Foundations of Data Science Final Project

Samira Rahman

2025-10-18

1. Data Preparation:

a. Download the Sample RNA-seq Count Matrix and associated Metadata. i. Ensure that you have both the count matrix and the metadata file available for your analysis.

```r
# Defining the file path where the data is stored
file_path <- "/Users/100sr/Dartmouth/Foundations in Data Science/Final Project/"
```

```r
## Read in the RNA-seq count matrix
counts <- read.csv(paste0(file_path,"counts.csv"), row.names = 1)

## Converting to data frame
counts <- data.frame(counts)

## Preview the first few rows of the counts data frame
head(counts[, 1:2], 5)
```

```
##                    TCGA.GM.A2DL.01A.11R.A18M.07 TCGA.AC.A2QI.01A.12R.A19W.07
## ENSG00000000003.15                         1262                         2922
## ENSG00000000005.6                           120                            4
## ENSG00000000419.13                         1535                         1779
## ENSG00000000457.14                          885                         2574
## ENSG00000000460.17                          328                          586
```

```r
## Read in the associated metadata file
metadata <- read.csv(paste0(file_path,"meta_data.csv"), row.names = 1)

## Converting to data frame
metadata <- data.frame(metadata)

## Preview the first few rows of the metadata data frame
head(metadata[, 1:3], 5)
```

```
##                                   patient           sample shortLetterCode
## TCGA-GM-A2DL-01A-11R-A18M-07 TCGA-GM-A2DL TCGA-GM-A2DL-01A              TP
## TCGA-AC-A2QI-01A-12R-A19W-07 TCGA-AC-A2QI TCGA-AC-A2QI-01A              TP
## TCGA-A8-A06R-01A-11R-A00Z-07 TCGA-A8-A06R TCGA-A8-A06R-01A              TP
## TCGA-EW-A1PD-01A-11R-A144-07 TCGA-EW-A1PD TCGA-EW-A1PD-01A              TP
## TCGA-AO-A12D-01A-11R-A115-07 TCGA-AO-A12D TCGA-AO-A12D-01A              TP
```

2. Gene Selection and Summary Statistics:

a. Select One Gene: choose a gene from the dataset that interests you.

```
# Selecting a gene (row 3) from counts data frame
gene1 <- counts[3, ]

# Converting from data frame row to numeric vector so we can calculate stats
gene1_vec <- as.numeric(gene1)

# Get the gene name (Ensemble ID) for the selected gene
gene1_name <- rownames(counts)[3]
print(paste("First selected gene is:",gene1_name))
```

```
## [1] "First selected gene is: ENSG00000000419.13"
```

b. Generate Summary Statistics: Using the count data from the selected gene, compute and report summary statistics, such as mean, median, standard deviation, minimum, and maximum.

```
## Generate full summary statistics for the selected gene1
summary(gene1_vec)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     312    1570    2052    2416    2818   17569
```

```
## Mean
Mean1 <- mean(gene1_vec)
print(paste("Mean:", Mean1))
```

```
## [1] "Mean: 2416.30544272949"
```

```
## Median
Median1 <- median(gene1_vec)
print(paste("Median:", Median1))
```

```
## [1] "Median: 2052"
```

```
## Standard Deviation
Standard_Deviation1 <- sd(gene1_vec)
print(paste("Standard Deviation:", Standard_Deviation1))
```

```
## [1] "Standard Deviation: 1459.48968895205"
```

```
## Minimum
Minimum1 <- min(gene1_vec)
print(paste("Minimum:", Minimum1))
```

```
## [1] "Minimum: 312"
## Maximum
Maximum1 <- max(gene1_vec)
print(paste("Maximum:", Maximum1))
```

```
## [1] "Maximum: 17569"
```

