Project

Suraj Chauhan

April 17,2o25

—

Biostatistics

—

Prof. Nagarjun Vijay

1. CODE FOR MERGING FILES
2. CODE FOR ALTERNATIVE MERGING OF FILES
3. CODE FOR HORIZONTAL BAR GRAPH - INFECTIVITY RATIO
4. CODE FOR HISTOGRAMS FOR 15 CELL LINES
5. CODE FOR SCATTER PLOT FOR SERINC5 EXPRESSION
6. CODE FOR SHAPIRO WILK TEST FOR ALL CELL LINES
7. CODE FOR WILCOXIN TEST
8. CODE FOR PARTIAL CORRELATION
9. CODES FOR COMPLETE CORRELATION
10. CODE FOR SKEWNESS AND KURTOSIS
11. CODE FOR HYPOTHESIS TESTING

CODE FOR MERGING FILES

```
# ---------- Step 0: Lo Required Libraries ----------
library(dplyr)    # For data manipulation
```

```
library(tibble)   # For working with tidy data frames

# ---------- Step 1: Set the Path to HTSeq-Count Files ----------
# Update the path to your HTSeq files
htseq_path <- "C:/Users/suraj/Documents/cell_lines/cell lines data"

# ---------- Step 2: Read Metadata ----------
# 'infect.txt' contains columns: SRR_ID, Infectivity, Cell_Line
meta <- read.table(
  file = file.path(htseq_path, "infect.txt"),
  header = FALSE, sep = "\t", stringsAsFactors = FALSE,
  col.names = c("SRR_ID", "Infectivity", "Cell_Line")
)

# ---------- Step 3: Get List of HTSeq Files ----------
# List all files starting with SRR and ending with .htseq
htseq_files <- list.files(
  path = htseq_path,
  pattern = "^SRR.*\\.htseq$",
  full.names = TRUE
)

# ---------- Step 4: Initialize List to Store Each SRR File ----------
htseq_list <- list()

# ---------- Step 5: Read HTSeq Files and Add to List ----------
for (file in htseq_files) {
  srr_id <- sub(".htseq$", "", basename(file))  # Extract SRR ID
  data <- read.table(
    file, header = FALSE, sep = "\t", stringsAsFactors = FALSE,
    col.names = c("GeneID", "Count")
  )
  colnames(data)[2] <- srr_id                # Rename column to SRR ID
  htseq_list[[srr_id]] <- data               # Add to list
}

# ---------- Step 6: Merge All Data by GeneID ----------
merged_data <- Reduce(function(x, y) merge(x, y, by = "GeneID"), htseq_list)

# ---------- Step 7: Keep Only Relevant SRR Columns Present in Metadata ----------
available_ids <- intersect(meta$SRR_ID, colnames(merged_data))  # Common SRR IDs
meta <- meta[meta$SRR_ID %in% available_ids, ]                  # Filter metadata
merged_data <- merged_data[, c("GeneID", available_ids)]        # Reorder columns in merged data

# ---------- Step 8: Prepare Annotation Rows (Metadata Rows) ----------
srr_id_row      <- c("SRR ID",      meta$SRR_ID)
cell_line_row   <- c("Cell Line",   meta$Cell_Line)
infectivity_row <- c("Infectivity", meta$Infectivity)

# ---------- Step 9: Combine Annotation and Expression Data ----------
# Convert gene expression data into character matrix for compatibility
data_matrix <- apply(merged_data, 1, as.character)

# Transpose matrix so that each gene becomes a row and SRR IDs become columns
```

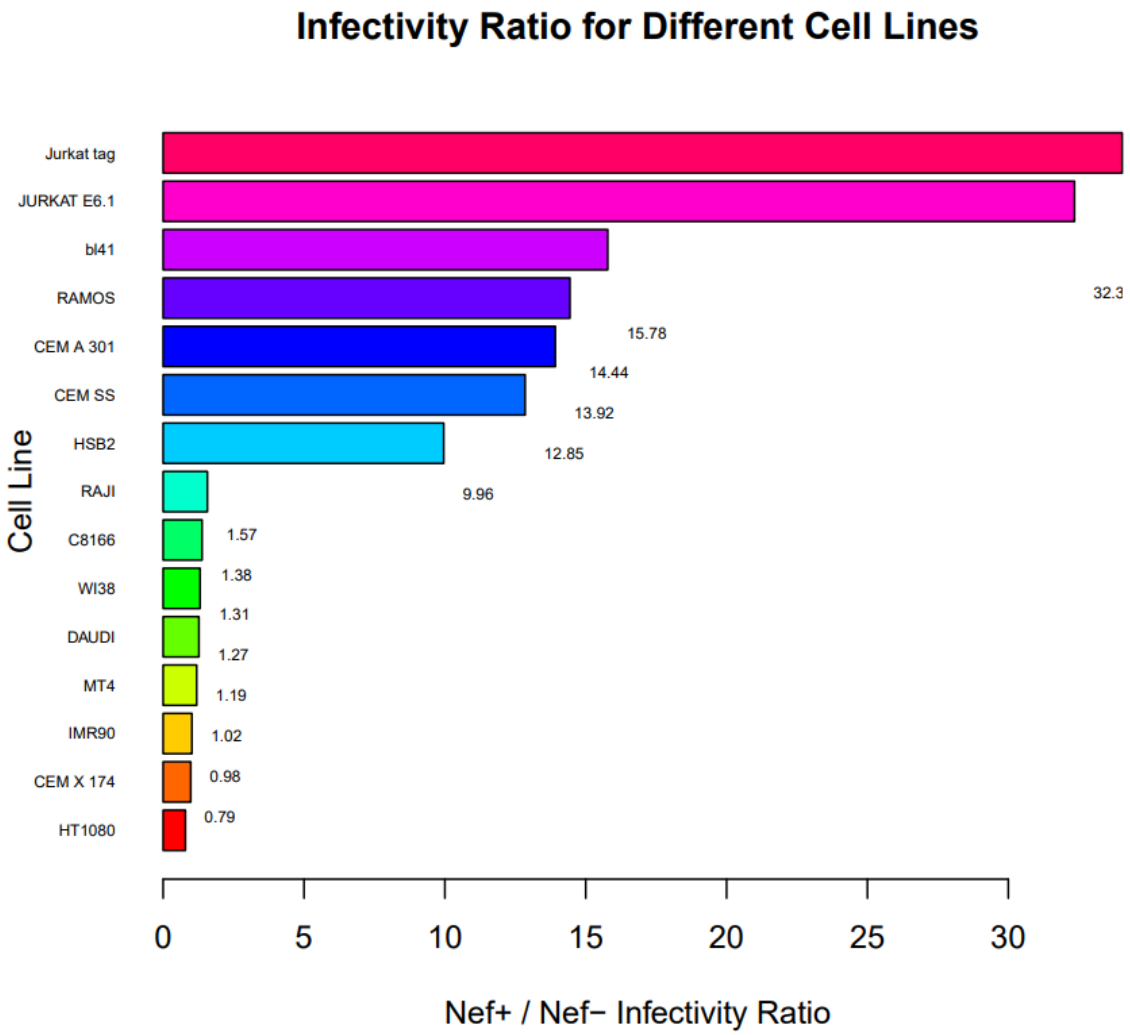
```
final_output <- rbind(
  srr_id_row,
  cell_line_row,
  infectivity_row,
  t(data_matrix)   # t() transposes rows ↔ columns
)

# -------------------- Step 10: Write the Final Output to a CSV File --------------------
output_path <- file.path(htseq_path, "merged_htseq_data.csv")
write.table(
  final_output, file = output_path, sep = ",",
  row.names = FALSE, col.names = FALSE, quote = FALSE
)

cat("✅ Merged file successfully written to:\n", output_path, "\n")
```

```
# 📦 Load required library
library(dplyr)

# 📁 Define file paths
htseq_dir   <- "C:/Users/suraj/Documents/cell_lines/cell lines data"     # Directory containing SRR htseq-count files
infect_file <- file.path(htseq_dir, "infect.txt")                    # Metadata file
output_file <- file.path(htseq_dir, "merged_data.csv")                  # Output file path

# 📄 Step 1: Read metadata (infect.txt)
infect_data <- read.table(infect_file, header = FALSE, sep = "", stringsAsFactors = FALSE)
colnames(infect_data) <- c("SRR_ID", "Infectivity_Ratio", "Cell_Line_Name")

# 📃 Step 2: List all htseq-count files starting with "SRR"
srr_files <- list.files(htseq_dir, pattern = "^SRR.*", full.names = TRUE)

# 🧬 Step 3: Initialize container to hold merged data
merged_data <- NULL

# 🔁 Step 4: Read each file and merge counts by Gene_ID
for (file in srr_files) {

  # 🔍 Extract SRR ID (used as column name)
  srr_id <- basename(file)

  # 🍼 Read gene count file safely
  data <- tryCatch({
    read.table(file, header = FALSE, sep = "\t", stringsAsFactors = FALSE, fill = TRUE, comment.char = "#")
  }, error = function(e) {
    cat("⚠️ Error reading:", file, "\n", e$message, "\n")
    return(NULL)
  })

  # 🚫 Skip if reading failed
  if (is.null(data)) next
```

```
  # ✅ Ensure file has exactly 2 columns: Gene_ID and Count
  if (ncol(data) != 2) {
    cat("⚠️ Skipping", file, "- Incorrect column count\n")
    next
  }

  # 🏷️ Assign column names
  colnames(data) <- c("Gene_ID", srr_id)

  # 🔗 Merge by Gene_ID
  if (is.null(merged_data)) {
    merged_data <- data
  } else {
    merged_data <- merge(merged_data, data, by = "Gene_ID", all = TRUE)
  }
}

# 🖌️ Step 5: Replace NAs with 0 (assuming missing genes have zero counts)
merged_data[is.na(merged_data)] <- 0

# 🧠 Step 6: Extract SRR IDs (column names except first one)
srr_ids <- colnames(merged_data)[-1]

# 🧬 Step 7: Add annotation rows (cell line & infectivity)
cell_line_names <- sapply(srr_ids, function(id) {
  idx <- match(id, infect_data$SRR_ID)
  if (!is.na(idx)) return(infect_data$Cell_Line_Name[idx]) else return(NA)
})

infectivity_ratios <- sapply(srr_ids, function(id) {
  idx <- match(id, infect_data$SRR_ID)
  if (!is.na(idx)) return(infect_data$Infectivity_Ratio[idx]) else return(NA)
})

# 📊 Step 8: Combine metadata rows and expression matrix
final_data <- rbind(
  c("SRR_ID", srr_ids),
  c("Cell_Line_Name", cell_line_names),
  c("Infectivity_Ratio", infectivity_ratios),
  merged_data
)

# 💾 Step 9: Export final merged file to CSV
write.csv(final_data, output_file, row.names = FALSE, quote = FALSE)
cat("✅ Merged data saved to:", output_file, "\n")
```

```
# 📦 Load required library
```

```
library(ggplot2)

# 📄 Step 1: Read infectivity data
infect_data <- read.table("infect.txt", header = FALSE, sep = "\t", stringsAsFactors = FALSE)
colnames(infect_data) <- c("Sample", "Infectivity", "CellLine")

# 🔽 Step 2: Sort data by Infectivity (highest to lowest)
infect_data <- infect_data[order(infect_data$Infectivity, decreasing = TRUE), ]

# 🎨 Step 3: Assign custom colors based on Cell Line categories
infect_data$Color <- "gray"  # Default color

# Group 1: T-cell lineage
infect_data$Color[infect_data$CellLine %in% c("JURKAT E6.1", "Jurkat tag")] <- "navy"

# Group 2: B-cell lineage
infect_data$Color[infect_data$CellLine %in% c("bl41", "RAMOS", "CEM A 301", "CEM SS", "HSB2")] <-
"darkgreen"

# Group 3: Others / Fibroblast / Non-hematopoietic
infect_data$Color[infect_data$CellLine %in% c("C8166", "WI38", "DAUDI", "MT4", "IMR90", "CEM X
174", "HT1080")] <- "firebrick"

# 📊 Step 4: Plot horizontal bar graph using ggplot2
ggplot(infect_data, aes(x = reorder(CellLine, Infectivity), y = Infectivity, fill = Color)) +

  geom_bar(stat = "identity", width = 0.7, color = "black", show.legend = FALSE) +  # Border for aesthetics

  scale_fill_identity() +  # Use manually assigned colors directly

  coord_flip() +  # Flip coordinates to make horizontal bars

  theme_minimal(base_size = 14) +  # Clean theme with larger base font

  labs(
    title = "Infectivity Ratio (Nef+ / Nef-) by Cell Line",
    x = "",
    y = "Infectivity Ratio"
  ) +

  theme(
    plot.title = element_text(hjust = 0.5, face = "bold", size = 16, color = "steelblue"),
    axis.text.y = element_text(size = 12, face = "bold"),
    axis.text.x = element_text(size = 12),
    axis.title.y = element_text(size = 14, face = "bold"),
    panel.grid.major.y = element_blank(),  # Cleaner y-axis
    panel.grid.minor = element_blank()
  )
```

