HSE711: Final Project

Pritish Ponaka

1. Data Preparation:

a. Download the Sample RNA-seq Count Matrix and associated Metadata.

b. Ensure that you have both the count matrix and the metadata file available for your analysis.

```r
# Assign the data path to a variable
data_path <- "C:/Users/f003cc3/Documents/hds/HSE 711 Foundations of Data Scie
nce/final project/"

# Assign file names to variables to read them below
counts_file <- "counts.csv"
meta_file <- "meta_data.csv"

# Read counts data in to a data frame
RNA_seq_Count_matrix <- read.csv(paste0(data_path, counts_file))

# Read meta data into a data frame
RNA_seq_metadata <- read.csv(paste0(data_path, meta_file))

# head(RNA_seq_Count_matrix)

# tail(RNA_seq_metadata)

# Change the first column header to gene_id
colnames(RNA_seq_Count_matrix)[1] <- "Gene_id"
```

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.

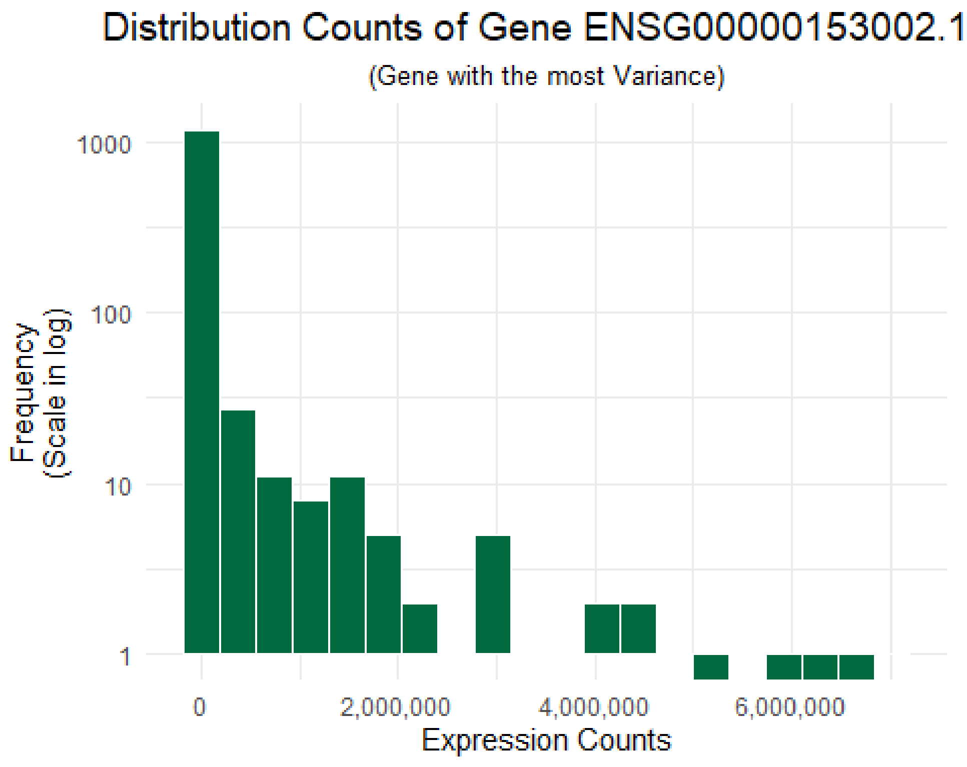
```r
# My selected Gene_id is ENSG00000153002.12.
# I would like select a gene based on variance. If the gene count has the mos
t variability that means it
# can show distinctive traits between individuals causing breast cancer. So,
I would like to select
# the gene with the most variance in the data set. To do that, I need calcula
te summary statistics
# for all the gene types in the data set.

# Check the data types of RNA_seq_Count_matrix.
# I've learned about apply, sapply() and lapply() functions a while ago.
# Reference: https://www.r-bloggers.com/2022/03/complete-tutorial-on-using-ap
ply-functions-in-r/
```

```
# They come in handy to apply a function over a data frame or a vector.
data_types <- sapply(RNA_seq_Count_matrix, class)

# Now see the data types summary using table() funtion.
table(data_types)

## data_types
## character   integer
##         1      1231

# Convert all columns except the first column(Gene_id) to numeric and save it
 as a list.
# Then create a data frame with summary stats that use numeric columns (excep
t the first column, Gene_id)
RNA_seq_Count_num <- RNA_seq_Count_matrix[, -1]
RNA_seq_Count_num[] <- lapply(RNA_seq_Count_num, as.numeric)

# Calculate summary stats for each Gene_id and store it in a data frame, summ
ary_stats.
summary_stats <- data.frame(
  Gene_id = RNA_seq_Count_matrix$Gene_id,
  Gene_mean = apply(RNA_seq_Count_num, 1, mean),
  Gene_median = apply(RNA_seq_Count_num, 1, median),
  Gene_sd = apply(RNA_seq_Count_num, 1, stats::sd),
  Gene_max = apply(RNA_seq_Count_num, 1, max),
  Gene_min = apply(RNA_seq_Count_num, 1, min)
)

# Now look for the max variance i.e., equivalently maximum standard deviation
# because variance == sd^2.

my_selected_Gene <- summary_stats$Gene_id[which.max(summary_stats$Gene_sd)]

print(my_selected_Gene)

## [1] "ENSG00000153002.12"

# summary stats of my selected gene
print(summary_stats[summary_stats$Gene_id == my_selected_Gene, ])

##                 Gene_id Gene_mean Gene_median  Gene_sd Gene_max Gene_min
## 9604 ENSG00000153002.12  97920.46         205 498573.3  7032374        0

# Save my selected gene data as a data frame by transposing columns to rows
my_df <- RNA_seq_Count_matrix[RNA_seq_Count_matrix$Gene_id == my_selected_Gen
e, ]

library(tidyr)
 my_sel_gene_df <- tidyr::pivot_longer(my_df, cols=starts_with("TCGA"))
```