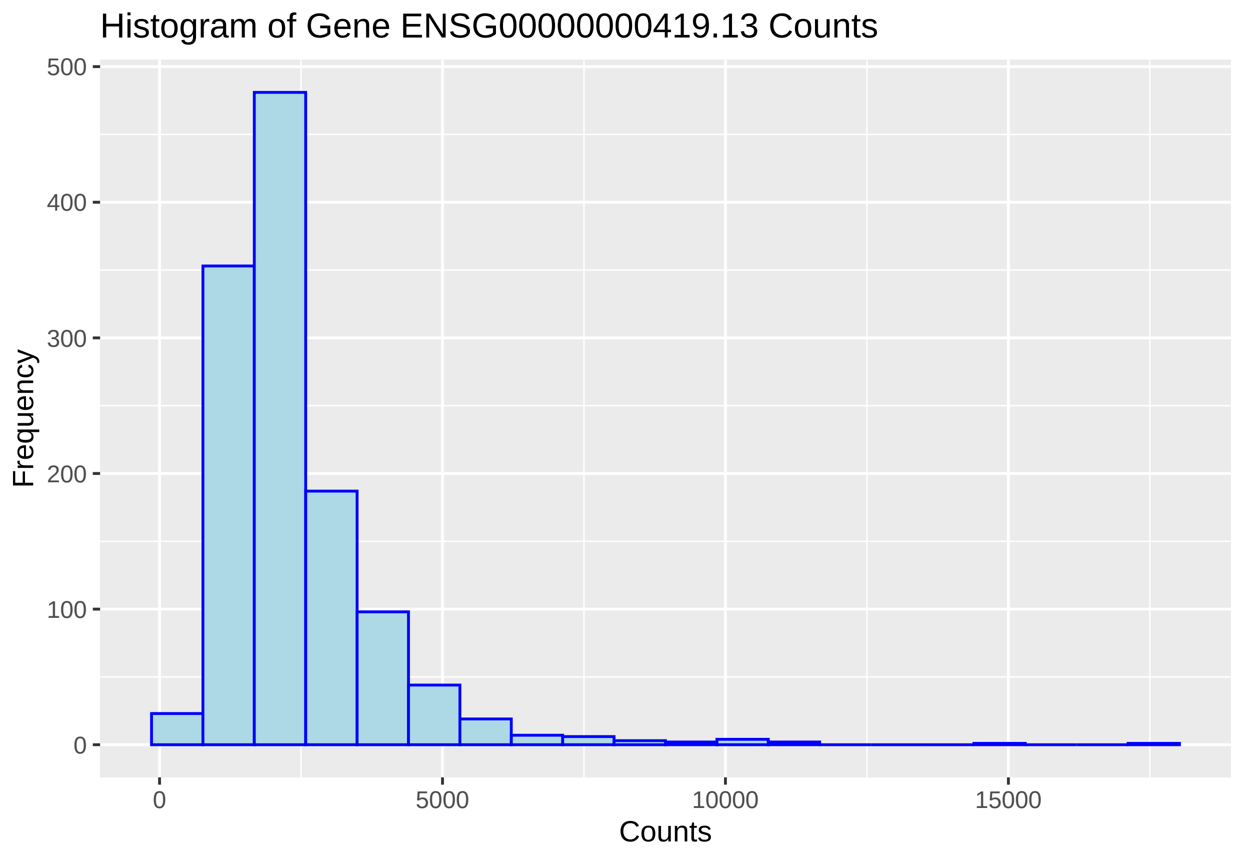
3. Visualization:

a. Create a Histogram: Use ggplot2 to generate a histogram of the count data for the selected gene. This visualization should effectively display the distribution of the counts.

```
# Create a data frame of the first selected gene counts for ggplot
gene1_df <- data.frame(Sample_ID = colnames(gene1), Count = gene1_vec)
#print(gene1_df)

# Load ggplot2 for visualization
library(ggplot2)

# Plot histogram of counts for the first selected gene
histogram <- ggplot(gene1_df, aes(x = Count)) +
  geom_histogram(fill = "lightblue", color = "blue", bins = 20) +
  labs(
    title = paste("Histogram of Gene", gene1_name, "Counts"),
    x = "Counts",
    y = "Frequency"
    )
print(histogram)
```



```r
# Save the plot
ggsave(paste0(file_path, "final_project_histogram.png"), histogram, width = 5, height = 4,dpi = 300)
```

b. Create a Scatter Plot: Select a second gene from the dataset. Create a scatter plot using ggplot2 to compare the count data of the two selected genes.

```r
# Selecting a second gene (row 5) from counts data frame
gene2 <- counts[5, ]

# Converting from data frame row to numeric vector
gene2_vec <- as.numeric(gene2)

# Get the gene name (Ensemble ID) for the selected gene
gene2_name <- rownames(counts)[5]
print(paste("Second selected gene is:", gene2_name))
```