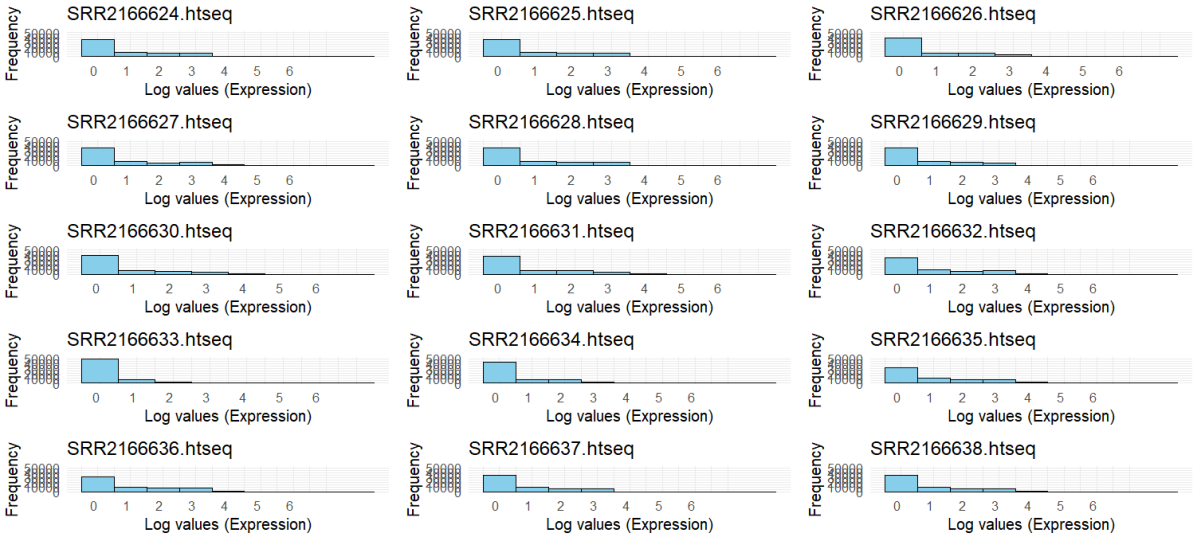


# 📦 Load required libraries
library(ggplot2)
library(gridExtra)
library(tidyr)
library(dplyr)

# 📄 Step 1: Read the merged expression data from the new directory
data <- read.csv("C:/Users/suraj/Documents/cell_lines/cell lines data/merged_data.csv", stringsAsFactors = FALSE)

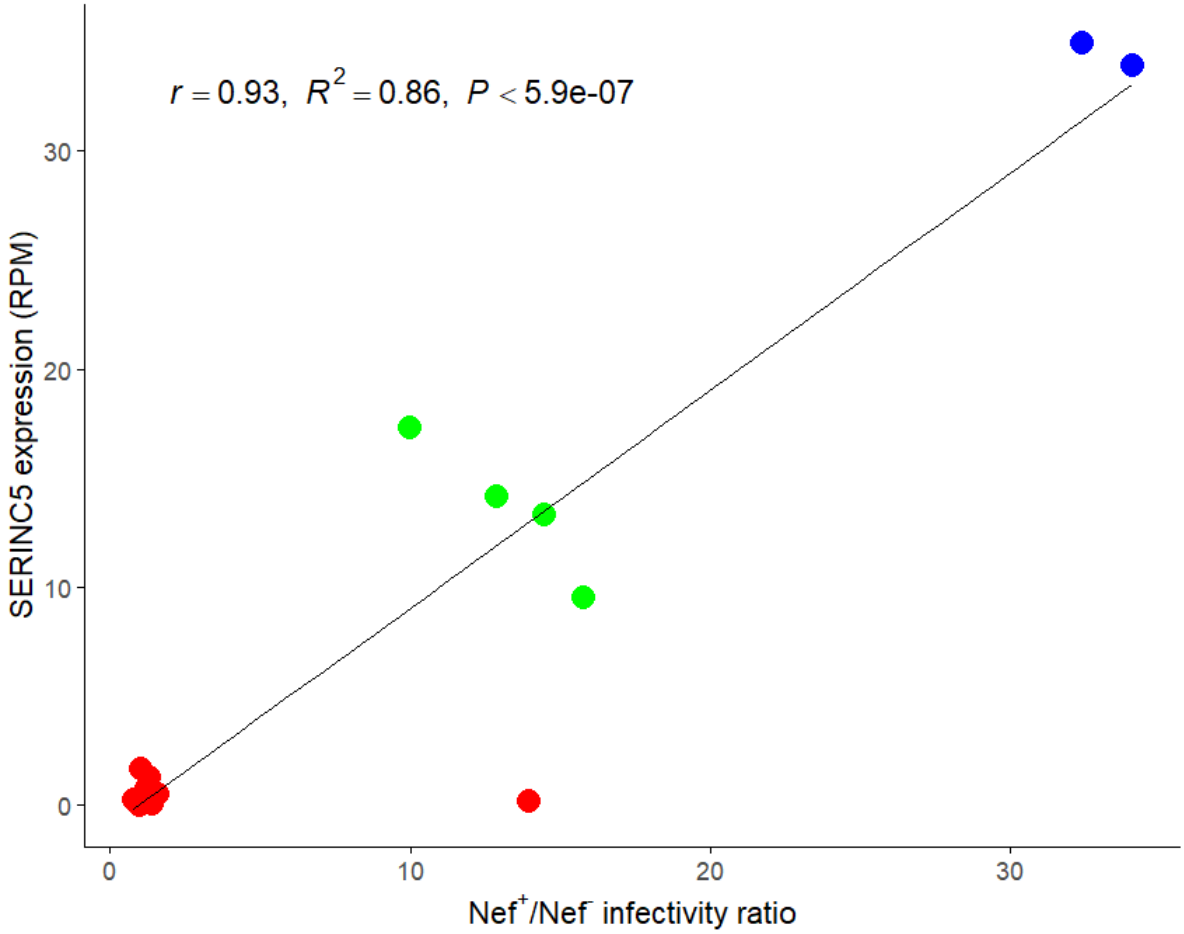
🛠️ Step 2: Rename the first column to "Gene" if it's not already
colnames(data)[1] <- "Gene"

```r
# 🔁 Step 3: Reshape to long format — one row per (Gene × CellLine)
long_data <- pivot_longer(
  data,
  cols = -Gene,
  names_to = "CellLine",
  values_to = "Expression"
)

# 🖌️ Step 4: Clean data and apply log transformation
long_data <- long_data %>%
  mutate(Expression = as.numeric(Expression)) %>%
  filter(!is.na(Expression) & is.finite(Expression)) %>%
  mutate(LogExpression = log10(Expression + 1))

# 🎨 Step 5: Generate histograms per cell line
plot_list <- long_data %>%
  split(.$CellLine) %>%
  lapply(function(df) {
    ggplot(df, aes(x = LogExpression)) +
      geom_histogram(binwidth = 0.25, fill = "#69b3a2", color = "black", size = 0.2) +
      labs(
        title = df$CellLine[1],
        x = "Log₁₀(Expression + 1)",
        y = "Gene Count"
      ) +
      theme_minimal(base_size = 12) +
      theme(
        plot.title = element_text(size = 13, face = "bold", hjust = 0.5, color = "steelblue"),
        axis.text = element_text(size = 10),
        axis.title = element_text(size = 11, face = "bold")
      ) +
      scale_x_continuous(breaks = seq(0, 6, by = 1), limits = c(0, 6)) +
      coord_cartesian(ylim = c(0, 50000))
  })

# 🧱 Step 6: Display all histograms in a grid layout (3 per row)
do.call(grid.arrange, c(plot_list, ncol = 3))
```




```
# 📦 Load required libraries
library(ggplot2)
library(ggpubr)

# 📄 Step 1: Read the merged expression data
df <- read.csv("C:/Users/suraj/Documents/cell_lines/cell lines data/merged_data.csv",
               header = TRUE, stringsAsFactors = FALSE)

# 🧬 Step 2: Extract gene expression for SERINC5 (ENSG00000164300)
# Row matching the SERINC5 gene ID, excluding first column
gene_row <- df[df$Gene == "ENSG00000164300", -1]
gene_counts <- suppressWarnings(as.numeric(gene_row)) # Convert to numeric safely

# 📊 Step 3: Get total reads from row 4 for RPM normalization
total_reads <- suppressWarnings(as.numeric(df[4, -1]))

# 🔬 Step 4: Calculate RPM (Reads Per Million)
rpm <- (gene_counts / total_reads) * 1e6

# 🧮 Step 5: Normalize RPM to a 0–35 scale for plotting
normalized_rpm <- (rpm / max(rpm, na.rm = TRUE)) * 35

# ⚙️ Step 6: Get Infectivity ratio from row 3
infectivity_ratio <- suppressWarnings(as.numeric(df[3, -1]))

# 📋 Step 7: Combine into one dataframe
plot_data <- data.frame(
  Infectivity_Ratio = infectivity_ratio,
  Normalized_RPM = normalized_rpm
)

# 🧼 Step 8: Clean up NA or infinite values
plot_data <- na.omit(plot_data)
plot_data <- plot_data[is.finite(plot_data$Infectivity_Ratio) & is.finite(plot_data$Normalized_RPM), ]

# 🎨 Step 9: Add expression-based color categories
```

```
plot_data$Color <- ifelse(
  plot_data$Normalized_RPM > 25, "high",
  ifelse(plot_data$Normalized_RPM > 7, "medium", "low")
)

# 🌟 Step 10: Final Scatter Plot with correlation line
ggplot(plot_data, aes(x = Infectivity_Ratio, y = Normalized_RPM, color = Color)) +
  geom_point(size = 4) +  # Larger points for better visibility
  scale_color_manual(values = c("low" = "red", "medium" = "orange", "high" = "blue")) +
  geom_smooth(method = "lm", se = FALSE, color = "black", size = 0.5) +  # Trend line
  stat_cor(
    method = "pearson",
    aes(label = paste0("italic(r)==", ..r.., "*','~~italic(R)^2==", ..rr.., "*','~~italic(P)==", ..p..)),
    parse = TRUE,
    label.x = 2, label.y = 33,  # Adjust based on max RPM scale
    size = 4.5, color = "black"
  ) +
  labs(
    x = expression("Nef"^"+"*"/Nef"^"-"* " Infectivity Ratio"),
    y = "SERINC5 Expression (Normalized RPM)"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    panel.grid = element_blank(),
    legend.position = "none",
    axis.title = element_text(size = 13, face = "bold"),
    axis.text = element_text(size = 11),
    axis.line = element_line(color = "black"),
    axis.ticks = element_line(color = "black")
  )
```