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.

```
library(ggplot2)
library(scales)

hist_plot <- ggplot(data = my_sel_gene_df, aes(x = value)) +
             geom_histogram(bins=20, fill = "#00693e", color = "white") +
             scale_y_log10() +
             scale_x_continuous( labels = comma) +
             theme_minimal() +
             labs(title = paste0("Distribution Counts of Gene ", my_selected_
Gene),
                  subtitle = "(Gene with the most Variance)",
                  x = 'Expression Counts',
                  y = 'Frequency\n(Scale in log)') +
             theme(plot.title = element_text(size = 14, hjust = 0.5),
                   plot.subtitle = element_text(size=10, hjust = 0.5))


plot(hist_plot)

## Warning in scale_y_log10(): log-10 transformation introduced infinite valu
es.
```



```
# save the plot to a png
png("C:/Users/f003cc3/Documents/hds/HSE 711 Foundations of Data Science/final
 project/Histogram.png", width = 2000, height = 1500, res = 300)
plot(hist_plot)

## Warning in scale_y_log10(): log-10 transformation introduced infinite valu
es.

dev.off()

## png
##   2
```

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.

```
# My second selected gene is ENSG00000198804.2
# my second selected Gene has the maximum mean value
# Added geom_smooth lm. Added cleaned up legend based on the feedback.
my_sec_Gene <- summary_stats$Gene_id[which.max(summary_stats$Gene_mean)]

print(my_sec_Gene)

## [1] "ENSG00000198804.2"

# Create a new data frame with my two selected Genes
my_sec_df <- RNA_seq_Count_matrix[RNA_seq_Count_matrix$Gene_id %in% c(my_sele
cted_Gene, my_sec_Gene), ]

# Transpose the data
my_gene_df <- tidyr::pivot_longer(my_sec_df, cols=starts_with("TCGA"))

my_gene_df_final <- tidyr::pivot_wider(my_gene_df, names_from = Gene_id)

# Merge my selected Genes RNA count data and meta data
RNA_seq_data <- merge(my_gene_df_final, RNA_seq_metadata, by="row.names")

RNA_seq_data <- RNA_seq_data[is.na(RNA_seq_data$gender) == FALSE, ]

# Change column names to identify gene1 and gene2
colnames(RNA_seq_data)[3] <- "Gene1_value"
colnames(RNA_seq_data)[4] <- "Gene2_value"

# regression equation line using lm method. One for Male and one for Female.
equation_lm_m <- lm(Gene2_value ~ Gene1_value, RNA_seq_data[RNA_seq_data$gend
er=="male", ])
equation_lm_f <- lm(Gene2_value ~ Gene1_value, RNA_seq_data[RNA_seq_data$gend
er=="female", ])

# Tweak the regression equation format.
female_lm <- paste0("Female: y = ", round(coef(equation_lm_m)[2], 2), "x + ",
