Final Project

2025-10-18

Instructions

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.

```
fp  <- "C:/Users/JeremyChildress/Downloads/final_project/counts.csv"



fp_meta <- "C:/Users/JeremyChildress/Downloads/final_project/meta_data.csv"
meta <- read.csv(fp_meta, header = TRUE, row.names = 1, check.names = FALSE,
                 stringsAsFactors = FALSE)
```

2. Gene Selection and Summary Statistics:

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

```
counts <- read.csv(fp, header = TRUE, row.names = 1, check.names = FALSE)

counts[] <- lapply(counts, function(x) as.numeric(as.character(x)))
class(counts)
```

```
## [1] "data.frame"
```

```
x <- as.numeric(counts[1, , drop = TRUE])
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      69    1562    2700    3208    4144   40780
```

```
length(x)
```

```
## [1] 1231
```

```r
sd(x)
```

```
## [1] 2510.336
```

```r
class(x)
```

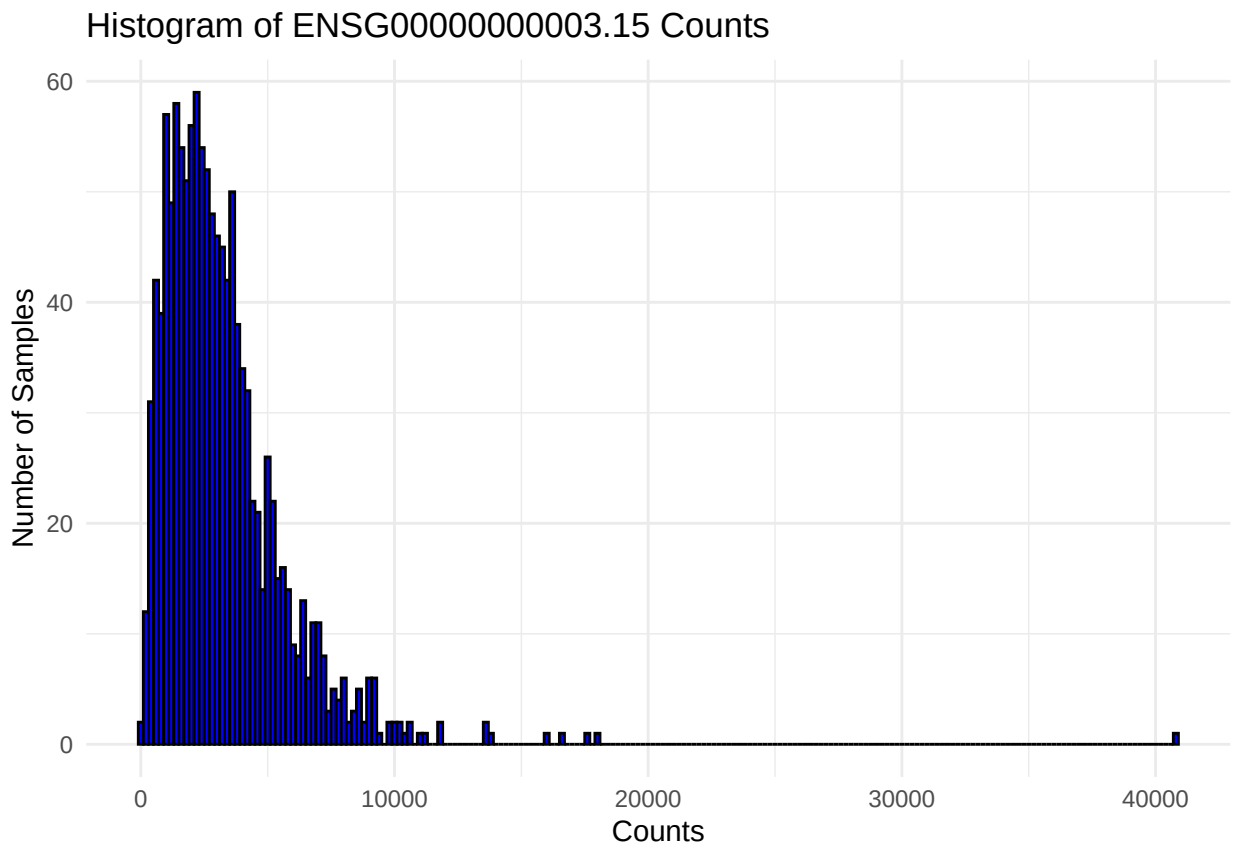
```
## [1] "numeric"
```

```r
df <- data.frame(expr = x)
```