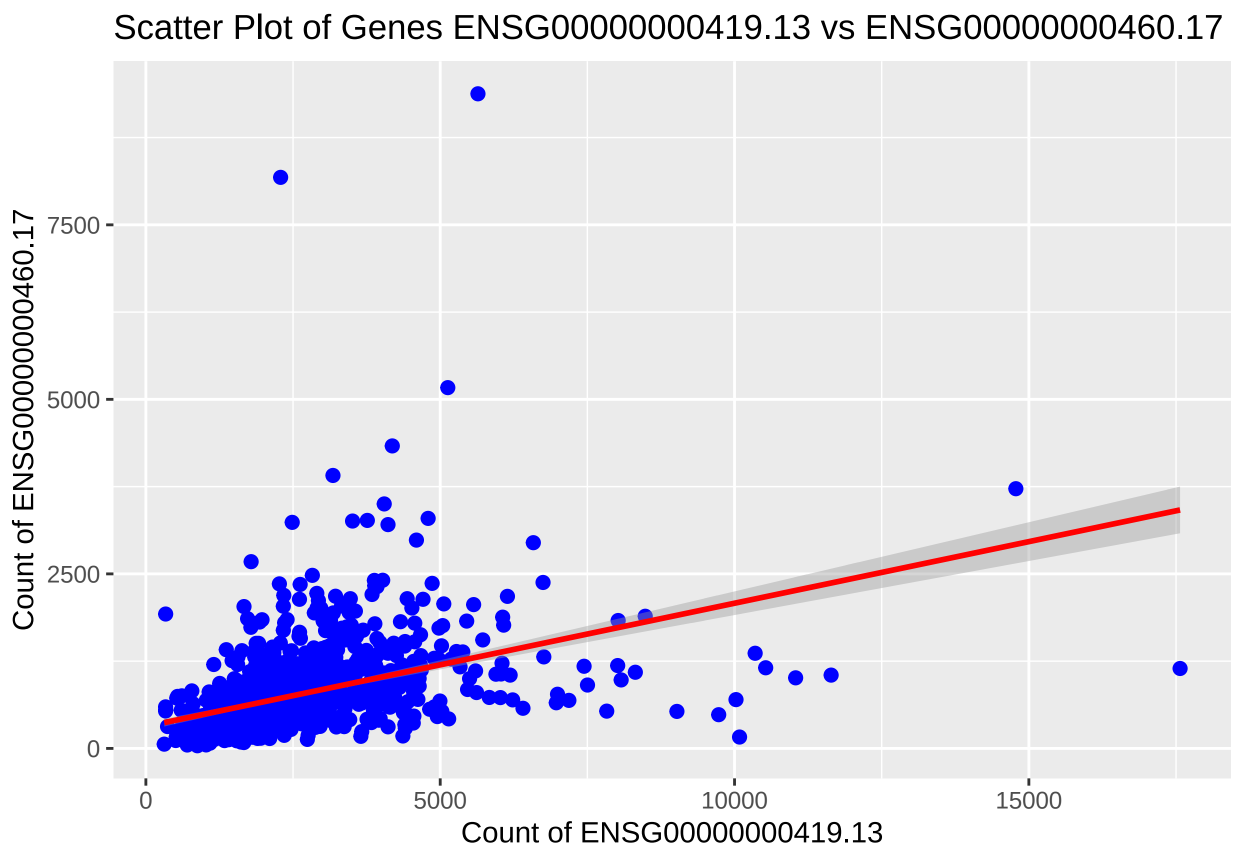
```
## [1] "Second selected gene is: ENSG00000000460.17"
```

```r
# Create a data frame of the second selected gene counts for ggplot
gene2_df <- data.frame(Sample_ID = colnames(gene2), Count = gene2_vec)
#print(gene2_df)
```

```
# Merge the two selected gene data frames by Sample ID so we can plot them together
scatter_df <- merge(gene1_df, gene2_df, by = "Sample_ID", suffixes = c(paste("", gene1_name), paste("",
#print(scatter_df)

# Scatter plot comparing expression counts of gene X vs gene Y
scatter_plot <- ggplot(scatter_df, aes(x = .data[[paste("Count", gene1_name)]], y = .data[[paste("Count
  geom_point(color = "blue", size = 2) +
  # Linear regression line with confidence interval(standard error) around the fitted line
  geom_smooth(method = "lm", se = TRUE, color = "red") +