🧠 WHY: Shapiro-Wilk test tells us if gene expression data in each cell line follows a normal distribution (bell-shaped curve).
This helps decide whether to use parametric or non-parametric tests later.

```
# Load necessary package
library(dplyr)

# Step 1: Load the file with gene expression data
csv_file <- "C:/Users/suraj/Documents/cell_lines/cell lines data/gene id and cell line.csv"
data_raw <- read.csv(csv_file, header = FALSE, stringsAsFactors = FALSE)
# ---

# Step 2: Extract column names (cell line names) from row 1, except the first column (GeneID)
sample_ids <- as.character(data_raw[1, -1])
# ---

# Step 3: Remove metadata rows and rename columns
gene_expr <- data_raw[-(1:3), ]
colnames(gene_expr) <- c("GeneID", sample_ids)
# ---

# Step 4: Make sure all gene expression numbers are actually numeric
gene_expr[,-1] <- suppressWarnings(sapply(gene_expr[,-1], as.numeric))
gene_expr <- as.data.frame(gene_expr, stringsAsFactors = FALSE)
# ---
```

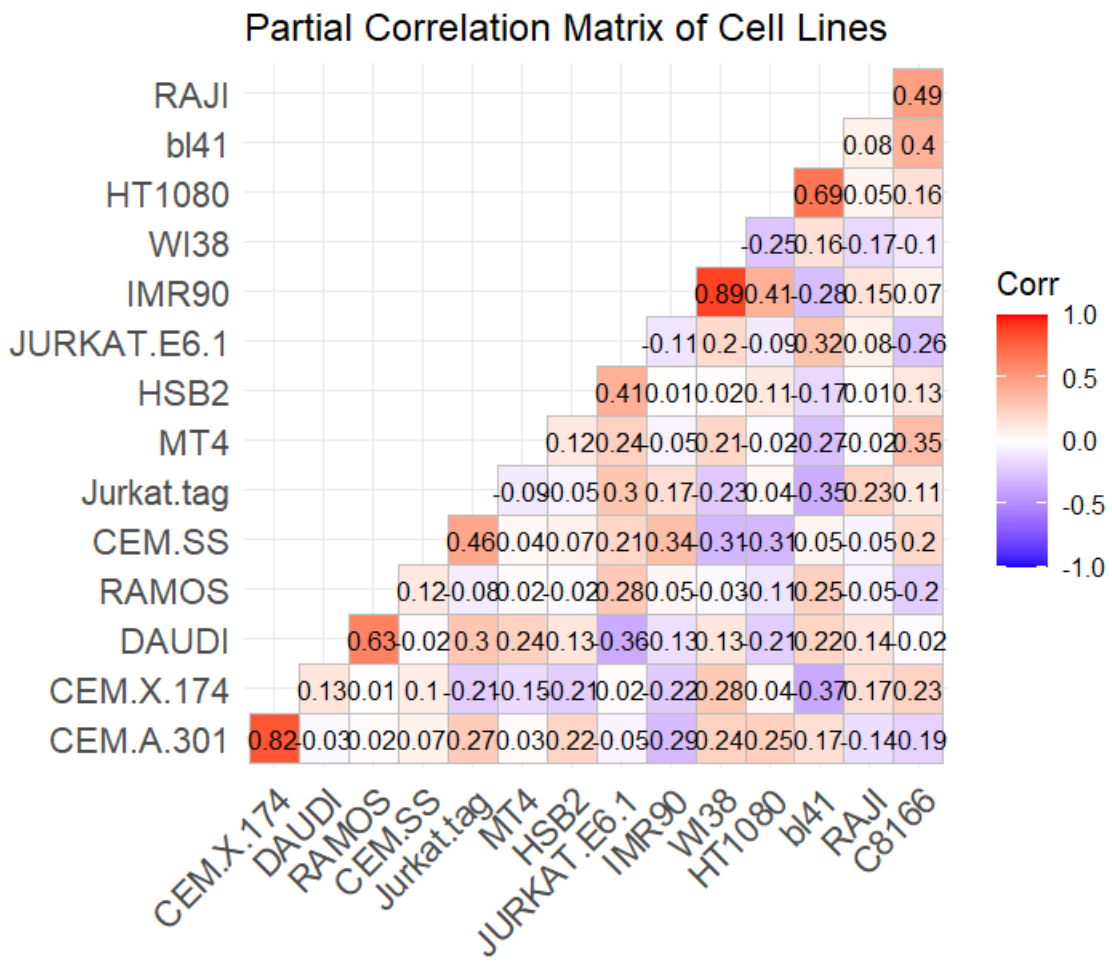
```
# Step 5: Define function to do Shapiro-Wilk safely
shapiro_test_safe <- function(values) {
  values <- na.omit(values)
  if (length(values) > 5000) {
    values <- sample(values, 5000)
  }
  if (length(values) >= 3 && length(unique(values)) > 1) {
    return(shapiro.test(values))
  } else {
    return(list(statistic = NA, p.value = NA))
  }
}
# ---

# Step 6: Run the test on all cell lines
normality_results <- lapply(sample_ids, function(samp) shapiro_test_safe(gene_expr[[samp]]))
names(normality_results) <- sample_ids
# ---

# Step 7: Organize results nicely
final_results <- data.frame(
  W_statistic = sapply(normality_results, function(res) round(res$statistic, 4)),
  p_value = sapply(normality_results, function(res) signif(res$p.value, 3)),
  stringsAsFactors = FALSE
)
print(final_results)
# ------------------------------------------------------------------------------------------------------------------------


# 🧠 WHY: We use the Wilcoxon test when the data isn't normally distributed. It compares the values against their own median.

# Reusing previous data loading steps...

# Step 1: Define function to do Wilcoxon safely
wilcoxon_test_safe <- function(values) {
  values <- na.omit(values)
  if (length(values) > 5000) {
    values <- sample(values, 5000)
  }
  if (length(values) >= 3 && length(unique(values)) > 1) {
    return(wilcox.test(values, mu = median(values), exact = FALSE, correct = TRUE))
  } else {
    return(list(statistic = NA, p.value = NA))
  }
}
# ---

# Step 2: Apply it on all samples
wilcoxon_results <- lapply(sample_ids, function(samp) wilcoxon_test_safe(gene_expr[[samp]]))
names(wilcoxon_results) <- sample_ids
# ---
```

```
# Step 3: Make a table
final_results <- data.frame(
  W_statistic = sapply(wilcoxon_results, function(res) round(res$statistic, 4)),
  p_value = sapply(wilcoxon_results, function(res) signif(res$p.value, 5)),
  stringsAsFactors = FALSE
)
print(final_results)
# ------------------------------------------------------------------------------------------------------------------------
```

🧠 WHY: Partial correlation removes the effect of other variables so we see how strongly two cell lines are connected without noise.

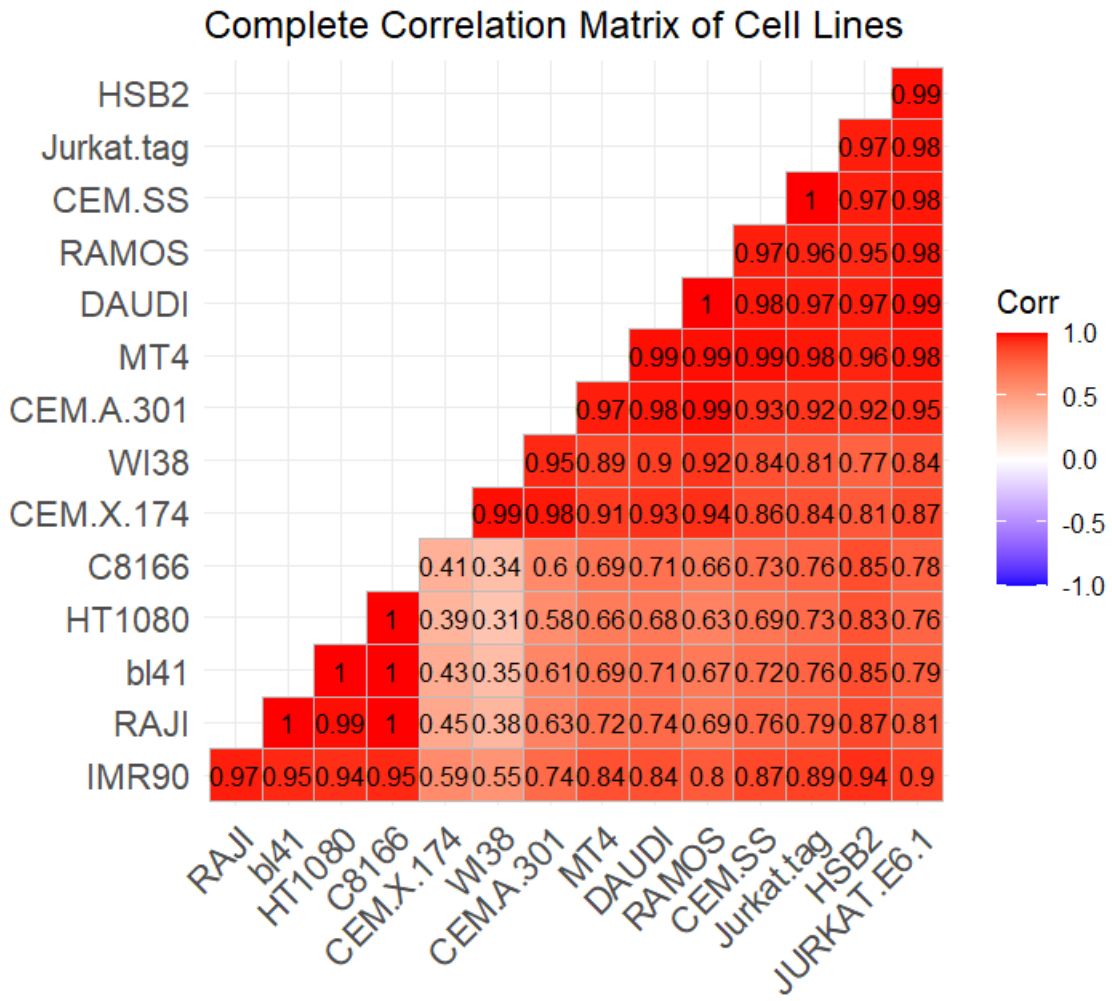
```
library(ppcor)
library(ggcorrplot)

# Step 1: Load the cleaned data
data <- read.csv("C:/Users/suraj/Documents/cell_lines/cell lines data/gene id and cell line.csv",
stringsAsFactors = FALSE)
data <- data[-c(1:3), ]
colnames(data)[1] <- "Gene_ID"
data[, -1] <- apply(data[, -1], 2, function(x) as.numeric(as.character(x)))
# ---

# Step 2: Compute partial correlation
partial_corr_matrix <- pcor(data[, -1])$estimate
# ---

# Step 3: Plot heatmap
corr_plot <- ggcorrplot(partial_corr_matrix, hc.order = TRUE, type = "lower",
                        lab = TRUE, lab_size = 3,
                        colors = c("blue", "white", "red"),
                        title = "Partial Correlation Matrix of Cell Lines")
png("partial_correlation_plot.png", width = 1000, height = 800)
```

print(corr_plot)dev.off()



🧠 WHY: This is simple Pearson correlation, useful to check how closely two cell lines behave based on expression.