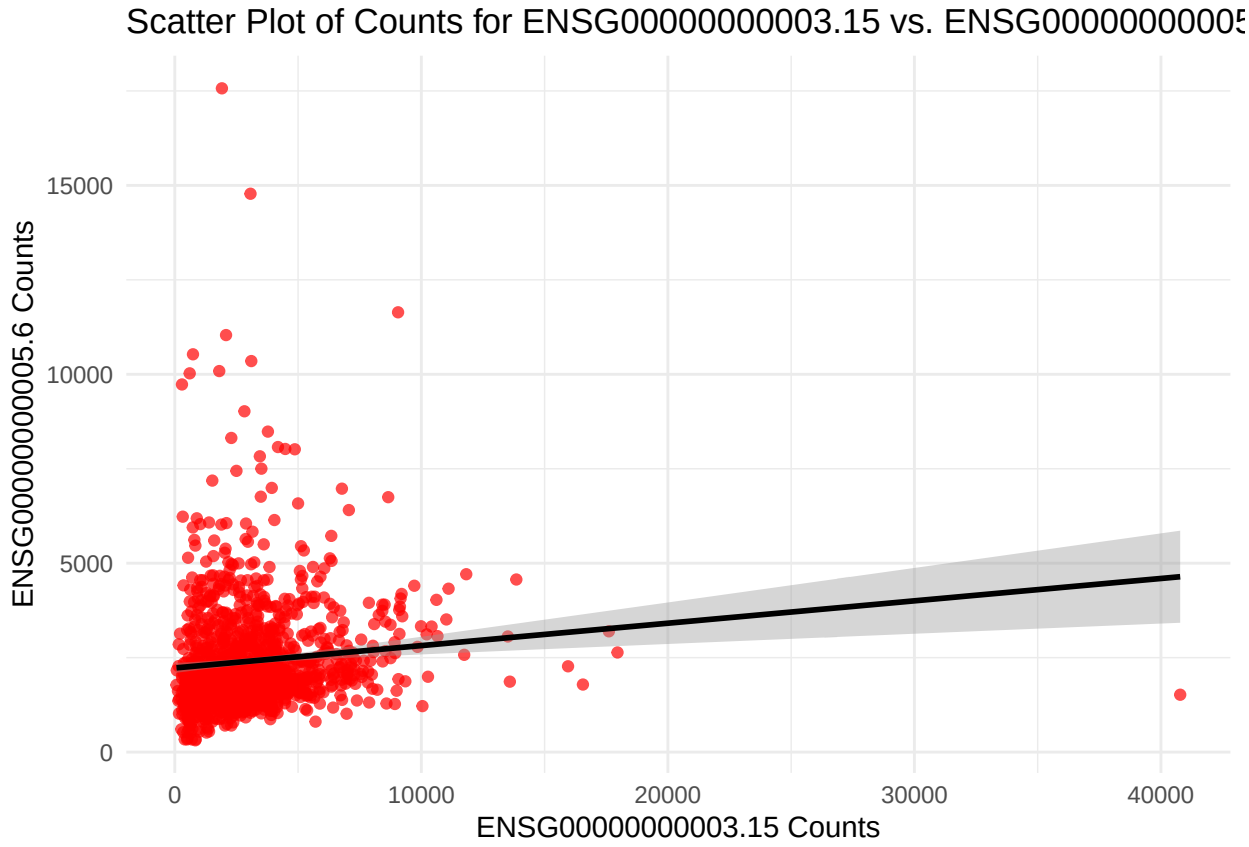
3. Visualization:

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.
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.
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".