  labs(
    x = paste("Count of", gene1_name),
    y = paste("Count of", gene2_name),
    title = paste("Scatter Plot of Genes", gene1_name, "vs", gene2_name)
  )
print(scatter_plot)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```
# Save the plot
ggsave(paste0(file_path, "final_project_scatter.png"), scatter_plot, width = 7, height = 5,dpi = 300)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

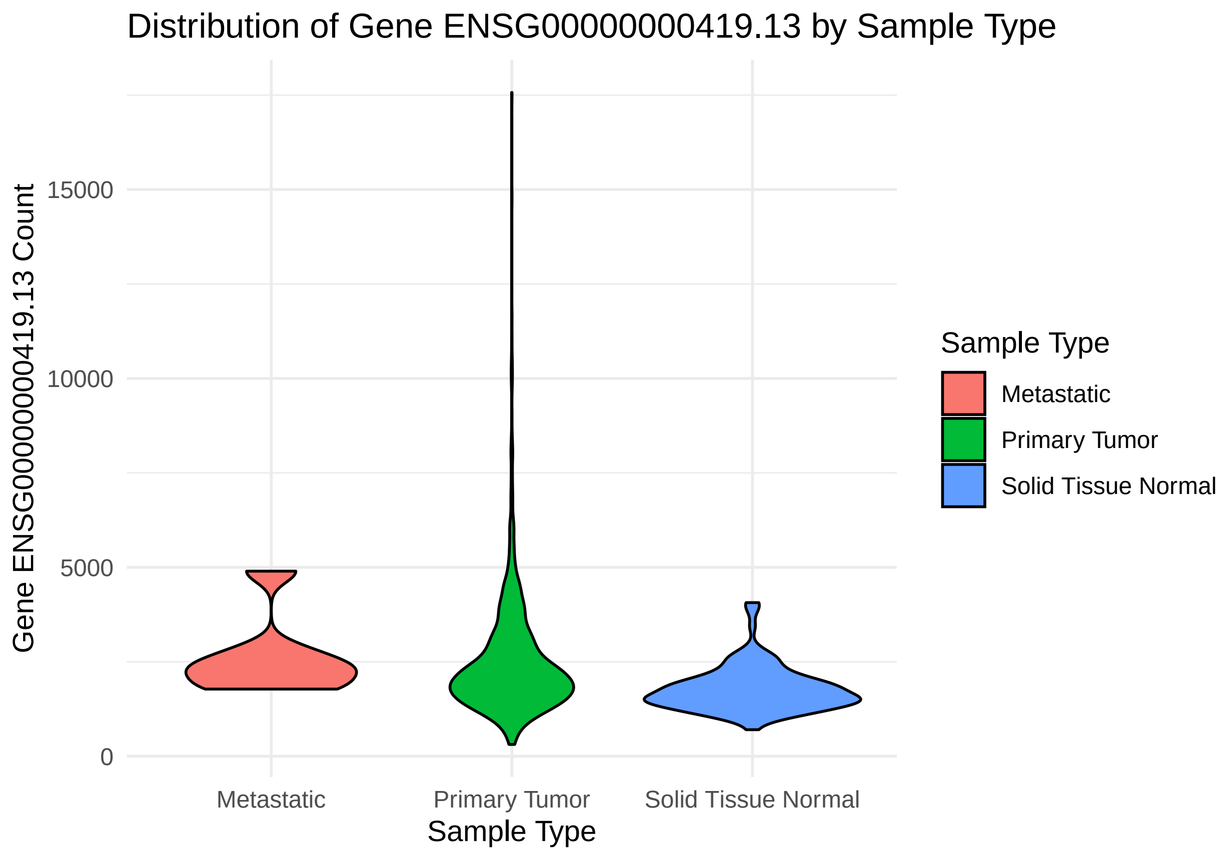
c. Create a Violin Plot: Select one covariate from your metadata. Using the count data from the first gene and the selected covariate, generate a violin plot that illustrates the distribution of count data based on the covariate. For example, if you choose "primary_diagnosis", your plot should display a violin plot for each level in "primary_diagnosis".

