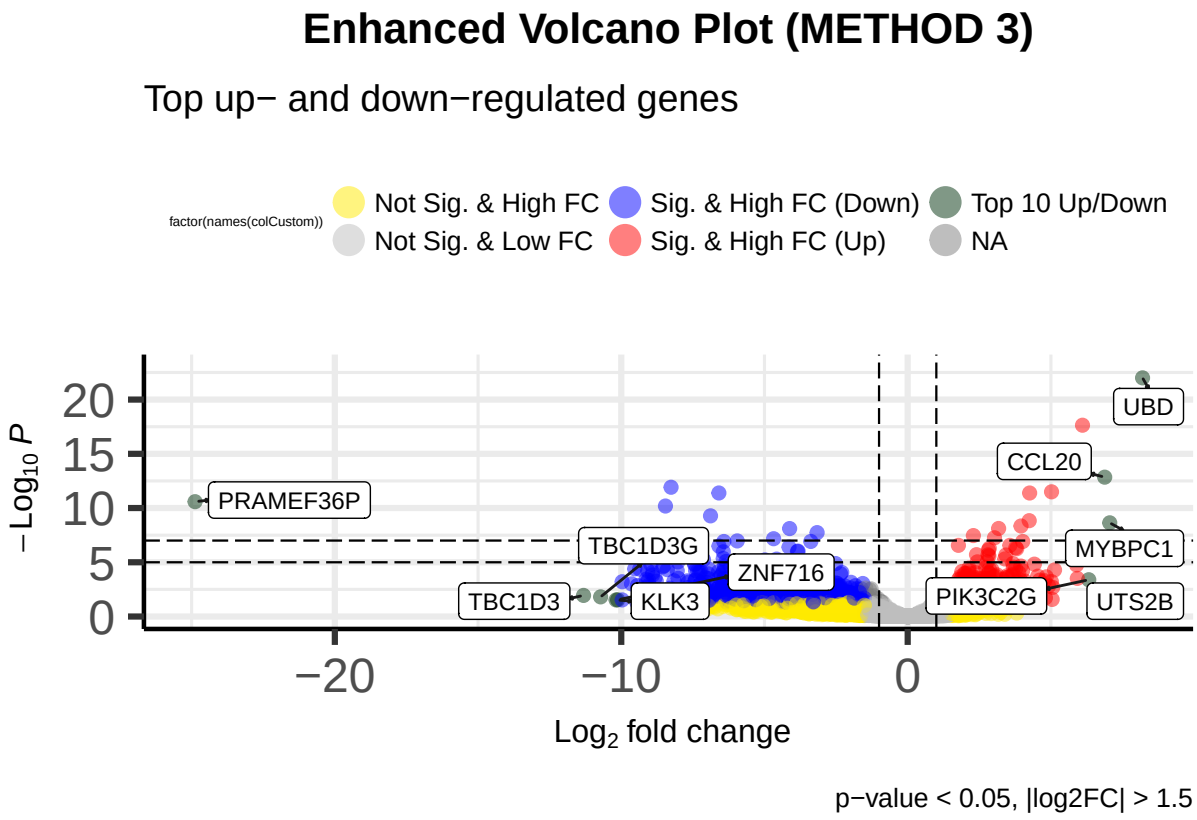
The plot is fine-tuned to be publication-ready, incorporating best practices in data visualization.