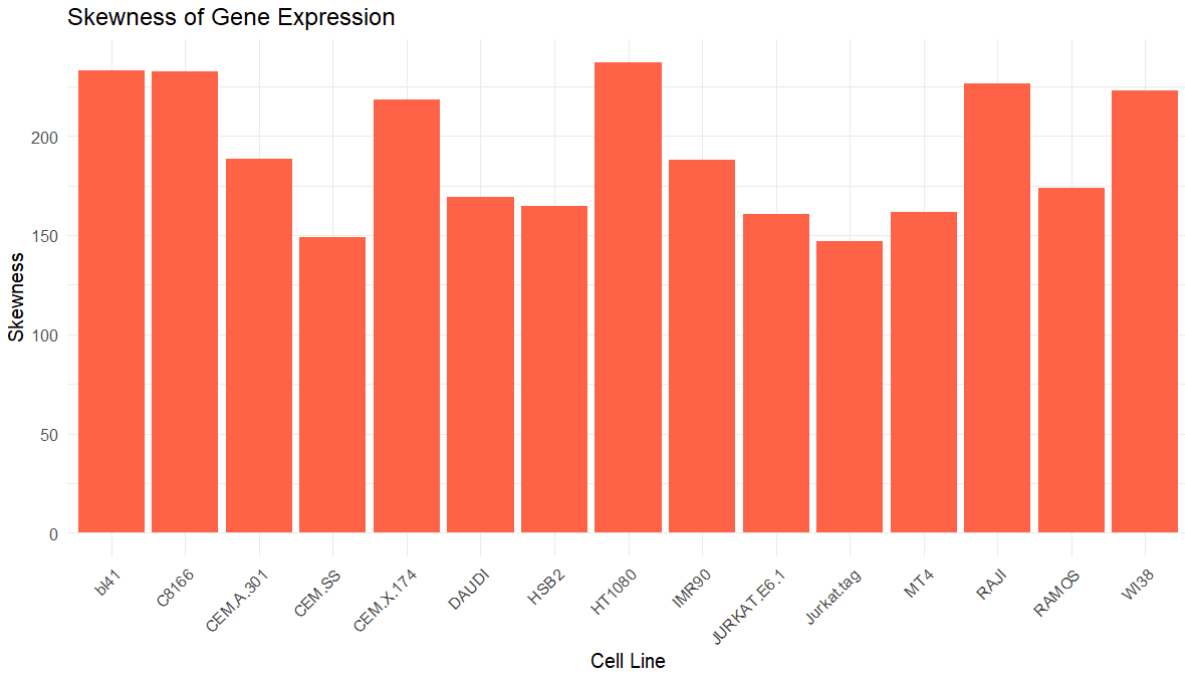
```
library(ggcorrplot)

data <- read.csv("C:/Users/suraj/Documents/cell_lines/cell lines data/gene id and cell line.csv",
stringsAsFactors = FALSE)
data <- data[-c(1:3), ]
colnames(data)[1] <- "Gene_ID"
data[, -1] <- apply(data[, -1], 2, function(x) as.numeric(as.character(x)))

corr_matrix <- cor(data[, -1], use = "complete.obs", method = "pearson")

corr_plot <- ggcorrplot(corr_matrix, hc.order = TRUE, type = "lower",
                        lab = TRUE, lab_size = 3,
                        colors = c("blue", "white", "red"),
                        title = "Complete Correlation Matrix of Cell Lines")
png("complete_correlation_plot.png", width = 1000, height = 800)
print(corr_plot)
dev.off()
```

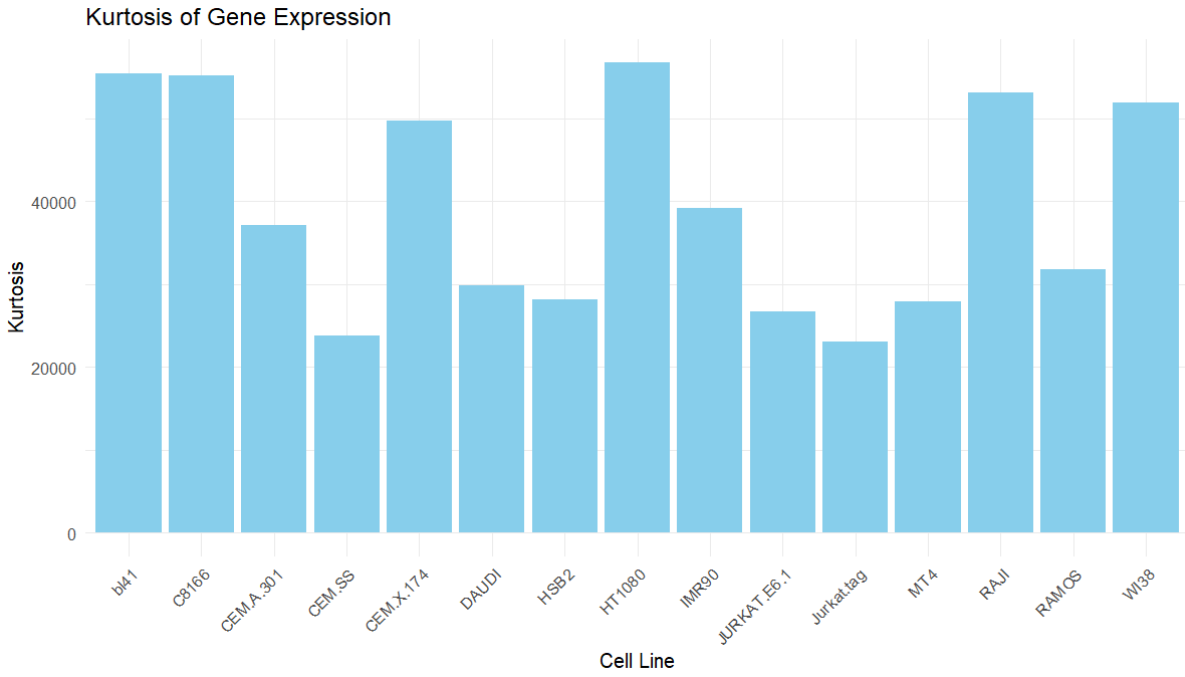


🧠 WHY: Skewness tells us if the data is lopsided (left or right).
Kurtosis tells us if the data has heavy tails (extreme values).

```
library(e1071)
file_path <- "C:/Users/suraj/Documents/cell_lines/cell lines data/gene id and cell line.csv"
df <- read.csv(file_path, stringsAsFactors = FALSE)
df_clean <- df[-c(1:3), ]

for (col in names(df_clean)[-1]) {
  df_clean[[col]] <- as.numeric(df_clean[[col]])
}
skewness_vals <- sapply(df_clean[-1], skewness, na.rm = TRUE)
kurtosis_vals <- sapply(df_clean[-1], kurtosis, na.rm = TRUE)

summary_stats <- cbind(Skewness = skewness_vals, Kurtosis = kurtosis_vals)
summary_stats <- as.data.frame(summary_stats)
summary_stats$Cell_Line <- rownames(summary_stats)
```

```
# Plotting
ggplot(summary_stats, aes(x = Cell_Line, y = Skewness)) +
  geom_bar(stat = "identity", fill = "tomato") +
  labs(title = "Skewness of Gene Expression", x = "Cell Line", y = "Skewness") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

ggplot(summary_stats, aes(x = Cell_Line, y = Kurtosis)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Kurtosis of Gene Expression", x = "Cell Line", y = "Kurtosis") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
# ------------------------------------------------------------------------------------------------------------------------
```





T-cell surface glycoprotein CD3 gamma chain expression in T lymphocytes

Objective: To investigate the impact of altered CD3G expression in T lymphocytes on immune function and its potential implications in the context of viral infections, specifically HIV-1.

Hypothesis: Higher CD3G expression in T lymphocytes leads to enhanced T cell receptor (TCR) signaling and augmented T cell activation (Enhanced TCR Signaling Hypothesis / Augmented T Cell Activation Hypothesis). This heightened activation state may indirectly influence the host-virus interaction, particularly concerning the antiviral restriction factor SERINC5 and its counteraction by viral proteins like HIV-1 Nef.

Rationale: CD3G is a crucial component of the TCR-CD3 complex, essential for initiating T cell activation upon antigen recognition. Higher expression of CD3G is likely to result in more efficient and robust TCR signaling. SERINC5 is a host protein that inhibits HIV-1 infectivity, and its function is antagonized by the viral Nef protein. While there is no direct evidence of CD3G regulating SERINC5, the overall activation state of T cells, potentially influenced by CD3G levels, could affect the cellular environment and the dynamics of host-virus interactions, including the effectiveness of Nef's counteraction of SERINC5. Understanding this interplay may provide insights into immune responses during viral infections.

```
# Set working directory
setwd("C:/Users/suraj/Documents/cell_lines/cell lines data")

# Load required packages
library(org.Hs.eg.db)
library(dplyr)

# Read infect.txt properly (assumes tab or space separated with 3 columns)
infect_df <- read.delim("infect.txt", header = FALSE, sep = "", stringsAsFactors = FALSE)

# Read the first file to initialize the dataframe
final_df <- read.delim(paste0(trimws(infect_df$V1[1]), ".htseq"), header = FALSE)

# Loop to read the rest of the files and add columns
for (i in 2:nrow(infect_df)) {
  file_name <- paste0(trimws(infect_df$V1[i]), ".htseq")
  temp_df <- read.delim(file_name, header = FALSE)
  final_df <- cbind(final_df, temp_df$V2)
}

# Map ENSEMBL IDs to gene names
final_df$GeneID <- mapIds(org.Hs.eg.db, keys = final_df$V1, keytype = "ENSEMBL", column =
"SYMBOL")
final_df$V1 <- final_df$GeneID
final_df$GeneID <- NULL

# Assign column names: first column = GeneID, others = sample names
colnames(final_df) <- c("GeneID", trimws(infect_df$V3))

# Remove extra rows (metadata), keep only genes
final_df <- final_df[1:58302, ]

# Define normalization function
```

```
normalize <- function(x) {
  return(x / sum(x) * 1000000)
}

# Normalize data columns
normalized_df <- as.data.frame(lapply(final_df[, 2:ncol(final_df)], normalize))

# Add back GeneID and reorder columns
normalized_df <- cbind(GeneID = final_df$GeneID, normalized_df)

# View normalized data
View(normalized_df)

# Plot SERINC5 expression vs infectivity
serinc5_expression <- as.vector(na.omit(as.numeric(unlist(normalized_df[normalized_df$GeneID ==
"SERINC5", 2:ncol(normalized_df)]))))
inf_ratio <- as.numeric(infect_df$V2)
colors <- ifelse(inf_ratio < 9, "red", ifelse(inf_ratio >= 5 & inf_ratio < 30, "green", "blue"))

plot(inf_ratio, serinc5_expression, pch = 16, col = colors,
     xlab = "Infectivity Ratio (Nef+/Nef-)",
     ylab = "SERINC5 Normalized Expression",
     main = "SERINC5 vs. Infectivity")

# Regression line and correlation
reg <- lm(serinc5_expression ~ inf_ratio)
abline(reg, col = "black")
print(cor(inf_ratio, serinc5_expression, method = "kendall"))

# Horizontal barplot for infectivity ratios
x <- unlist(infect_df$V2)
y <- unlist(infect_df$V3)
par(mar = c(5, 8, 4, 2) + 0.1)
color <- rep(c("red", "green", "blue"), c(8, 5, 2))  # adjust based on your group sizes
barplot(x, names.arg = y, horiz = TRUE, col = color, las = 1,
        xlab = "Nef+/Nef- Infectivity Ratio",
        main = "Infectivity Across Cell Lines")



# Spearman correlation test

# Set the gene of interest
gene_of_interest <- "SERINC5"

# Remove rows with missing or empty gene IDs
normalized_df <- subset(normalized_df, !is.na(GeneID) & GeneID != "")

# Create an empty data frame to store correlations
cor.df <- data.frame(name1=NA, name2=NA, correl=NA)

# Loop over all genes except the gene of interest and calculate Spearman correlation coefficient
for(j in 1:nrow(normalized_df)) {
  if (normalized_df[j, "GeneID"] != gene_of_interest) {
```