```

```r
 round(coef(equation_lm_m)[1], 2))
Male_lm <- paste0("Male: y = ", round(coef(equation_lm_f)[2], 2), "x + ", rou
nd(coef(equation_lm_f)[1], 2))

# scatter plot of gene1 vs. gene2 by gender
scat_plot <- ggplot(data = RNA_seq_data, aes(x = Gene1_value, y=Gene2_value,
color = gender)) +
             geom_point(size=2.2, alpha = 0.5) +
             scale_color_manual(values = c("#6290C3", "#5E0B15"), labels = c
("Female", "Male")) +
             scale_x_log10(labels = comma) +
             scale_y_log10(labels = comma) +
             theme_minimal() +
             geom_smooth(method = "lm", se = TRUE, size = 1.4) +          # R
egression line (linear method)
             labs(title = paste0("Gene Counts of ", my_selected_Gene, " & ",
my_sec_Gene, " by Gender"),
                  subtitle = paste0("(Gene with the most Variance vs. Gene wi
th the most Mean)\n\n", female_lm, " | ", Male_lm),
                  x = "\nGene with the most variance\n(log scale)",
                  y = "Gene with the most Mean\n(log scale)",
                  color = "Gender") +
             theme(plot.title = element_text(size = 14, hjust = 0.5),
                   plot.subtitle = element_text(size=10, hjust = 0.5),
                   legend.position = "bottom",
                    panel.grid.minor.y = element_blank(),
                    panel.grid.minor.x = element_blank())
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

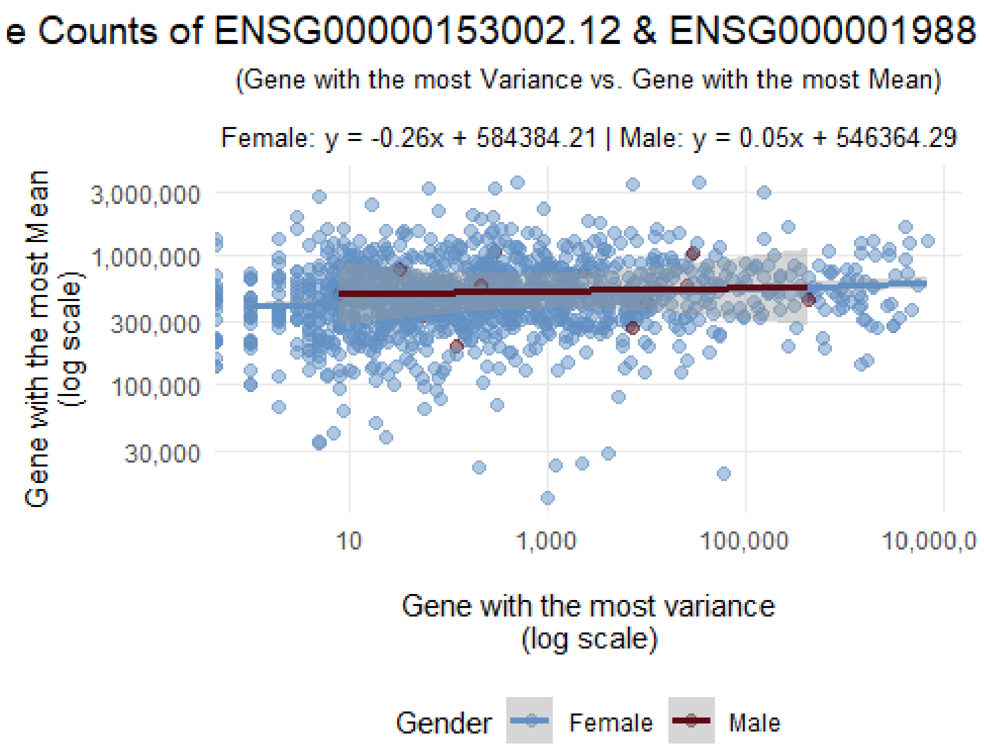
```r
plot(scat_plot)
```

```
## Warning in scale_x_log10(labels = comma): log-10 transformation introduced
## infinite values.
```

```
## Warning in scale_x_log10(labels = comma): log-10 transformation introduced
## infinite values.
```

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

```
## Warning: Removed 22 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

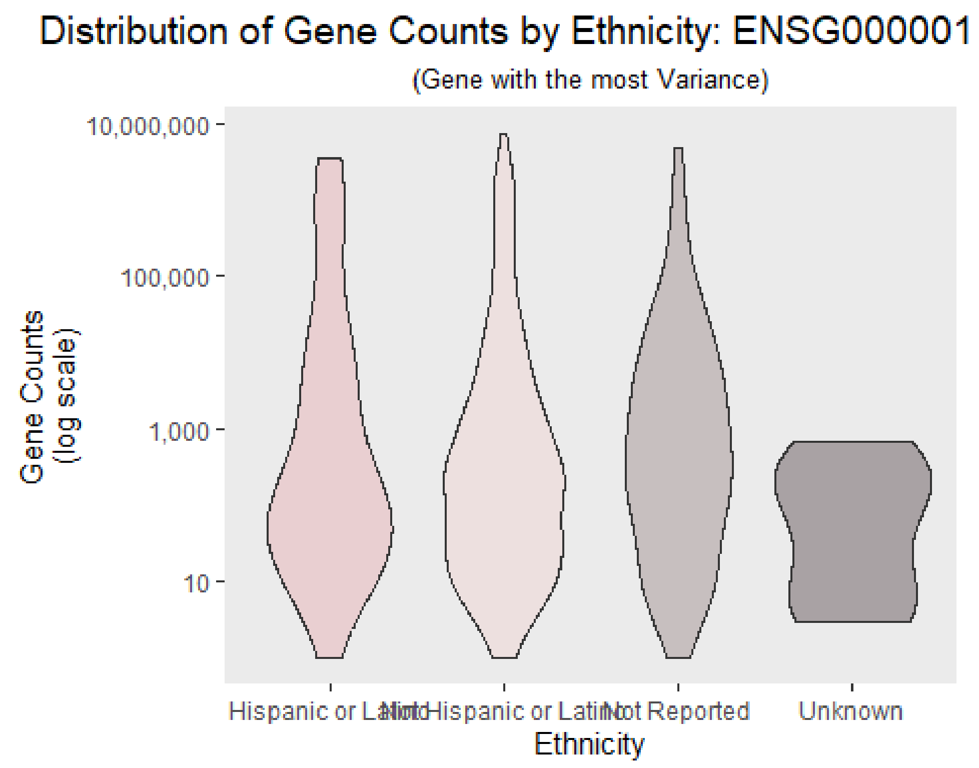


```r
# save the plot to a png
png("C:/Users/f003cc3/Documents/hds/HSE 711 Foundations of Data Science/final
 project/Scatter_plot.png", width = 2500, height = 1800, res = 300)
plot(scat_plot)

## Warning in scale_x_log10(labels = comma): log-10 transformation introduced
 infinite values.
## log-10 transformation introduced infinite values.

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

## Warning: Removed 22 rows containing non-finite outside the scale range
## (`stat_smooth()`).

dev.off()

## png
##   2
```