3.Enhanced Volcano Plot Generation

```
# ------------------------------------------------------------
# Load Required Packages
# ------------------------------------------------------------
library(EnhancedVolcano)
library(dplyr)

# ------------------------------------------------------------
# Read and Preprocess Data
# ------------------------------------------------------------
res <- read.csv("DESeq2/DESeq2_Unsorted_All_Genes.csv")

# ------------------------------------------------------------
# Identify Top 5 Upregulated and Downregulated Genes
# ------------------------------------------------------------
top_5_up_EnhancedVolcano <- res %>%
  filter(padj <= 0.05) %>%
  arrange(desc(log2FoldChange)) %>%
  head(5)

top_5_down_EnhancedVolcano <- res %>%
  filter(padj <= 0.05) %>%
  arrange(log2FoldChange) %>%
  head(5)

# ------------------------------------------------------------
# Create Keyvals for Custom Colors
# ------------------------------------------------------------
# Create Keyvals for Custom Colors
keyvals <- case_when(
    res$padj > 0.05 & abs(res$log2FoldChange) < 1.5  ~ 'gray',
```

```
    res$padj > 0.05 & abs(res$log2FoldChange) >= 1.5 ~ '#ffea00',
    res$padj <= 0.05 & res$log2FoldChange > 1.5      ~ '#FF0000',
    res$padj <= 0.05 & res$log2FoldChange < -1.5     ~ '#0000FF',
    TRUE                                           ~ 'default'
)


# Assign Names to the Keyvals for the Legend
names(keyvals)[keyvals == 'gray'] <- 'Not Sig. & Low FC'
names(keyvals)[keyvals == '#ffea00'] <- 'Not Sig. & High FC'
names(keyvals)[keyvals == '#FF0000'] <- 'Sig. & High FC (Up)'
names(keyvals)[keyvals == '#0000FF'] <- 'Sig. & High FC (Down)'

# Make Top 5 genes color green
keyvals[res$Symbol %in% top_5_up_EnhancedVolcano$Symbol] <- '#03330f'
keyvals[res$Symbol %in% top_5_down_EnhancedVolcano$Symbol] <- '#03330f'
names(keyvals)[keyvals == '#03330f'] <- 'Top 10 Up/Down'

# Prepare a list of gene symbols that should be labeled
select_labels <- c(top_5_up_EnhancedVolcano$Symbol, top_5_down_EnhancedVolcano$Symbol)

# Generate the EnhancedVolcano plot
ev_plot <- EnhancedVolcano(res,
                lab = res$Symbol,  # Labels all points
                x = 'log2FoldChange',
                y = 'padj',
                pointSize = 2,
                colCustom = keyvals,
                labSize = 3,
                drawConnectors = TRUE,
                widthConnectors = 0.5,
                box = TRUE,  # Draw box around labels
                # ALTERNATIVE: xlim = c(-13, 9),
                # ALTERNATIVE: ylim = c(0, 30),

                # Sets the y-axis limit based on -log10 transformation,
                # making smaller p-values more visually prominent
                xlim = c(-25, 8.3),
                ylim = c(0, -log10(10e-24)),
                legendLabSize = 10,

                hline = c(10e-8),
                # Only labels for top 5 up/down will be displayed
                selectLab = select_labels,
                title = "Enhanced Volcano Plot (METHOD 3)",
                subtitle = "Top up- and down-regulated genes",
```

```
                caption = "p-value < 0.05, |log2FC| > 1.5"
)


# Modify the text sizes
ev_plot <- ev_plot + theme(
    #legend.text = element_text(size = 5),
    legend.title = element_text(size = 5),
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12),
    plot.title = element_text(size = 16, hjust = 0.5),  # Centered title
    plot.subtitle = element_text(size = 14),
    plot.caption = element_text(size = 10)
)

# Print the modified plot
print(ev_plot)
```



About the above code

```
# 1- I used ifelse statements:
# ifelse:
```

```r
keyvals <-
  ifelse(res$padj > 0.05 & abs(res$log2FoldChange) < 1.5, 'gray',
    ifelse(res$padj > 0.05 & abs(res$log2FoldChange) >= 1.5, '#ffea00',
      ifelse(res$padj <= 0.05 & res$log2FoldChange > 1.5, '#FF0000',
        ifelse(res$padj <= 0.05 & res$log2FoldChange < -1.5, '#0000FF', 'default'))))

# case_when:
keyvals <- case_when(
    res$padj > 0.05 & abs(res$log2FoldChange) < 1.5  ~ 'gray',
    res$padj > 0.05 & abs(res$log2FoldChange) >= 1.5 ~ '#ffea00',
    res$padj <= 0.05 & res$log2FoldChange > 1.5      ~ '#FF0000',
    res$padj <= 0.05 & res$log2FoldChange < -1.5     ~ '#0000FF',
    TRUE                                           ~ 'default'
)


## we can use dplyr::case_when() as a replacement for nested ifelse() statements.
## case_when() can be more readable and maintainable,
## especially when you have multiple conditions to check.
```