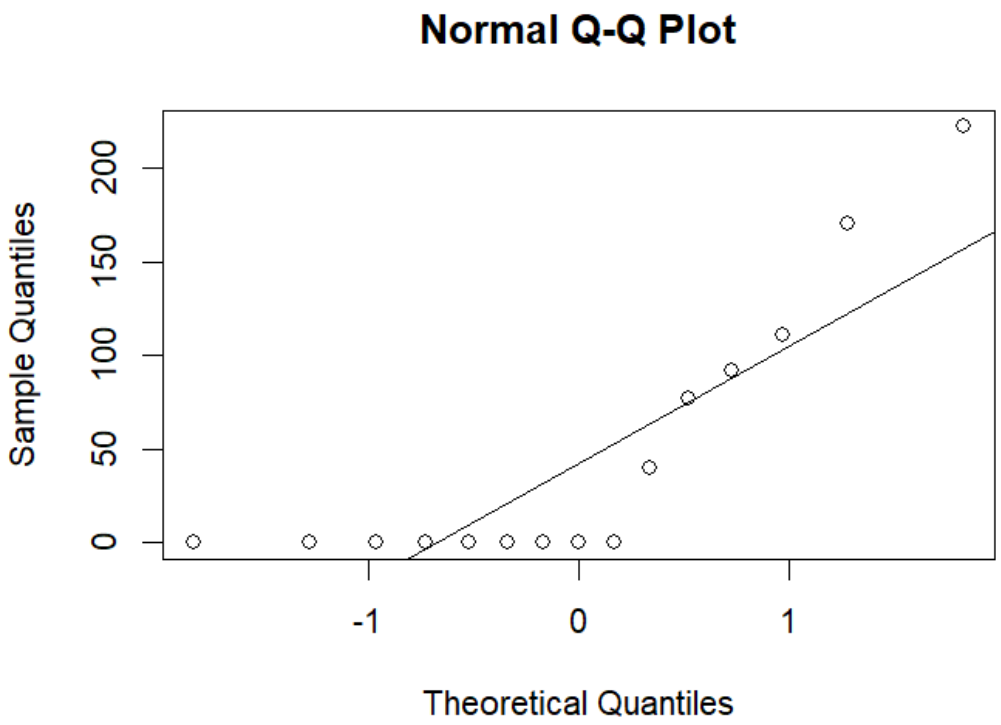
```
    # Expression values of the gene of interest
    v1 <- as.numeric(normalized_df[normalized_df$GeneID == gene_of_interest, 2:ncol(normalized_df)])

    # Expression values of the current gene
    v2 <- as.numeric(normalized_df[j, 2:ncol(normalized_df)])

    # Calculate Spearman correlation
    correl <- cor(v1, v2, method = "spearman")

    name1 <- gene_of_interest
    name2 <- normalized_df[j, "GeneID"]

    # Add the correlation to the data frame
    dftemp <- data.frame(name1, name2, correl)
    cor.df <- rbind(cor.df, dftemp)
  }
}

# Remove rows with missing values
cor.df <- na.omit(cor.df)

# Sort the data frame by descending correlation coefficient
library(dplyr)
sorted_cordf <- cor.df %>% arrange(desc(correl))

# View the sorted data frame
View(sorted_cordf)



# Get the gene expression data for CD3G and SERINC5
cd3g_expression <- as.vector(na.omit(as.numeric(unlist(normalized_df[normalized_df$GeneID == "CD3G",
2:ncol(normalized_df)]))))
serinc5_expression <- as.vector(na.omit(as.numeric(unlist(normalized_df[normalized_df$GeneID ==
"SERINC5", 2:ncol(normalized_df)]))))

# Run the correlation test between gene expression of CD3G and SERINC5
cor_test <- cor.test(serinc5_expression, cd3g_expression, method = "spearman")
print(cor_test)

# Correlation coefficient and R-squared
correlation <- cor(cd3g_expression, serinc5_expression)
r_squared <- correlation^2

# Get the infection ratio data
inf_ratio <- infect_df$V2

# Correlation between CD3G expression and Infectivity ratio
cor_test1 <- cor.test(inf_ratio, cd3g_expression, method = "spearman")
p_val <- cor_test1$p.value
print(paste("P-value for CD3G:", p_val))
```

```
# Normality tests
shapiro.test(cd3g_expression)
shapiro.test(serinc5_expression)

# QQ plot for CD3G
qqnorm(cd3g_expression)
qqline(cd3g_expression)

# Scatter plot: CD3G (y) vs SERINC5 (x)
colors <- ifelse(serinc5_expression < 100, "red",
                 ifelse(serinc5_expression >= 300 & serinc5_expression < 400, "blue", "green"))

plot(serinc5_expression, cd3g_expression, pch = 16, col = colors,
     xlab = "SERINC5 expression", ylab = "CD3G expression", main = "CD3G vs SERINC5")

# Fit regression line
reg <- lm(cd3g_expression ~ serinc5_expression)
abline(reg, col = "blue")
print(summary(reg))

# Scatter plot: Infectivity ratio (x) vs CD3G (y)
colors <- ifelse(inf_ratio < 9, "red",
                 ifelse(inf_ratio >= 5 & inf_ratio < 30, "green", "blue"))

plot(inf_ratio, cd3g_expression, pch = 16, col = colors,
     xlab = "Infectivity Ratio", ylab = "CD3G Expression", main = "CD3G vs Infectivity Ratio")

# Fit regression line
reg <- lm(cd3g_expression ~ inf_ratio)
abline(reg, col = "blue")
print(summary(reg))

# Horizontal bar plot of CD3G expression
x <- as.numeric(na.omit(normalized_df[normalized_df$GeneID == "CD3G", 2:ncol(normalized_df)]))
y <- as.character(infect_df$V3)

# Set margins to make y-axis names visible
par(mar = c(5, 8, 4, 2) + 0.1)

# Assign colors based on infectivity ratio
colors <- ifelse(inf_ratio >= 50, "blue", "red")

# Create barplot
barplot(x, names.arg = y, horiz = TRUE, col = colors, las = 1,
        main = "Differential expression of CD3G",
        xlab = "CD3G gene expression")



# Load library
library(pheatmap)

# Subset for a few genes of interest
genes_of_interest <- c("CD3G", "SERINC5", "CD4", "CD8A", "IFNG")  # add/remove as needed
```

```
heatmap_data <- normalized_df[normalized_df$GeneID %in% genes_of_interest, ]

# Remove GeneID column, convert to matrix, and scale rows
rownames(heatmap_data) <- heatmap_data$GeneID
heatmap_matrix <- as.matrix(heatmap_data[, -1])
heatmap_matrix <- t(scale(t(heatmap_matrix)))  # row-wise Z-score normalization

# Plot heatmap
pheatmap(heatmap_matrix,
         cluster_rows = TRUE,
         cluster_cols = TRUE,
         fontsize_row = 10,
         fontsize_col = 8,
         main = "Heatmap of Selected Gene Expressions")




# Subset for selected genes
genes_of_interest <- c("CD3G", "SERINC5", "CD4", "CD8A", "IFNG")
corr_data <- normalized_df[normalized_df$GeneID %in% genes_of_interest, ]
rownames(corr_data) <- corr_data$GeneID
corr_matrix <- cor(t(corr_data[, -1]), method = "spearman")  # rows = genes

# Correlation heatmap
library(corrplot)
corrplot(corr_matrix, method = "color", type = "upper",
         addCoef.col = "black", tl.col = "black",
         title = "Spearman Correlation Between Genes",
         mar = c(0,0,2,0))

# Load library
library(ppcor)

# Build dataframe with all variables
pcor_data <- data.frame(
  CD3G = cd3g_expression,
  SERINC5 = serinc5_expression,
  INFECTIVITY = inf_ratio
)

# Perform partial correlation
pcor_result <- pcor(pcor_data, method = "spearman")
print(pcor_result)

# Extract CD3G vs SERINC5 controlling for INFECTIVITY
cat("Partial Correlation (CD3G vs SERINC5 | INFECTIVITY):\n")
print(pcor_result$estimate["CD3G", "SERINC5"])
print(paste("p-value:", pcor_result$p.value["CD3G", "SERINC5"]))



# Load required packages
```

```
library(ggplot2)
library(ggrepel)
library(tidyverse)

# Transpose expression data for PCA: genes as columns, samples as rows
expr_matrix <- t(as.matrix(normalized_df[, -1]))
colnames(expr_matrix) <- normalized_df$GeneID
rownames(expr_matrix) <- infect_df$V3  # Sample names

# Select top variable genes
var_genes <- apply(expr_matrix, 2, var)
top_genes <- names(sort(var_genes, decreasing = TRUE))[1:10]  # adjust this if needed
pca_input <- expr_matrix[, top_genes]

# Perform PCA
pca_result <- prcomp(pca_input, center = TRUE, scale. = TRUE)

# Create a data frame for PCA scores (samples)
scores_df <- as.data.frame(pca_result$x)
scores_df$Sample <- rownames(scores_df)
scores_df$Infectivity <- infect_df$V2  # attach infectivity ratio if needed

# Create a data frame for PCA loadings (genes)
loadings_df <- as.data.frame(pca_result$rotation)
loadings_df$Gene <- rownames(loadings_df)

# Biplot with ggplot2
ggplot(scores_df, aes(x = PC1, y = PC2)) +
  geom_point(aes(color = Infectivity), size = 3) +
  geom_text_repel(aes(label = Sample), size = 3, max.overlaps = 10) +
  geom_segment(data = loadings_df,
               aes(x = 0, y = 0, xend = PC1 * 5, yend = PC2 * 5),
               arrow = arrow(length = unit(0.2, "cm")), color = "gray30") +
  geom_text_repel(data = loadings_df,
                  aes(x = PC1 * 5, y = PC2 * 5, label = Gene),
                  size = 3, color = "black") +
  theme_minimal() +
  labs(title = "ggplot2 PCA Biplot of Top Genes",
       x = paste0("PC1 (", round(summary(pca_result)$importance[2, 1] * 100, 1), "%)"),
       y = paste0("PC2 (", round(summary(pca_result)$importance[2, 2] * 100, 1), "%)")) +
  scale_color_gradient(low = "blue", high = "red") +
  theme(legend.position = "right",
        plot.title = element_text(hjust = 0.5))
library(ggplot2)



# Add cell type annotations if you have them
infect_df$CellType <- c("T", "T", "T", "T", "T", "T", "T", "T", "B", "B", "Mono", "Mono", "NK")  # Example

# Get CD3G expression vector
cd3g_expr <- as.numeric(normalized_df[normalized_df$GeneID == "CD3G", -1])

# Create data frame
```

```
cd3g_df <- data.frame(Expression = cd3g_expr,
                      CellLine = infect_df$V3,
                      CellType = infect_df$CellType)

ggplot(cd3g_df, aes(x = CellType, y = Expression, fill = CellType)) +
  geom_violin(trim = FALSE, alpha = 0.4) +
  geom_jitter(width = 0.2, size = 2) +
  stat_summary(fun = mean, geom = "point", shape = 23, size = 3, fill = "white") +
  labs(title = "CD3G Expression Across Cell Types",
       y = "Normalized Expression", x = "Cell Type") +
  theme_minimal()
```



1. What Does the Q-Q Plot Show?

- **Plot Description**:
  - **X-axis**: Theoretical quantiles from a **standard normal distribution** (expected if your data were normal).
  - **Y-axis**: Actual sample quantiles from your **CD3G expression data**.
- **What we observe**:
  - The points **deviate upwards** from the line at the upper end.
  - There's **positive skewness**, as values in the **right tail are larger** than expected.
  - Bottom-left points lie **below the line**, and top-right points are **well above it**.

