FinalSubmission

Antara Sengupta

2024-08-27

QBS 103 Final Project

The code below works with a dataset containing gene expression and other biological and health information for patients with and without COVID-19. This specific project focuses on the gene AASDHPPT.

```
# Loading in all required packages
library(readr)
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
```

```
library(ggplot2)
library(tibble)
library(tidyr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v purrr     1.0.2
## v lubridate 1.9.2     v stringr   1.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(table1)
```

```
##
## Attaching package: 'table1'
##
## The following objects are masked from 'package:base':
##
##     units, units<-
```

```
library(ggridges)
library(boot)
library(pheatmap)
```

Cleaning Data

Using very similar data cleaning methods from Submission 1 and 2. Ensuring that data is clean prior to analysis.

Loading Data

```
# loading in the two provided datasets (metadata & gene expression data)
series <- read.csv("data/QBS103_GSE157103_series_matrix.csv")
genes <- read.csv("data/QBS103_GSE157103_genes.csv")
```

```
# Goal is to merge series matrix and genes csv based upon Participant ID

# Transposing the genes dataframe  so it can have 'participant_id' as rows
# to match the metadata
genes_transposed <- as.data.frame(t(genes))

# Setting column headers as the name of the gene
colnames(genes_transposed) <- as.character(genes_transposed[1, ])

# Removing 'X' row due to redundant information
genes_transposed <- genes_transposed[-1, ]

# Setting the row name to Participant ID to match with metadata
genes_transposed <- rownames_to_column(genes_transposed, var = "participant_id")

# Reshaping the gene DF so that genes and their expression can be their own columns
#using pivot_longer
genes_long <- genes_transposed %>%
  pivot_longer(
    cols = -participant_id,  # Accessing all columns with the exception of participant_id because it i
    names_to = "gene", # Setting name column to gene
    values_to = "expression"  # Setting values column to expression
  )

# Merging the long gene DF and series metadata
all_data <- merge(genes_long, series, by = "participant_id")

# Replacing all unknown values in the data, formatted as " unknown", with NA
all_data[all_data == " unknown"] <- NA
all_data[all_data == "unknown"] <- NA

# Drop rows with any NA values
all_data <- na.omit(all_data)
```