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
# Filter out the co-variate NA values i.e., ethnicity for a cleaner looking g
raph
RNA_seq_data <- RNA_seq_data[!is.na(RNA_seq_data$ethnicity), ]
```

```
# Use viridis package
library(viridis)

## Loading required package: viridisLite

##
## Attaching package: 'viridis'

## The following object is masked from 'package:scales':
##
##     viridis_pal

violin_plot <- ggplot(data = RNA_seq_data, aes(x = ethnicity, y=Gene1_value,
fill=ethnicity)) +
             geom_violin(alpha = 0.5) +
             scale_fill_manual(values = c("#e8b4b8", "#eed6d3", "#a49393", "#
67595e")) +
             scale_x_discrete(labels = c("hispanic or latino"="Hispanic or La
tino", "not hispanic or latino"="Not Hispanic or Latino", "not reported"="Not
 Reported",             "Unknown"="Unknown")) +
             scale_y_log10(labels = comma) +
             labs(title = paste0("Distribution of Gene Counts by Ethnicity: ",
 my_selected_Gene),
                  subtitle = "(Gene with the most Variance)",
                  x = "Ethnicity",
                  y ="Gene Counts\n(log scale)",
                  color = "Ethnicity") +
             theme(plot.title = element_text(size = 14, hjust = 0.5),
                   plot.subtitle = element_text(size=10, hjust = 0.5),
                   legend.position = "none",
                    panel.grid.minor.y = element_blank(),
                    panel.grid.minor.x = element_blank(),
                    panel.grid.major.x = element_blank(),
                    panel.grid.major.y = element_blank())


plot(violin_plot)

## Ignoring unknown labels:
## • colour : "Ethnicity"

## Warning in scale_y_log10(labels = comma): log-10 transformation introduced
## infinite values.

## Warning: Removed 22 rows containing non-finite outside the scale range
## (`stat_ydensity()`).
```



```
# save the plot to a png
png("C:/Users/f003cc3/Documents/hds/HSE 711 Foundations of Data Science/final
 project/violin_plot.png", width = 1800, height = 1500, res = 300)
plot(violin_plot)

## Ignoring unknown labels:
## • colour : "Ethnicity"

## Warning in scale_y_log10(labels = comma): log-10 transformation introduced
 infinite values.
## Removed 22 rows containing non-finite outside the scale range
## (`stat_ydensity()`).

dev.off()

## png
##   2
```

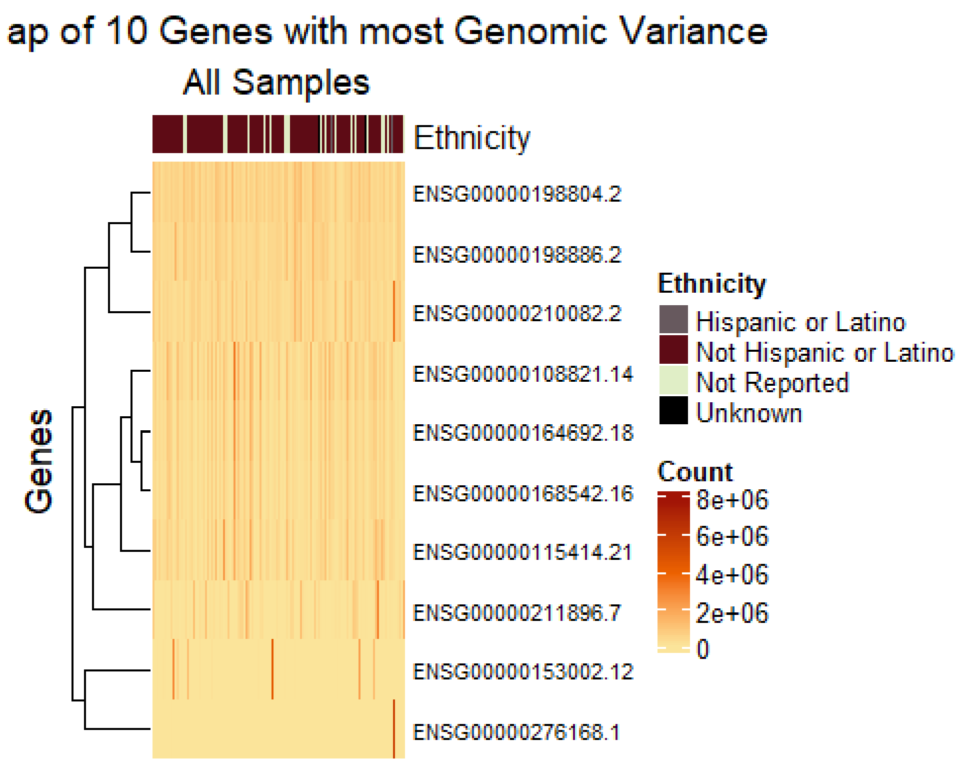
3. Heatmap Analysis:
   a. Select 10 genes: Choose a set of 10 different genes from the count matrix for your heatmap.
   b. Generate a Heatmap: Use the ComplexHeatmap package in R to create a heatmap of the count data for the selected genes
   c. Add an Annotation Bar: Include an annotation bar reflecting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

First Version with all samples