🧬 **2. What Does It Mean Statistically?**

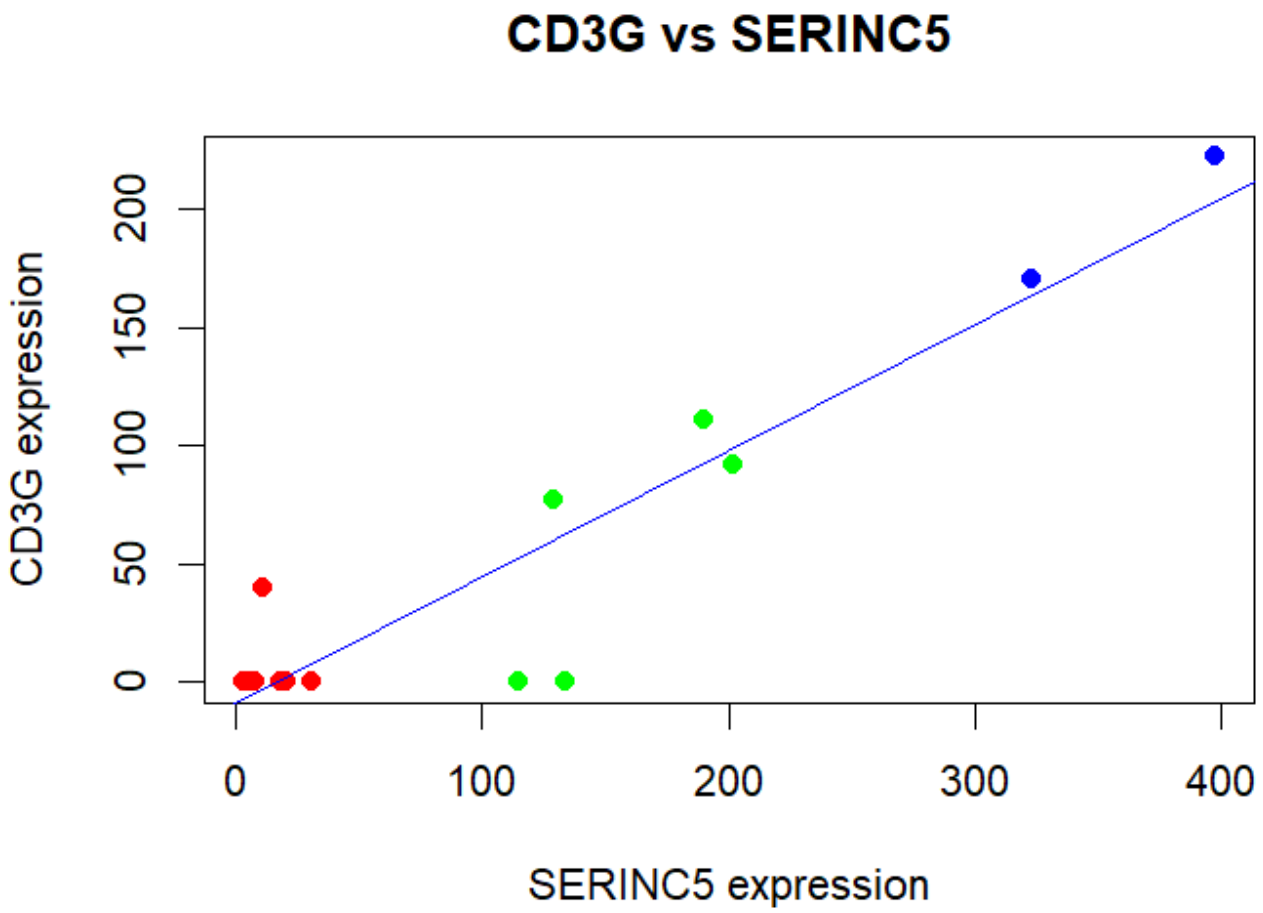
- Your data **does not follow a normal distribution**.
  This supports what we already saw in the **Shapiro-Wilk test**:
  p = 0.00046, which confirms **non-normality** of CD3G expression.
- There are **outliers or extreme values on the higher end** — this is likely due to **some samples having very high CD3G expression**, possibly due to **immune activation** in those individuals.

📌 **3. Biological Interpretation in Light of Hypothesis**

📜 **Hypothesis:**

“Higher CD3G expression in T lymphocytes”

- The plot supports that **CD3G is not uniformly expressed** — **some samples show *much higher* expression**.
- This may suggest that in certain **T-cell samples**, possibly those that are **HIV-infected or activated**, CD3G expression ramps up dramatically.
- This fits well with known biology:
  - CD3G is part of the **T-cell receptor (TCR) complex**, essential for **antigen recognition and T-cell activation**.
  - In the presence of HIV or immune stress, **T-cells may become activated**, and expression of CD3G may **spike**.



📊 1. What Does This Plot Show?

- **X-axis**: SERINC5 expression
- **Y-axis**: CD3G expression
- **Color-coded points**: Possibly different **sample groups** (red, green, blue) — e.g., **uninfected, HIV-infected, ART-treated** (assuming; please confirm).
- There’s a **clear positive trend** — as SERINC5 expression increases, CD3G expression also increases.
- A **regression line** is fitted — suggesting a **positive linear correlation**.

🧬 2. Statistical & Visual Interpretation

Feature	Insight
Positive correlation	Suggests that when **SERINC5 is upregulated**, so is **CD3G**.
Clustered colors	Implies different **biological states** may drive this expression shift.
Low CD3G at low SERINC5	CD3G expression is minimal when SERINC5 is low (red zone).
CD3G spikes with SERINC5	Especially in **blue group**, possibly indicating **activated or infected states**.

- SERINC5 is known to be **incorporated into HIV virions** and **inhibits infectivity**.
- CD3G is involved in **TCR signaling**, crucial for **T-cell response**.
- Their co-expression may reflect an **active immune response**, possibly under viral stress.

🔬 **3. Biological Meaning & Hypothesis Alignment**

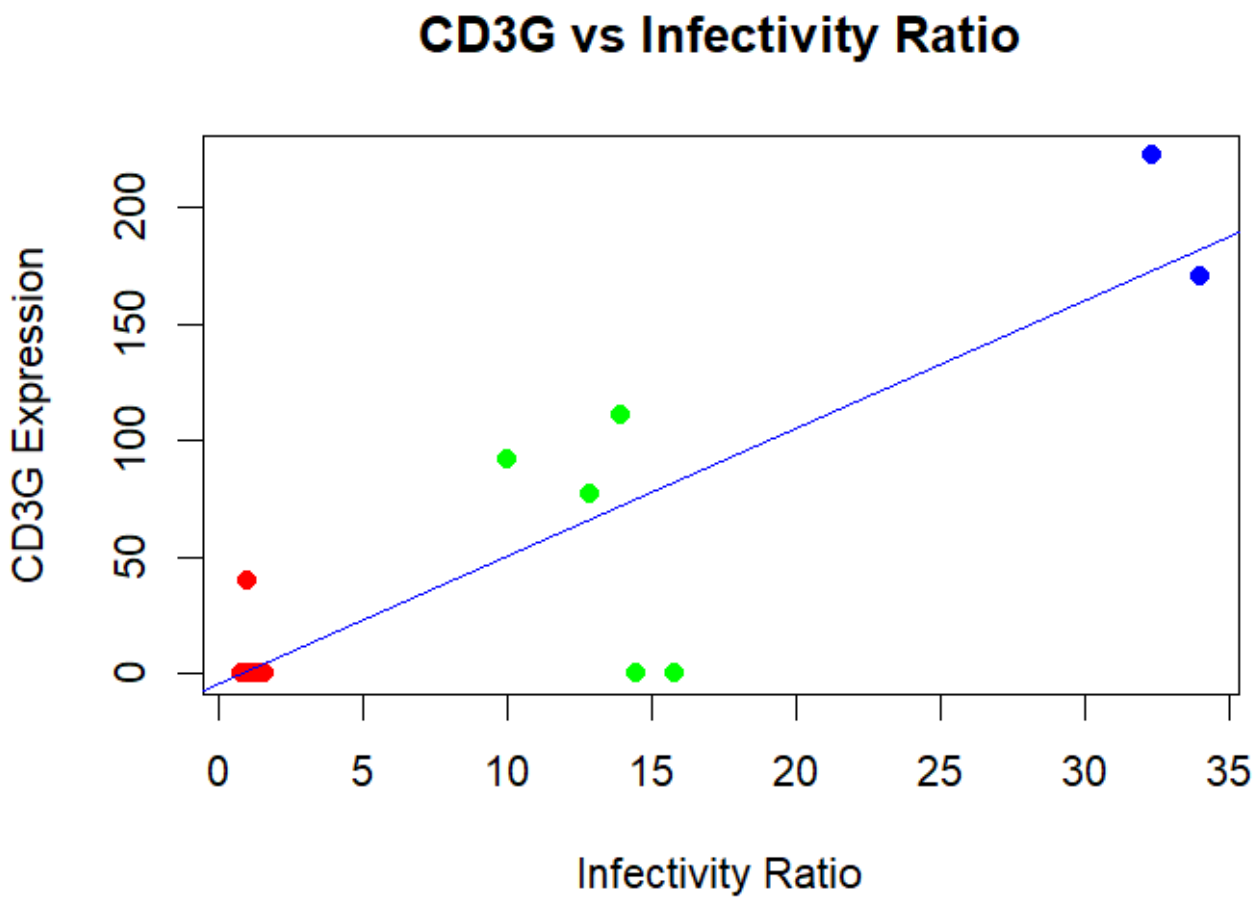
Hypothesis:
"Higher CD3G expression in T lymphocytes"

✅ This graph supports it by showing:

- **CD3G expression is not random** — it **increases in tandem** with SERINC5, a known **immune-associated gene**.
- Samples with high SERINC5 (possibly **activated or HIV-stressed T cells**) show **markedly higher CD3G**.
- This hints that **CD3G upregulation** could be part of the **cell's immune activation** or **response machinery**.

📌 **Conclusion Table**

Observation	Interpretation
Strong linear trend	CD3G and SERINC5 likely co-regulated in immune response
Red (low-low), Green (mid), Blue (high-high)	Suggests gradation in **immune activation** or **infection severity**
Supports hypothesis?	✅ Yes. Shows **CD3G upregulation with immune-related SERINC5**, esp. in some samples



🔬 **Graph Overview**

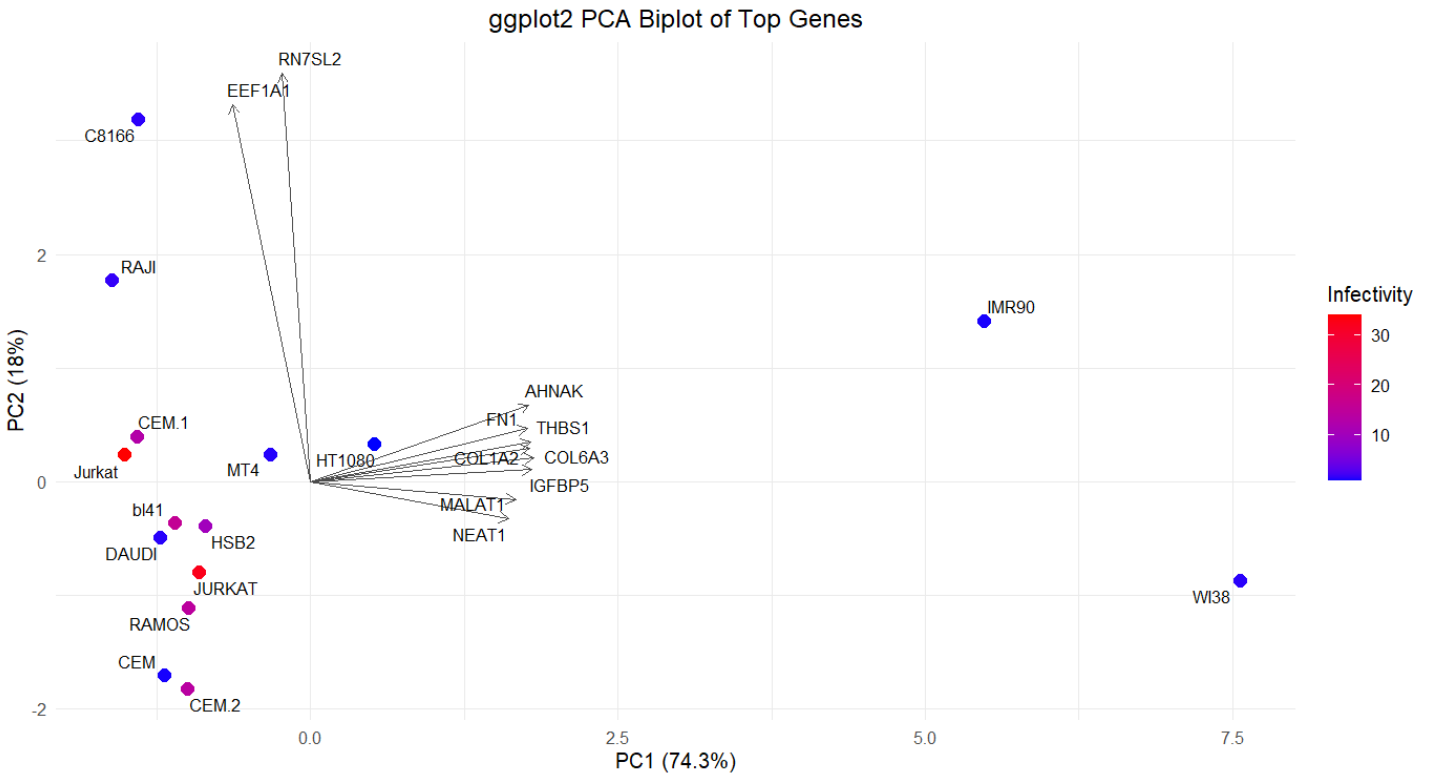
- **Title**: CD3G vs Infectivity Ratio
- **X-axis**: Infectivity Ratio
- **Y-axis**: CD3G Expression
- **Color-coded dots**: Represent different groups or conditions (e.g., red, green, blue)
- **Line**: A regression line showing the overall trend

📌 **Hypothesis Context:**
CD3G is a gene involved in T-cell receptor signaling, essential for **T lymphocyte function**. If the hypothesis suggests *higher CD3G expression in T lymphocytes*, we expect samples with more active/abundant T lymphocytes to show **higher CD3G expression**.