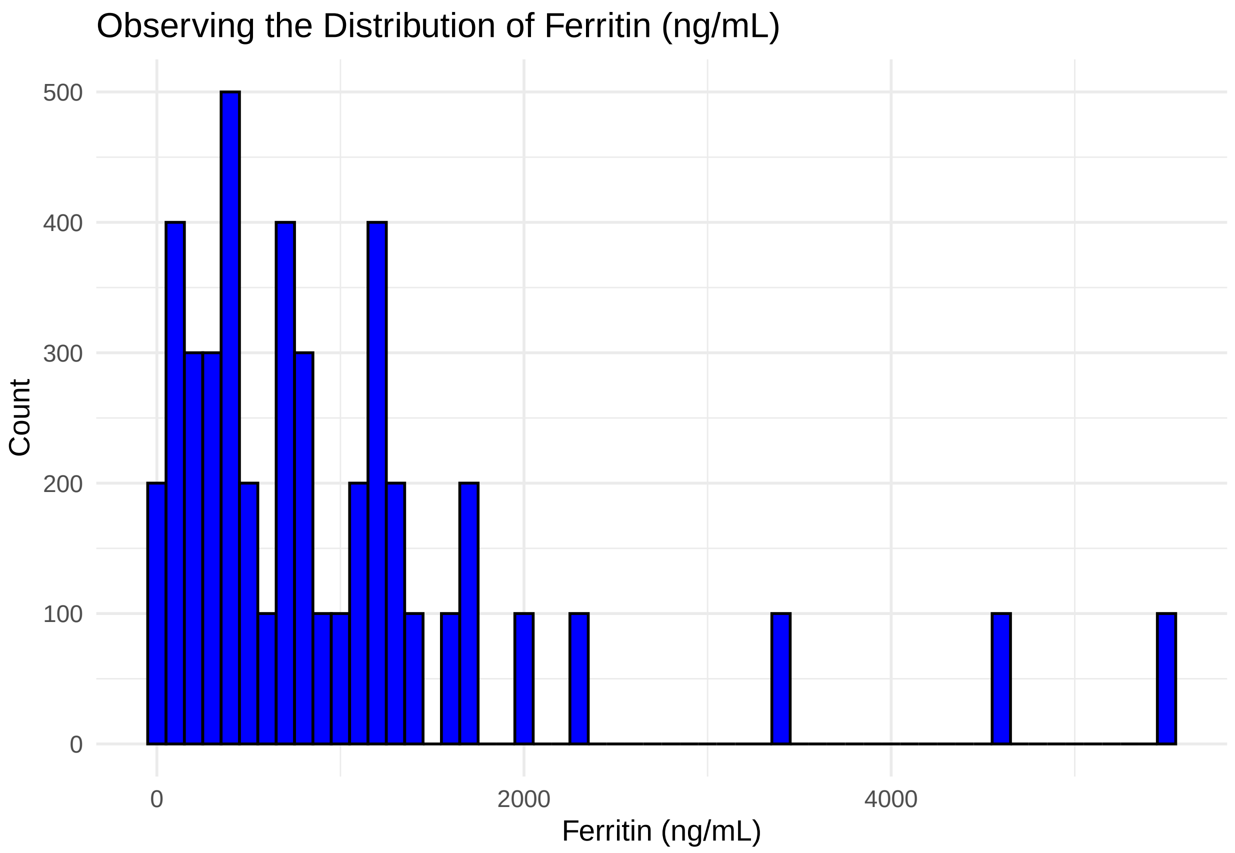
Generating Summary Table

The code below generates a table of summary statistics for ICU status, sex, ferritin levels, lactate levels and D-Dimer levels. It stratifies these covariates by disease status. It also reports the mean (sd) or median [IQR] for continuous variables based on the distribution of the continuous variables To figure out the distribution of the continuous variables, I have plotted their distribution to see if it is normal or not. If it is normal, the summary statistics table will include the mean for that covariate - if it is not, it will include the median IQR.

Exploring Distributions

The first distribution I will check is ferritin.

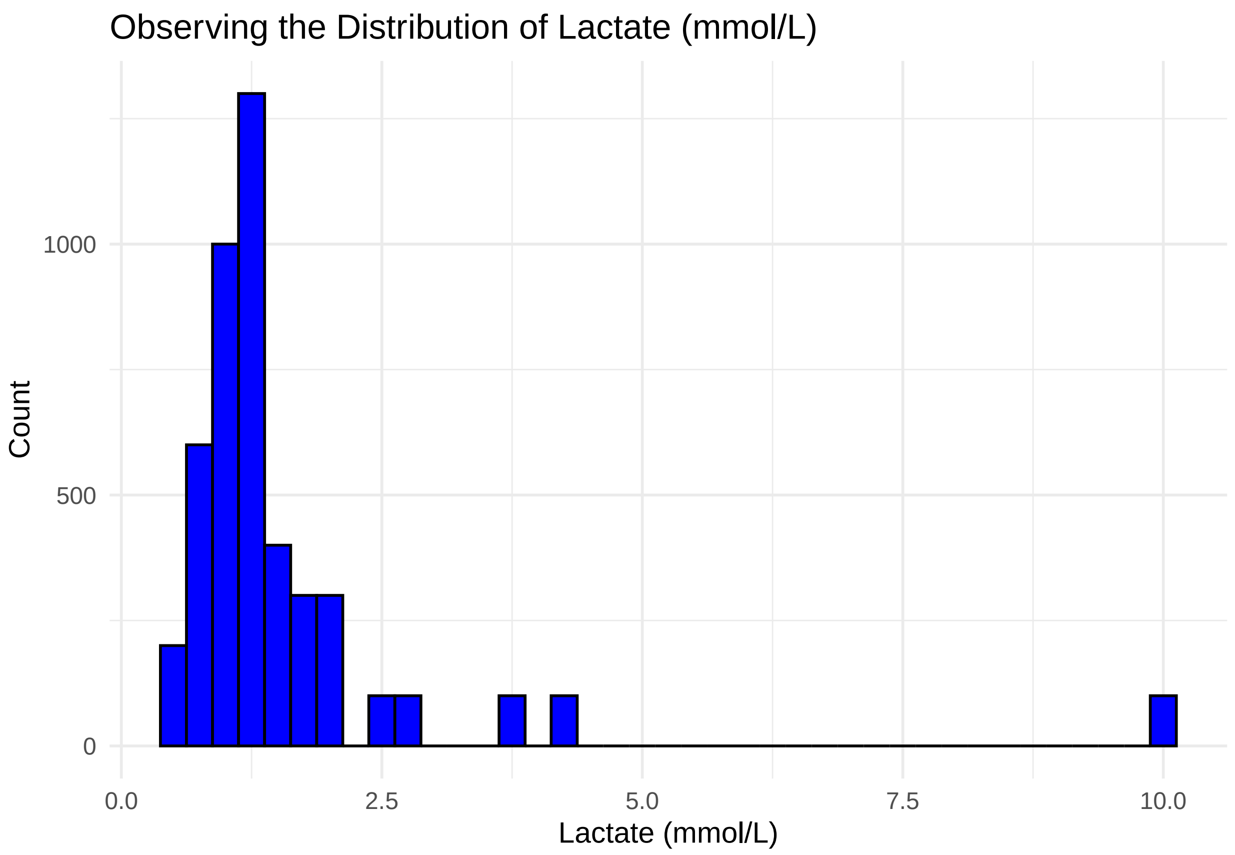
```
# Plotting Distribution of Ferritin
ggplot(all_data, aes(x=as.numeric(ferritin.ng.ml.))) +
   # Setting appropriate bin-width - larger one works better here
  geom_histogram(binwidth = 100, color = "black", fill = "blue") +
  labs(title = "Observing the Distribution of Ferritin (ng/mL)",
       x= "Ferritin (ng/mL)" ,
       y = "Count"
       ) +
  theme_minimal() # Choosing preferred theme
```