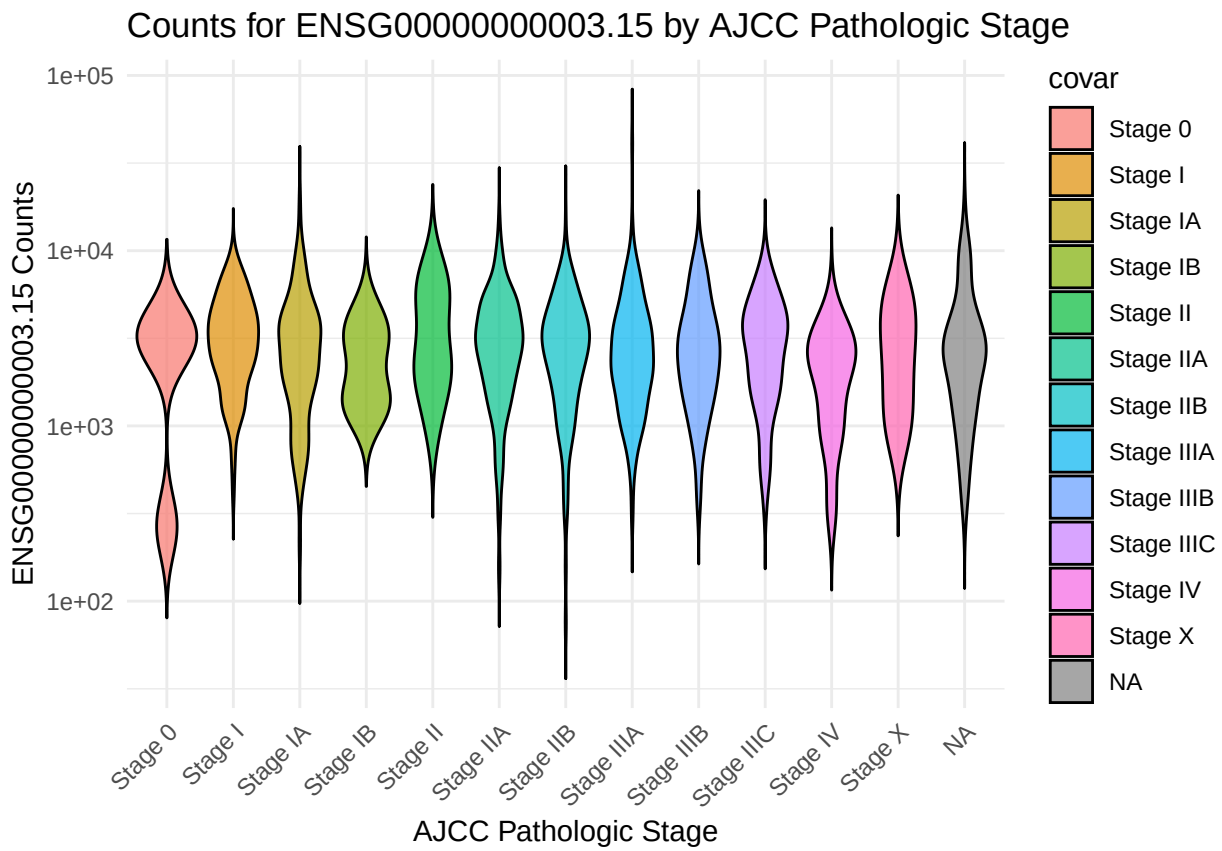
```r
library(ggplot2)
# histogram

gene_id <- rownames(counts)[1]

ggplot(df, aes(x = expr)) +
  geom_histogram(binwidth = 200, na.rm = TRUE, fill = "blue", color = "black") +
  labs(
    title = paste("Histogram of", gene_id, "Counts"),
    x = "Counts",
    y = "Number of Samples"
  ) +
  theme_minimal()
```




```
# scatter plot
gene_id_2 <- rownames(counts)[2]
z <- as.numeric(counts[3, , drop = TRUE])

ggplot(data.frame(x = x, z = z), aes(x = x, y = z)) +
  geom_point(alpha = 0.7, color = "red") +
  geom_smooth(method = "lm", formula = y ~ x, se = TRUE, color = "black") +
  labs(
    title = paste("Scatter Plot of Counts for", gene_id, "vs.", gene_id_2),
    x = paste(gene_id, "Counts"),
    y = paste(gene_id_2, "Counts")
  ) +
  theme_minimal()
```