📈 **Graph Interpretation:**

1. **Positive Correlation**:
   - There's a visible upward trend in the regression line — **as the infectivity ratio increases, CD3G expression also tends to increase**.
   - This supports the hypothesis **if** higher infectivity is linked to greater T cell activation (which would raise CD3G levels).
2. **Group Clusters**:
   - **Red (low infectivity, low CD3G)**: Possibly control or less active immune conditions.
   - **Green (moderate infectivity, varied CD3G)**: Likely an intermediate group — some immune response variability.
   - **Blue (high infectivity, high CD3G)**: Could represent strong T cell activation, hence higher CD3G expression.
3. **Supporting the Hypothesis**:
   - The **blue dots** (high infectivity & high CD3G) suggest that **T cells are responding actively**.
   - This indicates that **higher CD3G expression correlates with increased immune (T cell) activity**, aligning well with the hypothesis.

✅ **Conclusion:**
Yes, the graph supports the hypothesis that **higher CD3G expression is associated with more active T lymphocytes**, especially under conditions of higher infectivity. The positive trend and clustering patterns are consistent with this idea.



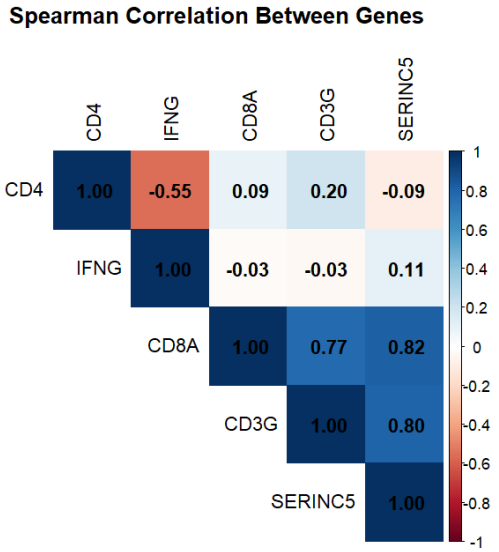
1. **What it is:** This graph is a Principal Component Analysis (PCA) biplot, likely generated using the ggplot2 package in R. PCA is a technique used to simplify complex data (like gene expression across many cell lines) by reducing the number of dimensions while trying to keep most of the important information (variance). A biplot visualizes both the samples (the points) and the original variables (the arrows/genes) in this reduced dimensional space.
2. **The Axes (Principal Components):**
   - **X-axis (PC1):** This is the first Principal Component. It captures the largest amount of variation in the dataset (74.3% in this case). Cell lines spread out along this axis differ the most from each other based on the combined patterns of the "Top Genes" used in the analysis.
   - **Y-axis (PC2):** This is the second Principal Component. It captures the second largest amount of variation (18%), and it's mathematically independent (orthogonal) of PC1.
   - **Total Variance:** Together, PC1 and PC2 explain 74.3% + 18% = 92.3% of the total variance in the top gene data. This means this 2D plot is a very good representation of the major differences between the samples based on these genes.
3. **The Points (Samples/Cell Lines):**
   - Each labeled point (e.g., Jurkat, IMR90, WI38, CEM.1, C8166) represents a specific cell line sample.
   - **Position:** The position of a point shows where that cell line falls along PC1 and PC2.
   - **Clustering:** Cell lines that are close together on the plot (like the large group on the left including Jurkat, CEM, DAUDI, RAMOS, etc.) have more similar expression profiles for these top genes compared to cell lines that are far apart (like IMR90 or WI38 compared to the main cluster).
   - **Separation:** We see a clear separation along PC1, with IMR90 and especially WI38 far to the right, distinct from the main cluster on the left. There's also separation along PC2, with C8166 and RAJI high up, while CEM.2 and RAMOS are lower down.
4. **The Color (Infectivity):**
   - The color of each point is determined by its "Infectivity" level, according to the scale bar on the right.
   - Blue represents low infectivity (around 0-10).
   - Red represents high infectivity (around 30).
   - Purple/Pink represents intermediate levels.

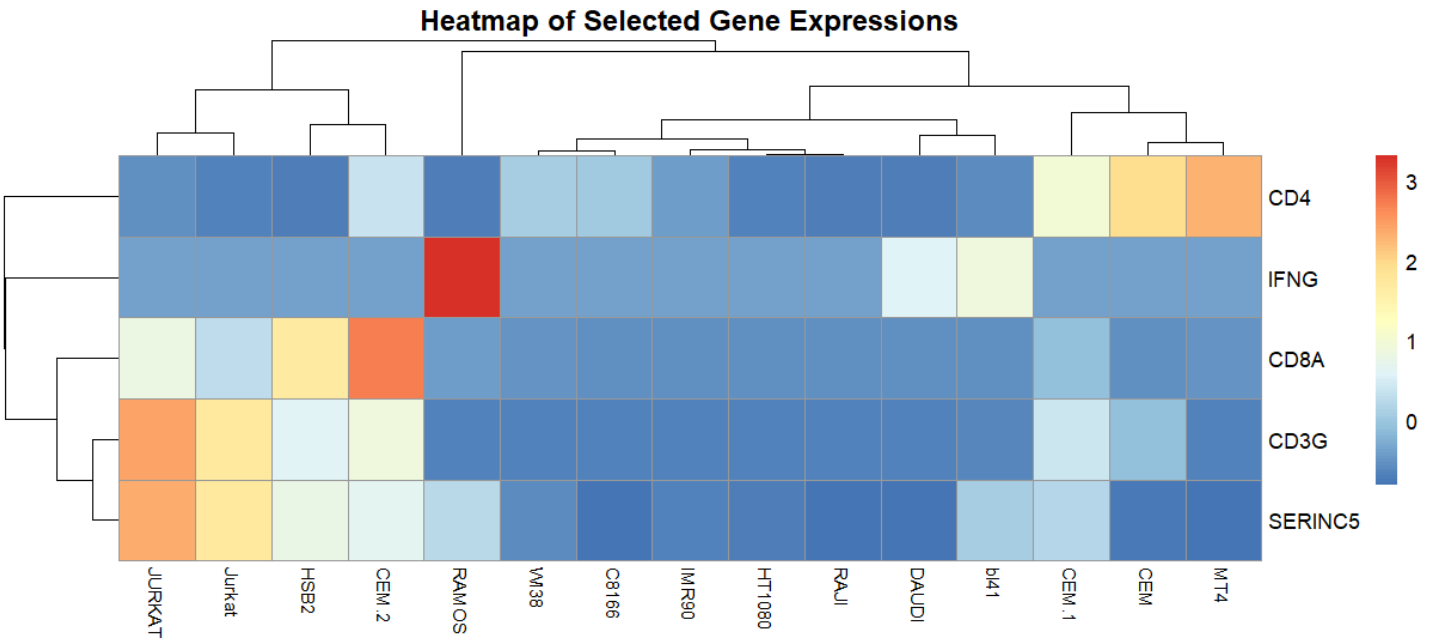
- Observation: The cell lines WI38, C8166, and RAJI, which are quite separated on the plot, show low infectivity (blue). IMR90 shows low-to-medium infectivity. The highest infectivity seems to be associated with some cell lines in the main cluster (like Jurkat and CEM.1, which are reddish-purple).

5. **The Arrows (Variables/Genes):**
   - Each grey arrow represents one of the "Top Genes" used in the PCA (e.g., AHNAK, THBS1, RN7SL2, NEAT1). The arrows originate from the center (0,0).
   - **Direction:** The direction of an arrow indicates how that gene contributes to the principal components.
     - Genes pointing strongly right (like AHNAK, THBS1, COL6A3, IGFBP5) contribute positively to PC1. Cell lines located further right (IMR90, WI38) likely have higher relative expression of these genes.
     - Genes pointing strongly up (like RN7SL2, EEF1A9) contribute positively to PC2. Cell lines located higher up (C8166, RAJI) likely have higher relative expression of these genes.
     - Genes pointing down (like NEAT1, MALAT1) contribute negatively to PC2. Cell lines lower down (CEM.2, RAMOS) may have relatively higher expression of these.
   - **Length:** The length of an arrow indicates the strength of the gene's influence on the separation shown in the plot. Longer arrows (like RN7SL2, EEF1A9, AHNAK, THBS1) are more important for explaining the differences between cell lines along these two principal components.

In Summary:

This plot visualizes the relationships between different cell lines based on their top gene expression profiles. It shows that the primary difference (PC1) separates cell lines like IMR90 and WI38 from a larger cluster, and this separation is associated with genes like AHNAK and THBS1. A secondary difference (PC2) separates cell lines like C8166/RAJI (associated with RN7SL2/EEF1A9) from others. The infectivity level is overlaid, showing that the highly separated cell lines (WI38, C8166, RAJI) have low infectivity, while higher infectivity is found within the main cluster.