It is apparent that that ferritin does not have a normal distribution in this dataset, and will therefore have its median IQR calculated in the summary table.

Steps above repeated below except with lactate.

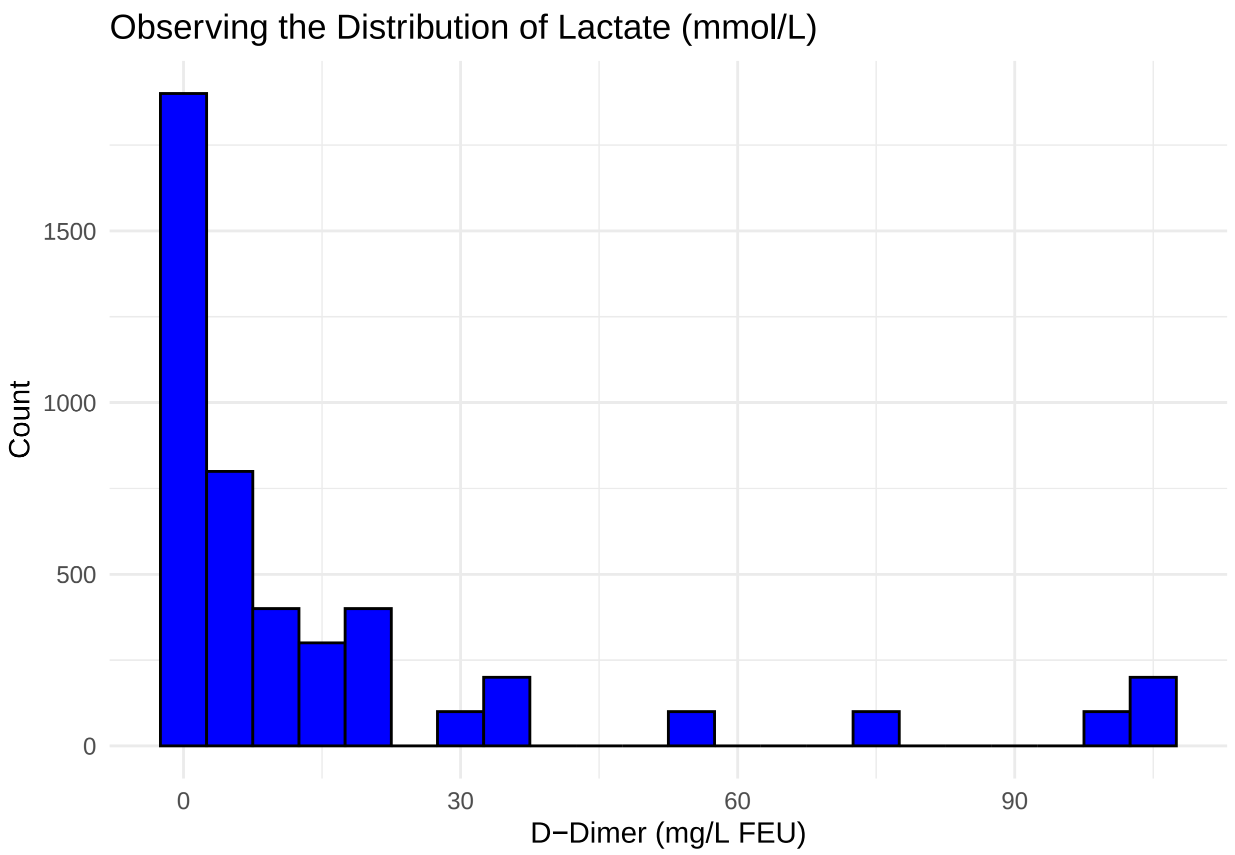
```
# Plotting Distribution of Lactate
ggplot(all_data, aes(x=as.numeric(lactate.mmol.l.))) +
  # Setting appropriate bin-width - smaller one works better here
  geom_histogram(binwidth = 0.25, color = "black", fill = "blue") +
  labs(title = "Observing the Distribution of Lactate (mmol/L)",
       x= "Lactate (mmol/L)" ,
       y = "Count"
       ) +
  theme_minimal() # Choosing preferred theme
```



It is apparent that lactate does not have a normal distribution in this dataset, and will therefore have its median IQR calculated in the summary table.