```
# Pick a covariate from metadata. Here we are selecting sample type
covariate <- metadata$sample_type

# Create data frame combing gene counts with the metadata covariate
violin_df <- data.frame(Count = gene1_vec, sample_type = covariate)
#print(violin_df)

# Violin plot showing distribution of gene expression across sample types
violin_plot <- ggplot(violin_df, aes(x = sample_type, y = Count, fill = sample_type)) +
  geom_violin(color = "black") +
  labs(
    fill = "Sample Type",
    x = paste("Sample Type"),
    y = paste("Gene", gene1_name, "Count"),
    title = paste("Distribution of Gene", gene1_name, "by Sample Type")
  ) + theme_minimal()

print(violin_plot)
```



```
# Save the plot
ggsave(paste0(file_path, "final_project_violin.png"), violin_plot, width = 7, height = 5,dpi = 300)
```

4. Heatmap Analysis:

a. Select 10 genes: Choose a set of 10 different genes from the count matrix for your heatmap.
b. Generate a Heatmap: Use the ComplexHeatmap package in R to create a heatmap of the count data for the selected genes.
c. Add an Annotation Bar: Include an annotation bar reflecting your chosen covariate for further context and interpretation of the data.

```
# Reading in library necessary for creating heatmaps
library(ComplexHeatmap)
```

```
## Loading required package: grid
```

```
## ========================================
## ComplexHeatmap version 2.24.1
## Bioconductor page: http://bioconductor.org/packages/ComplexHeatmap/
## Github page: https://github.com/jokergoo/ComplexHeatmap
## Documentation: http://jokergoo.github.io/ComplexHeatmap-reference
##
## If you use it in published research, please cite either one:
## - Gu, Z. Complex Heatmap Visualization. iMeta 2022.
## - Gu, Z. Complex heatmaps reveal patterns and correlations in multidimensional
##     genomic data. Bioinformatics 2016.
##
##
## The new InteractiveComplexHeatmap package can directly export static
## complex heatmaps into an interactive Shiny app with zero effort. Have a try!
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(ComplexHeatmap))
## ========================================
```

```
library(circlize)
```

```
## ========================================
## circlize version 0.4.16
## CRAN page: https://cran.r-project.org/package=circlize
## Github page: https://github.com/jokergoo/circlize
## Documentation: https://jokergoo.github.io/circlize_book/book/
##
## If you use it in published research, please cite:
## Gu, Z. circlize implements and enhances circular visualization
##   in R. Bioinformatics 2014.
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(circlize))
## ========================================
```

```
# Selecting first 10 genes from the count matrix
genes_to_plot <- rownames(counts)[1:10]

# Visualizing which 10 genes were selected
print(genes_to_plot)
```

```
##  [1] "ENSG00000000003.15" "ENSG00000000005.6"  "ENSG00000000419.13"
##  [4] "ENSG00000000457.14" "ENSG00000000460.17" "ENSG00000000938.13"
##  [7] "ENSG00000000971.16" "ENSG00000001036.14" "ENSG00000001084.13"
## [10] "ENSG00000001167.14"
```

```
# Subset the count matrix to keep only the 10 random genes
counts_subset <- counts[genes_to_plot, ]