```r
# Create a new data frame with my ten selected Genes_id's.

#Load packages
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

# Select top 10 genes with the most variance.
my_ten_v <- summary_stats %>% arrange(desc(Gene_sd)) %>% head(10) %>% pull(Ge
ne_id)

# Create a new data frame with my ten selected Genes_id's.
my_ten_df <- RNA_seq_Count_matrix[RNA_seq_Count_matrix$Gene_id %in% c(my_ten_
v), ]

# Change row names
rownames(my_ten_df) <- my_ten_df[[1]]

# Transpose the data
my_gene_df <- tidyr::pivot_longer(my_ten_df, cols=starts_with("TCGA"))

my_gene_df_final <- tidyr::pivot_wider(my_gene_df, names_from = Gene_id)

# Merge my selected Genes RNA count data and meta data
RNA_seq_data <- merge(my_gene_df_final, RNA_seq_metadata, by="row.names")

my_ten_df <- my_ten_df[, -1]

# Read the necessary packages.
library(ComplexHeatmap)

## Loading required package: grid

## ========================================
## ComplexHeatmap version 2.22.0
## 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 multidimensi
onal
##     genomic data. Bioinformatics 2016.
##
##
## The new InteractiveComplexHeatmap package can directly export static
## complex heatmaps into an interactive Shiny app with zero effort. Have a tr
y!
##
## 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))
## ========================================

library(stringr)

# Add the co-variate as a bar.
co_Var <- HeatmapAnnotation(
  Ethnicity = RNA_seq_data$ethnicity,
  col = list(
    Ethnicity = c("hispanic or latino"="#67595e", "not hispanic or latino" ="
#5E0B15", "not reported"="#E0EEC6", "Unknown" = "black")
  ),
  annotation_legend_param = list(
                                Ethnicity=list(title="Ethnicity",
                                           labels = c("hispanic or latino"=
"Hispanic or Latino", "not hispanic or latino" ="Not Hispanic or Latino", "no
t reported"="Not Reported", "Unknown" = "Unknown")))
)


# First I've created a heatmap with all 1231 observations, the plot was too b
usy.
```

```
hm_plot <- Heatmap(
            my_ten_df,
            name = "Counts",
            top_annotation = co_Var,
            show_row_names = TRUE,
            show_column_names = FALSE,
            cluster_columns = FALSE,  #remove dendrograms
            cluster_rows = TRUE,  #remove dendrograms
            column_names_gp = gpar(fontsize = 8),
            row_names_gp = gpar(fontsize = 8),
            column_title = "All Samples",
            row_title = "Genes",
            column_labels = paste0(substr(colnames(my_ten_df), 6, 17), "\n",
substr(colnames(my_ten_df), 18, 28)),
            heatmap_legend_param = list(title = "Count"),
            col = c("#fbe49a", "#ed6200", "#a21406")

)

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

# Add plot title, move legends and wrap the text.
# Reference: https://www.rdocumentation.org/packages/ComplexHeatmap/versions/
1.10.2/topics/Heatmap
draw(hm_plot, column_title = "Heatmap of 10 Genes with most Genomic Variance",