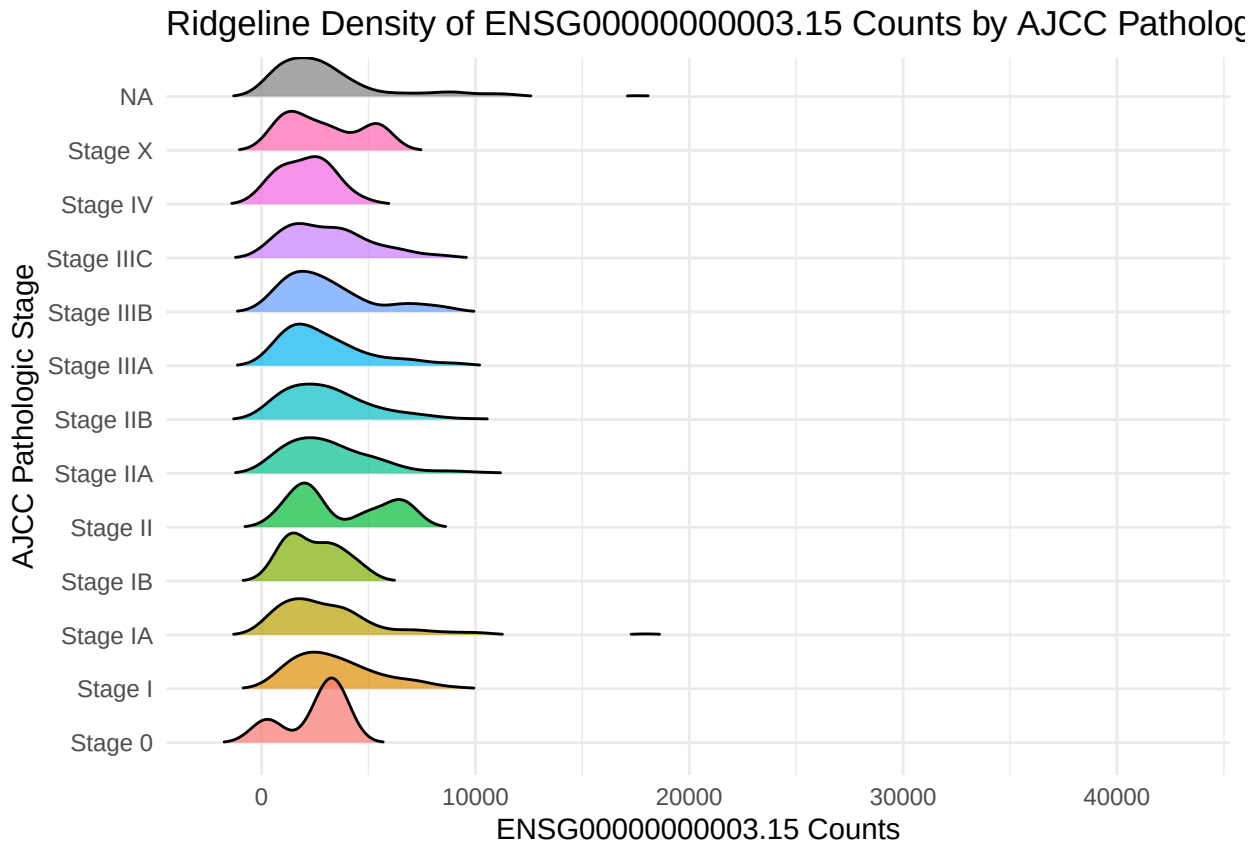



```r
# Violin plot
gene_id <- rownames(counts)[1]
expr    <- as.numeric(counts[gene_id, ])
covar   <- meta[colnames(counts), "ajcc_pathologic_stage"]

df <- data.frame(expr = expr, covar = covar)
df$covar <- factor(df$covar)

p_violin <- ggplot(df, aes(x = covar, y = expr, fill = covar)) +
  geom_violin(color = "black", alpha = 0.7, trim = FALSE) +
  scale_fill_hue() +
  labs(
    title = paste("Counts for", gene_id, "by AJCC Pathologic Stage"),
    x = "AJCC Pathologic Stage",
    y = paste(gene_id, "Counts")
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Peer suggestion: log scale for better fit; set to FALSE to disable
use_log10 <- TRUE
if (use_log10) p_violin <- p_violin + scale_y_log10()

p_violin
```




```
# Additional plot: Ridgeline density (requires ggridges)
suppressWarnings(suppressMessages(library(ggridges)))

# Reuse objects from above for consistency
df_ridge <- df  # df has expr + covar

ggplot(df_ridge, aes(x = expr, y = covar, fill = covar)) +
  ggridges::geom_density_ridges(alpha = 0.7, rel_min_height = 0.01, scale = 1.2) +
  labs(
    title = paste("Ridgeline Density of", gene_id, "Counts by AJCC Pathologic Stage"),
    x = paste(gene_id, "Counts"),
    y = "AJCC Pathologic Stage"
  ) +
  theme_minimal() +
  theme(legend.position = "none")
```

```
## Picking joint bandwidth of 754
```



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

```
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))
## ========================================
```