# Heatmap annotation
col_ann <- HeatmapAnnotation(
  Sample_Type = covariate,
  col = list(Sample_Type = c("Metastatic" = "red", "Primary Tumor" = "orange", "Solid Tissue Normal" =
  annotation_legend_param = list(title = "Sample Type")
)

# Save the plot
png(paste0(file_path, "final_project_heatmap.png"), width = 2500, height = 1500, res = 300)
# Create Heatmap
Heatmap(
  counts_subset,
  name = "Counts",
  top_annotation = col_ann,
  show_row_names = TRUE,
  show_column_names = FALSE,
  cluster_rows = TRUE,
  cluster_columns = TRUE,
  column_title = "Samples",
  row_title = "Genes"
)
```

```
## Warning: The input is a data frame-like object, convert it to a matrix.
```

```
## The automatically generated colors map from the 1^st and 99^th of the
## values in the matrix. There are outliers in the matrix whose patterns
## might be hidden by this color mapping. You can manually set the color
## to `col` argument.
##
## Use `suppressMessages()` to turn off this message.
```

```
dev.off()
```

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

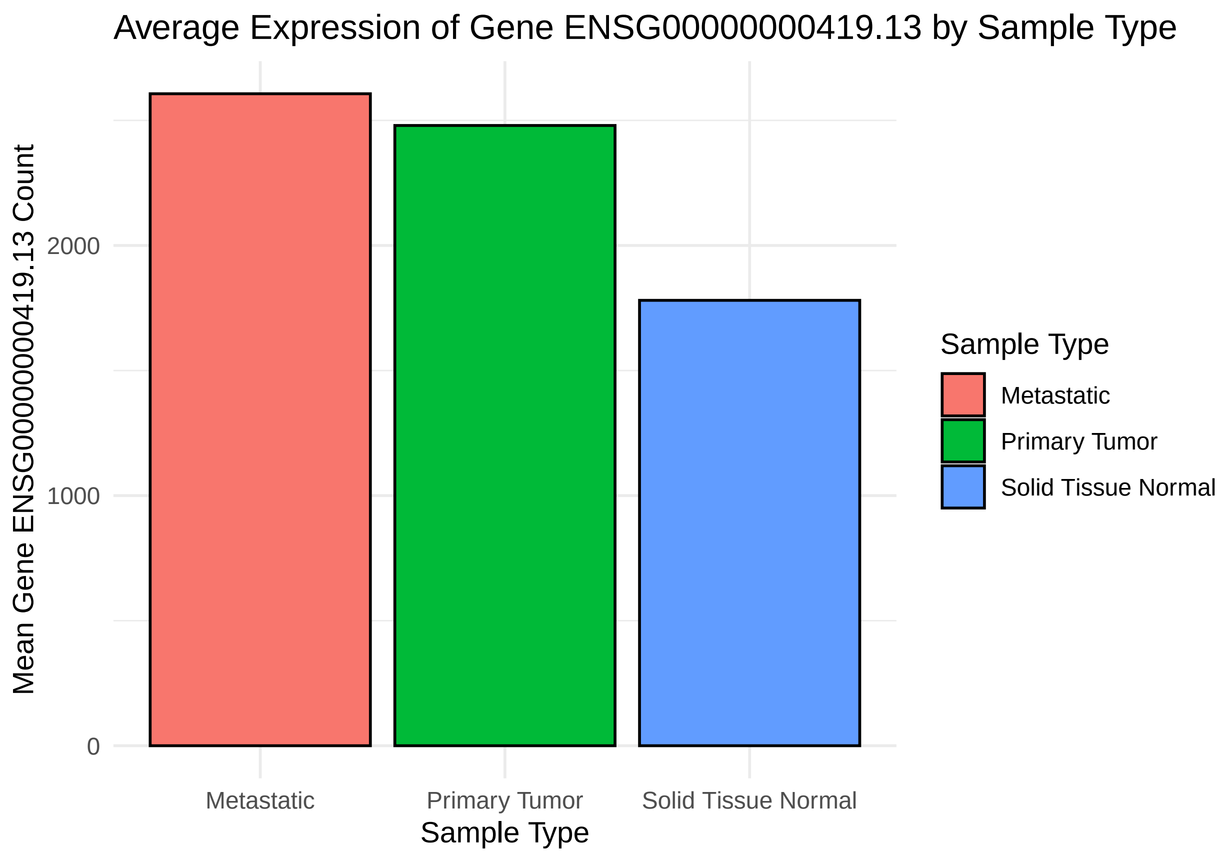
```
# New plot type: Bar chart of average gene1 expression by sample type