     heatmap_legend_side = "right",
     annotation_legend_side = "right",
     merge_legend = TRUE,
     column_title_gp = gpar(fontsize = 14))
```

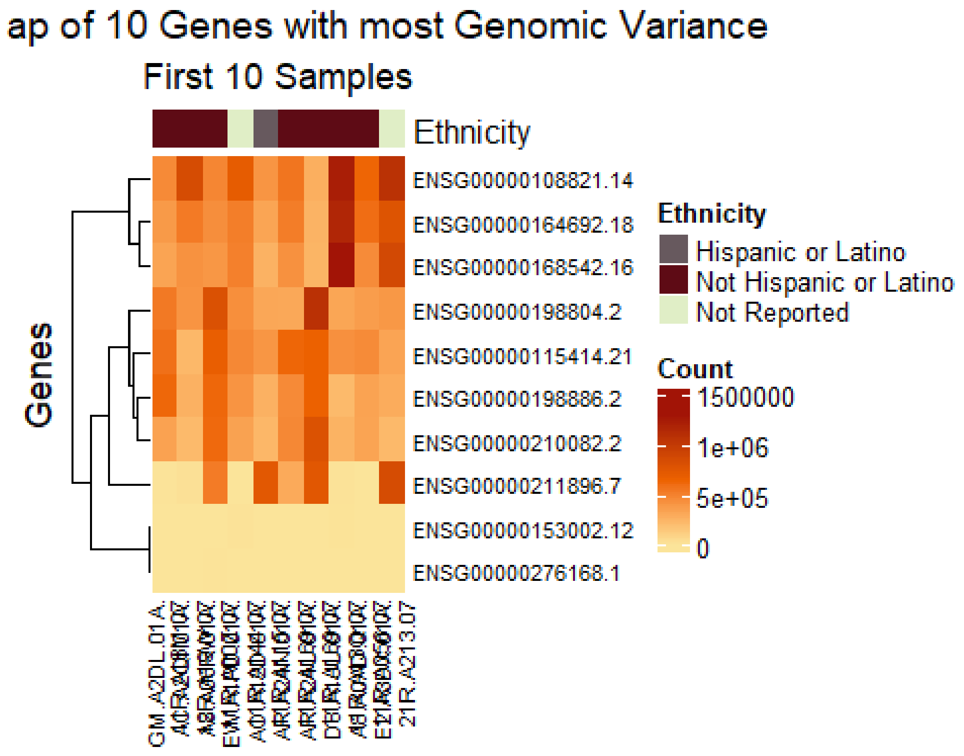


```
# save the plot to a png
png("C:/Users/f003cc3/Documents/hds/HSE 711 Foundations of Data Science/final
 project/Heatmap_Overall.png", width = 10000, height = 2000, res = 300)
draw(hm_plot, column_title = "Heatmap of 10 Genes with most Genomic Variance",

     heatmap_legend_side = "right",
     annotation_legend_side = "right",
     merge_legend = TRUE,
     column_title_gp = gpar(fontsize = 14))
dev.off()

## png
##   2
```

With 10 samples

```
# Limiting the data to the first 10 samples. I've created one with all the sa
mples and it was hard to interpret.
# Create a new data frame with my ten selected Genes_id's.

#Load packages
library(dplyr)

# Select top 10 genes with the most variance.
my_ten_v <- summary_stats %>% arrange(desc(Gene_sd)) %>% head(10) %>% pull(Ge
ne_id)
```

```r
# Create a new data frame with my ten selected Genes_id's.
my_ten_df <- RNA_seq_Count_matrix[RNA_seq_Count_matrix$Gene_id %in% c(my_ten_
v), ]

# Change row names
rownames(my_ten_df) <- my_ten_df[[1]]

# Select the first 11 columns (expressions).
my_ten_df <- my_ten_df[, 1:11]

# Transpose the data
my_gene_df <- tidyr::pivot_longer(my_ten_df, cols=starts_with("TCGA"))

my_gene_df_final <- tidyr::pivot_wider(my_gene_df, names_from = Gene_id)

# Merge my selected Genes RNA count data and meta data
RNA_seq_data <- merge(my_gene_df_final, RNA_seq_metadata, by="row.names")

my_ten_df <- my_ten_df[, -1]

# Read the necessary packages.
library(ComplexHeatmap)
library(circlize)
library(stringr)

# Add the co-variate as a bar.
co_Var <- HeatmapAnnotation(
  Ethnicity = RNA_seq_data$ethnicity,
  col = list(
    Ethnicity = c("hispanic or latino"="#67595e", "not hispanic or latino" ="
#5E0B15", "not reported"="#E0EEC6")
  ),
  annotation_legend_param = list(
                                Ethnicity=list(title="Ethnicity",
                                           labels = c("hispanic or latino"=
"Hispanic or Latino", "not hispanic or latino" ="Not Hispanic or Latino", "no
t reported"="Not Reported")))
)


# First I've created a heatmap with all 1231 observations, the plot was too b
usy, so, limiting it to 10 samples.
hm_plot <- Heatmap(
            my_ten_df,
            name = "Counts",
            top_annotation = co_Var,
            show_row_names = TRUE,
            show_column_names = TRUE,
            cluster_columns = FALSE,  #remove dendrograms
            cluster_rows = TRUE,
            column_names_gp = gpar(fontsize = 8),
            row_names_gp = gpar(fontsize = 8),
            column_title = "First 10 Samples",
            row_title = "Genes",
            column_labels = paste0(substr(colnames(my_ten_df), 6, 17), "\n",
substr(colnames(my_ten_df), 18, 28)),
            heatmap_legend_param = list(title = "Count"),
            col = c("#fbe49a", "#ed6200", "#a21406")