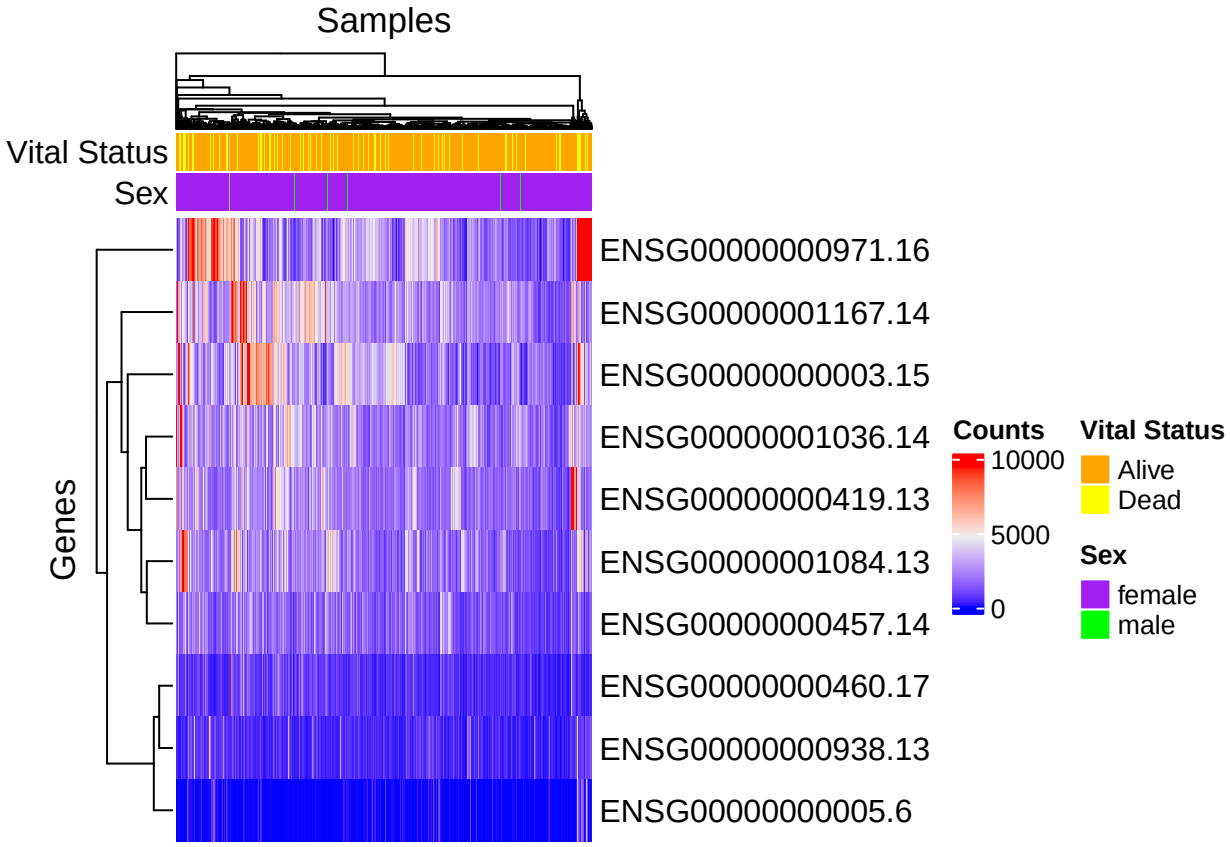
```r
library(circlize)
```

```
## ========================================
## 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))
## ========================================
```

```r
metadata <- meta

# Select 10 genes
genes_to_plot <- rownames(counts)[1:10]
counts_subset <- counts[genes_to_plot, , drop = FALSE]

# Build a data.frame for annotations, then rename columns to clean labels
ann_df <- data.frame(
  `Vital Status` = metadata$vital_status,
  Sex            = metadata$gender,
  check.names = FALSE
)

column_ha <- HeatmapAnnotation(
  df = ann_df,
  col = list(
    `Vital Status` = c(Dead = "yellow", Alive = "orange"),
    Sex            = c(male = "green", female = "purple")
  ),
  annotation_name_side = "left"
)

Heatmap(
  counts_subset,
  name = "Counts",
  top_annotation   = column_ha,
  show_row_names   = TRUE,
  show_column_names= FALSE,
  cluster_columns  = TRUE,
  cluster_rows     = TRUE,
  column_title     = "Samples",
  row_title        = "Genes",
  heatmap_legend_param = list(title = "Counts")
)
```

```
## 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.
```



4. Record a 3-Minute Video: In this video, explain your selected gene, the summary statistics you computed, and discuss each of your visualizations. See below for more info on the video.