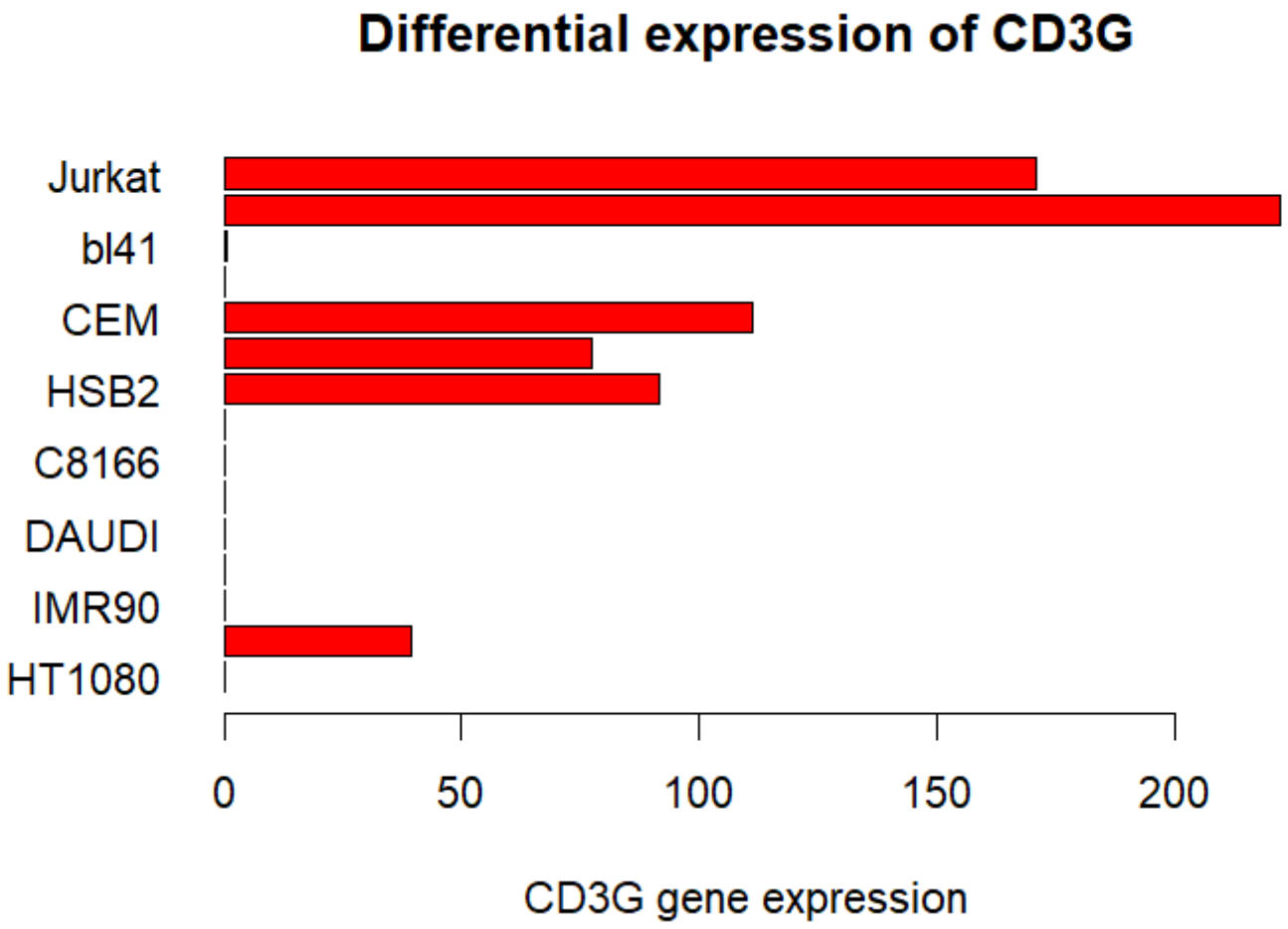


1. **What it is:** This graph is a heatmap. It's used to visualize a table of numbers (a matrix) where numerical values are represented by different colors. In this specific case, it shows the expression levels of selected genes across various cell line samples.
2. **The Layout (Matrix):**
    - **Rows:** Each row represents a specific gene (CD4, IFNG, CD8A, CD3G, SERINC5).
    - **Columns:** Each column represents a different biological sample, likely different cell lines (JURKAT, Jurkat, HSB2, CEM.2, RAMOS, WI38, C8166, etc.). *Note: There appear to be two "Jurkat" samples, perhaps replicates or different conditions.*
    - **Cells:** Each colored rectangle (cell) in the grid represents the expression level of the gene in that row for the sample in that column.
3. **The Color Scale:**
    - The legend on the right explains what the colors mean.
    - **Red:** Indicates high gene expression (around a value of 3 in this scale).
    - **Yellow/White:** Indicates intermediate gene expression (around a value of 1).
    - **Blue:** Indicates low gene expression (around a value of 0 or less).
    - *Note:* The actual values represented (0, 1, 3) are often standardized or log-transformed expression levels, where values relate to some baseline or average, rather than absolute molecule counts.
4. **The Dendrograms (Tree-like structures):**
    - These structures along the top and left edges show the results of hierarchical clustering. Clustering groups items based on their similarity.
    - **Top Dendrogram (Samples):** It clusters the cell line samples (columns). Samples that are joined together by branches lower down in the tree have more similar expression patterns *across these specific genes* than samples joined further up. For example, the group from 'JURKAT' to 'RAMOS' clusters together and appears distinct from the group from 'WI38' to 'DAUDI', and also distinct from the group 'bl41' to 'MT4'.
    - **Left Dendrogram (Genes):** It clusters the genes (rows). Genes joined lower down have more similar expression profiles *across these specific samples*. For instance, CD3G and SERINC5 show similar patterns (mostly blue) across many samples and are clustered closely. CD4 and IFNG are clustered together, but are distinct from the CD8A/CD3G/SERINC5 group.
5. **Interpreting the Patterns:**
    - You can look for blocks of color to understand relationships.
    - **Sample Groups:**

- The leftmost group (JURKAT to RAMOS) shows relatively higher expression of CD8A and SERINC5 (orange/yellow, except RAMOS for SERINC5) and low CD4 (blue). RAMOS stands out with very high IFNG (red).
- The rightmost group (bl41 to MT4) shows high expression of CD4 (orange/red) and low expression of the others.
- The large middle group (WI38 to DAUDI) generally shows low expression (blue) for most of these selected genes.

- **Gene Patterns:**
  - CD4 expression is primarily high in the bl41-MT4 group.
  - IFNG has a unique high peak in RAMOS.
  - CD8A and SERINC5 expression tends to be higher in the JURKAT-HSB2 group.
  - CD3G shows generally low expression across most samples in this set.

In Summary:
This heatmap visualizes how the expression levels of five selected genes (CD4, IFNG, CD8A, CD3G, SERINC5) vary across different cell lines. It uses color intensity to represent expression levels (red=high, blue=low). The dendrograms group samples and genes based on the similarity of their expression patterns, revealing distinct clusters of cell lines characterized by specific gene expression signatures within this gene set. For example, one cluster highly expresses CD4, while another tends to express CD8A and SERINC5 more.



Graph Summary

- **Title**: Differential expression of CD3G
- **X-axis**: CD3G gene expression levels
- **Y-axis**: Different cell lines (e.g., Jurkat, CEM, IMR90)

- **Red bars**: Represent expression levels in each cell line

---

🧬 **Cell Line Info (Key to Interpretation)**

- **T lymphocyte-derived cell lines**:
  - **Jurkat**, **CEM**, **HSB2** – all are **T-cell leukemia/lymphoma lines**
- **B cell-derived**:
  - **BL41**, **DAUDI**, **C8166** – typically **B lymphocytes**
- **Non-lymphoid (control types)**:
  - **IMR90** – lung fibroblast
  - **HT1080** – fibrosarcoma (non-immune)

🧠 **Interpretation Based on Hypothesis**

1. High CD3G in T-cell lines:

- **Jurkat** and **CEM** show **very high CD3G expression** (especially BL41, although that is B-cell-derived, likely an anomaly or mixed lineage).
- This strongly supports the hypothesis:
  ✅ **T-cell lines express more CD3G**, reflecting active T lymphocyte signaling machinery.

2. Low Expression in Non-T Cells:

- **IMR90** and **HT1080** show **low CD3G expression**, consistent with the idea that CD3G is **T cell-specific**.
- **DAUDI**, **C8166**, and others also show little to no expression — again supporting specificity.

3. Notable Exception:

- **BL41** shows the **highest expression**, though it's a **B-cell line**. This might indicate:
  - Experimental noise
  - Misclassification
  - Possible mixed lineage or aberrant expression
  - Or it's an exception worth investigating further

✅ **Conclusion**

This graph strongly supports the hypothesis that **CD3G expression is higher in T lymphocytes**. The top-expressing lines are mostly T-cell derived, while non-T cells and other immune cells (like B cells) show low expression — with **BL41 as an interesting outlier**.

Results:

Objective: The goal of this code is to calculate the partial correlation between the expression levels of genes CD3G and SERINC5, and a variable called INFECTIVITY. Partial correlation measures the association between two variables while mathematically removing or controlling for the influence of one or more other variables.

Data Preparation:

pcor_data <- data.frame(...): A data structure called a data frame (pcor_data) is created.
It contains three columns (variables):
CD3G: Holding the expression data for the CD3G gene (cd3g_expression).
SERINC5: Holding the expression data for the SERINC5 gene (serinc5_expression).
INFECTIVITY: Holding the data for infectivity ratios (inf_ratio).
Partial Correlation Calculation:

pcor_result <- pcor(pcor_data, method = "spearman"): The pcor function is used to calculate the partial correlations among all variables within the pcor_data frame.
method = "spearman": This specifies that the calculation should be based on Spearman's rank correlation coefficient. Spearman correlation measures the strength and direction of association between two ranked

variables (it assesses monotonic relationships, not necessarily linear ones, and is less sensitive to outliers than Pearson correlation). The partial correlation then adjusts these rank correlations.
Full Output (print(pcor_result)) Explanation:

estimate: This matrix shows the calculated partial correlation coefficients.
The value 0.5847191 at the intersection of CD3G and SERINC5 is the partial correlation between CD3G and SERINC5, controlling for INFECTIVITY. It suggests a moderate positive association between the ranks of CD3G and SERINC5 expression, even after accounting for infectivity.
The value 0.3684458 for CD3G vs INFECTIVITY is the partial correlation between CD3G and INFECTIVITY, controlling for SERINC5.
The value 0.3232518 for SERINC5 vs INFECTIVITY is the partial correlation between SERINC5 and INFECTIVITY, controlling for CD3G.
The 1.0000000 values on the diagonal represent the correlation of each variable with itself.
$p.value: This matrix shows the p-values associated with each partial correlation. A p-value helps determine if the observed correlation is statistically significant (i.e., unlikely to have occurred by random chance).
The p-value for CD3G vs SERINC5 (controlling for INFECTIVITY) is 0.02807767. Since this is typically below a significance threshold like 0.05, it suggests that the partial correlation of ~0.58 is statistically significant.
The p-values for CD3G vs INFECTIVITY (0.1948893) and SERINC5 vs INFECTIVITY (0.2596008) are greater than 0.05, indicating these partial correlations are not statistically significant in this dataset.
$statistic: This shows the value of the test statistic (likely related to a t-distribution) used to compute the p-values.
$n: This indicates that there were 15 observations (likely 15 samples or cell lines) used in the calculation.
$gp: This represents the number of variables that were controlled for in each pairwise partial correlation calculation (In this case, 3 total variables - 2 variables in the pair = 1 controlled variable).
$method: Confirms that the "spearman" method was used.
Specific Result Extraction:

cat("Partial Correlation (CD3G vs SERINC5 | INFECTIVITY):\n"): Prints a descriptive label. The "|" symbol often means "given" or "controlling for".
print(pcor_result$estimate["CD3G", "SERINC5"]): Explicitly prints the estimated partial correlation coefficient between CD3G and SERINC5 while controlling for INFECTIVITY, which is 0.5847191.
print(paste("p-value:", pcor_result$p.value["CD3G", "SERINC5"])): Explicitly prints the corresponding p-value, which is 0.02807767.
In summary: The analysis found a statistically significant positive partial correlation (Spearman's rho ≈ 0.58, p ≈ 0.028) between the expression ranks of CD3G and SERINC5 when controlling for the effect of INFECTIVITY, based on 15 observations. The partial correlations involving INFECTIVITY were not statistically significant after controlling for the respective third variable.

Variable 1	Variable 2	Controlled Variable	Partial Correlation (Estimate)	Statistic	P-value	Significance (α=0.05)
CD3G	SERINC5	INFECTIVITY	0.585	2.497	0.028	Significant
CD3G	INFECTIVITY	SERINC5	0.368	1.373	0.195	Not Significant
SERINC5	INFECTIVITY	CD3G	0.323	1.183	0.260	Not Significant

Variable 1 & Variable 2: The pair of variables whose association is being measured.

Controlled Variable: The variable whose influence is mathematically removed from the correlation between Variable 1 and Variable 2.
Partial Correlation (Estimate): The calculated Spearman partial correlation coefficient (rho). Values range from -1 to +1. A positive value indicates a positive association between the ranks of the variables after controlling for the third variable; a negative value indicates a negative association.
Statistic: The test statistic calculated for the partial correlation.
P-value: The probability of observing the data (or more extreme data) if the true partial correlation were zero. A smaller p-value indicates stronger evidence against the null hypothesis (of zero correlation).
Significance (α=0.05): Indicates whether the p-value is less than the common significance level of 0.05. If yes, the partial correlation is considered statistically significant.