)

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

# Add plot title, move legends and wrap the text.
# Reference: https://www.rdocumentation.org/packages/ComplexHeatmap/versions/
1.10.2/topics/Heatmap
draw(hm_plot, column_title = "Heatmap of 10 Genes with most Genomic Variance",

     heatmap_legend_side = "right",
     annotation_legend_side = "right",
     merge_legend = TRUE,
     column_title_gp = gpar(fontsize = 14))
```



```r
# save the plot to a png
png("C:/Users/f003cc3/Documents/hds/HSE 711 Foundations of Data Science/final
 project/Heatmap.png", width = 2000, height = 1500, res = 300)
draw(hm_plot, column_title = "Heatmap of 10 Genes with most Genomic Variance",

     heatmap_legend_side = "right",
     annotation_legend_side = "right",
     merge_legend = TRUE,
     column_title_gp = gpar(fontsize = 14))
dev.off()
```

```
## png
##   2
```

```
# Summary statistics for ethnicity
summary_stats_co_var <- RNA_seq_data %>% filter(is.na(ethnicity) == FALSE) %>%
 group_by(ethnicity) %>% summarize(`Mean` = mean(ENSG00000153002.12),
                                                  `Median` = median(ENSG00000
153002.12),
                                                  `Standard Deviation` = stat
s::sd(ENSG00000153002.12),
                                                  `Variance` = stats::var(ENS
G00000153002.12),
                                                  `Max` = max(ENSG00000153002.
12),
                                                  `Min` = min(ENSG00000153002.
12)
                                                )
colnames(summary_stats_co_var)[1] <- "Ethnicity"

summary_stats_co_var
```

```
## # A tibble: 3 × 7
##   Ethnicity                Mean Median `Standard Deviation` Variance   Max
   Min
##   <chr>                   <dbl>  <dbl>                <dbl>    <dbl> <int>
 <int>
## 1 hispanic or latino     10035   10035                   NA       NA  10035
 10035
## 2 not hispanic or latino  1321.    137                2764. 7641236.   7500
    10
## 3 not reported             730     730                 987.   974408   1428
    32
```

```
# Summary statistics for pathologic stage
summary_stats_co_var2 <- RNA_seq_data %>% filter(is.na(ajcc_pathologic_stage)
 == FALSE) %>% group_by(ajcc_pathologic_stage) %>% summarize(`Mean` = mean(EN
SG00000153002.12),
                                                  `Median` = median(ENSG00000
153002.12),
                                                  `Standard Deviation` = stat
s::sd(ENSG00000153002.12),
                                                  `Variance` = stats::var(ENS
G00000153002.12),
                                                  `Max` = max(ENSG00000153002.
12),
                                                  `Min` = min(ENSG00000153002.
12)
                                                )

colnames(summary_stats_co_var2)[1] <- "Pathologic Stage"
library(gridExtra)

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine

# save the summary statistics to a png
png("C:/Users/f003cc3/Documents/hds/HSE 711 Foundations of Data Science/final
 project/summ1.png", width = 2500, height = 500, res = 300)
grid.table(summary_stats_co_var)
dev.off()

## png
##   2

# save the summary statistics to a png
png("C:/Users/f003cc3/Documents/hds/HSE 711 Foundations of Data Science/final
 project/summ2.png", width = 2600, height = 1200, res = 300)
grid.table(summary_stats_co_var2)
dev.off()