# Create data frame combing gene counts with the metadata covariate
bar_df <- data.frame(Count = gene1_vec, sample_type = covariate)

# Bar plot showing average gene1 expression across sample types
bar_plot <- ggplot(bar_df, aes(x = sample_type, y = Count, fill = sample_type)) +
  geom_bar(stat = "summary", fun = "mean",color = "black") +
```

```r
  labs(
    fill = "Sample Type",
    x = paste("Sample Type"),
    y = paste("Mean Gene", gene1_name, "Count"),
    title = paste("Average Expression of Gene", gene1_name, "by Sample Type")
  ) + theme_minimal()

print(bar_plot)
```



```r
# Save the plot
ggsave(paste0(file_path, "final_project_bar_plot.png"), bar_plot, width = 7, height = 5,dpi = 300)
```

```r
## Generate full summary statistics for the second gene
summary(gene2_vec)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    38.0   374.0   587.0   741.1   914.5  9377.0
```

```r
## Mean
Mean2 <- mean(gene2_vec)
print(paste("Mean:", Mean2))
```

```
## [1] "Mean: 741.095857026807"
```

```r
## Median
Median2 <- median(gene2_vec)
print(paste("Median:", Median2))
```

```
## [1] "Median: 587"
```

```r
## Standard Deviation
Standard_Deviation2 <- sd(gene2_vec)
print(paste("Standard Deviation:", Standard_Deviation2))
```

```
## [1] "Standard Deviation: 627.801147048056"
```

```r
## Minimum
Minimum2 <- min(gene2_vec)
print(paste("Minimum:", Minimum2))
```

```
## [1] "Minimum: 38"
```

```r
## Maximum
Maximum2 <- max(gene2_vec)
print(paste("Maximum:", Maximum2))
```

```
## [1] "Maximum: 9377"
```

```r
# Dataframe combining the summary stats of gene1 and gene2
summary_df <- data.frame(
  Statistics = c("Mean", "Median", "Standard Deviation", "Minimum","Maximun"),
  ENSG00000000419.13 = c(Mean1, Median1, Standard_Deviation1, Minimum1, Maximum1),
  ENSG00000000460.17 = c(Mean2, Median2, Standard_Deviation2, Minimum2, Maximum2)
)

print(summary_df)
```

```
##            Statistics ENSG00000000419.13 ENSG00000000460.17
## 1                Mean           2416.305           741.0959
## 2              Median           2052.000           587.0000
## 3 Standard Deviation           1459.490           627.8011
## 4             Minimum            312.000            38.0000
## 5             Maximun          17569.000          9377.0000
```

```r
library(xtable)
print(xtable(summary_df, caption = "Summary Statistics of Gene ENSG00000000419.13 and ENSG00000000460.1
```

```
## % latex table generated in R 4.5.1 by xtable 1.8-4 package
## % Sat Oct 18 23:48:21 2025
## \begin{table}[ht]
## \centering
## \begin{tabular}{rlrr}
##   \hline
##  & Statistics & ENSG00000000419.13 & ENSG00000000460.17 \\
##   \hline
## 1 & Mean & 2416.31 & 741.10 \\
##   2 & Median & 2052.00 & 587.00 \\
##   3 & Standard Deviation & 1459.49 & 627.80 \\
##   4 & Minimum & 312.00 & 38.00 \\
##   5 & Maximun & 17569.00 & 9377.00 \\
##    \hline
## \end{tabular}
## \caption{Summary Statistics of Gene ENSG00000000419.13 and ENSG00000000460.17}
## \end{table}
```