## png
##   2
```

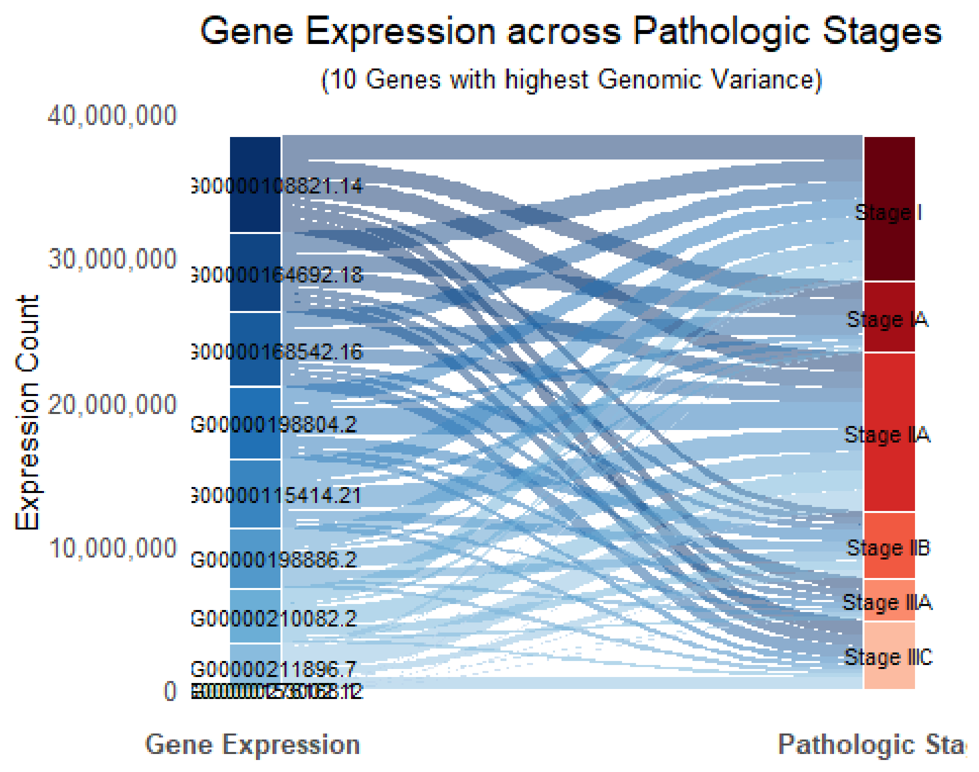
Sankey Chart for 10 selected genes acroos pathologic stages

```
library(tidyr)
library(dplyr)
library(ggalluvial) # for sankey outside d3
library(ggrepel)

# First create a data frame for the sankey chart
# Sankey chart data format:
# Source ---------> Target
# Give indices for each node point (unique values of gene names and pathologi
c stages)
# group by gene name and pathologic stages.
# Order the data in the descending order of expression counts so that
# genes order is descending and the stages are alphabetical.
```

```r
RNA_df_long <- RNA_seq_data %>%
  pivot_longer(
    cols = starts_with("ENSG"),           # Select columns starting with ENSG
    names_to = "gene_name",                # New column for gene names
    values_to = "expression_count"         # New column for values
  ) %>% select(gene_name, expression_count, ajcc_pathologic_stage) %>% group_
by(gene_name, ajcc_pathologic_stage) %>%
  summarize(exp_count = sum(expression_count)) %>% filter(!is.na(ajcc_patholo
gic_stage))
```

```
## `summarise()` has grouped output by 'gene_name'. You can override using th
e
## `.groups` argument.
```

```r
# descending order of genes by expression count
gene_order <- RNA_df_long %>% group_by(gene_name) %>%
              summarize(total_exp = sum(exp_count)) %>%
              arrange(desc(total_exp)) %>% pull(gene_name)

# pathologic stages ordered alphabetically
stage_order <- sort(unique(RNA_df_long$ajcc_pathologic_stage))

# Apply ordering
RNA_df_long <- RNA_df_long %>%
  mutate(
    gene_name = factor(gene_name, levels = gene_order),
    ajcc_pathologic_stage = factor(ajcc_pathologic_stage, levels = stage_orde
r)
  )

# Create color palettes for genes and pathologic stages
blue_gradient <- colorRampPalette(c("#08306b", "#2171b5", "#6baed6", "#c6dbef
"))(length(gene_order))
red_gradient <- colorRampPalette(c("#67000d", "#cb181d", "#fb6a4a", "#fcbba1
"))(length(stage_order))

# create startum colors (the bars for the nodes)
stratum_colors <- c(
  setNames(blue_gradient, gene_order),
  setNames(red_gradient, stage_order)
)

# for the sankey curves, create colors
flow_colors <- setNames(blue_gradient, gene_order)

# Create alluvial plot with titles
sankey_plot <- ggplot(RNA_df_long,
       aes(y = exp_count,
           axis1 = gene_name,
           axis2 = ajcc_pathologic_stage)) +
  geom_alluvium(aes(fill = gene_name), width = 1/12) +
  geom_stratum(aes(fill = after_stat(stratum)),
               width = 1/12, color = "white") +       # this will add bars to
 source and target nodes
  scale_x_discrete(limits = c("Gene Expression", "Pathologic Stage"),
                   expand = c(.05, .05)) +
  scale_y_continuous(labels=label_comma()) +
  scale_fill_manual(
    values = c(flow_colors, stratum_colors),
    breaks = c(gene_order, stage_order)
  ) +
  labs(title = "Gene Expression across Pathologic Stages",
       subtitle = "(10 Genes with highest Genomic Variance)",
       y = "Expression Count") +
  theme_minimal() +
  theme(
    legend.position = "none",
    plot.title = element_text(size = 14, hjust = 0.5),
    plot.subtitle = element_text(size=10, hjust = 0.5),
    axis.text.y = element_text(size = 10),
    axis.text.x = element_text(size = 10, face = "bold"),
    axis.title.x = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank()
  ) +
  geom_text(stat = "stratum", aes(label = after_stat(stratum)), size = 3)

plot(sankey_plot)
```



```
# save the sankey to a png
png("C:/Users/f003cc3/Documents/hds/HSE 711 Foundations of Data Science/final
 project/sankey.png", width = 2600, height = 2000, res = 300)
plot(sankey_plot)
dev.off()

## png
##   2
```