Steps above repeated below except with D-Dimer.

```r
# Plotting Distribution of D-Dimer
ggplot(all_data, aes(x=as.numeric(ddimer.mg.l_feu.))) +
  # Setting appropriate bin-width
  geom_histogram(binwidth = 5, color = "black", fill = "blue") +
  labs(title = "Observing the Distribution of Lactate (mmol/L)",
       x= "D-Dimer (mg/L FEU)" ,
       y = "Count"
       ) +
  theme_minimal() # Choosing preferred theme
```



It is apparent that D-Dimer does not have a normal distribution in this dataset, and will therefore have its median IQR calculated in the summary table.

Now that we have all this information, we can go ahead and create the summary table.

Building table

```r
# Creating a new DF that will be used to build summary tab;e
# Referred to code from:
# https://cran.r-project.org/web/packages/table1/vignettes/table1-examples.html
table_data <- all_data
table_data <- na.omit(table_data)

# Build rndr function to customize statistics calculated
rndr <- function(x, name, ...) {
    if (!is.numeric(x)) return(render.categorical.default(x))
    what <- switch(name,
        # median IQR for all three b/c none are normal
        ferritin.ng.ml. = "Median [IQR]",
        lactate.mmol.l.  = "Median [IQR]",
        ddimer.mg.l_feu.  = "Median [IQR]"
        )
    parse.abbrev.render.code(c("", what))(x)
}

# Changing values of continuous covariate from character to numeric
table_data$ferritin.ng.ml. <- as.numeric(table_data$ferritin.ng.ml.)
table_data$ddimer.mg.l_feu. <- as.numeric(table_data$lactate.mmol.l.)
```

```r
table_data$lactate.mmol.l. <- as.numeric(table_data$ddimer.mg.l_feu.)

# Changing disease status values to clearer/more professional ones
table_data <- table_data %>%
  mutate(disease_status = case_when(
    disease_status == "disease state: COVID-19" ~ "COVID-19 Positive",
    disease_status == "disease state: non-COVID-19" ~ "COVID-19 Negative",
    TRUE ~ disease_status
  ))

# Changing sex values to clearer/more professional ones
table_data <- table_data %>%
  mutate(sex = case_when(
    sex == " female" ~ "Female",
    sex == " male" ~ "Male",
    TRUE ~ sex
  ))

# Changing icu status values to clearer/more professional ones
table_data <- table_data %>%
  mutate(icu_status = case_when(
    icu_status == " yes" ~ "Yes",
    icu_status == " no" ~ "No",
    TRUE ~ icu_status
  ))

# Setting the labels of each covariate within the table
label(table_data$icu_status)  <- "ICU Status"
label(table_data$disease_status)  <- "Disease Status"
label(table_data$sex) <- "Sex"
label(table_data$ferritin.ng.ml.) <- "Ferritin"
label(table_data$lactate.mmol.l.) <- "Lactate"
label(table_data$ddimer.mg.l_feu.) <- "D-Dimer"

# Setting units for continuous covariates
units(table_data$ferritin.ng.ml.)   <- "ng/mL"
units(table_data$lactate.mmol.l.)    <- "mmol/L"
units(table_data$ddimer.mg.l_feu.)    <- "mg/L FEU"

# Building the table with my covariates of choice facet wrapped up disease status
table1(~ icu_status + sex + ferritin.ng.ml. + lactate.mmol.l. + ddimer.mg.l_feu.|disease_status,
       data=table_data,
       topclass="Rtable1-grid Rtable1-shade Rtable1-times",
       render=rndr)
```

```
## Get nicer `table1` LaTeX output by simply installing the `kableExtra` package
```

	COVID-19 Negative	COVID-19 Positive	Overall
	(N=500)	(N=4100)	(N=4600)
ICU Status			
Yes	500 (100%)	3700 (90.2%)	4200 (91.3%)
No	0 (0%)	400 (9.8%)	400 (8.7%)
Sex			

	COVID-19 Negative	COVID-19 Positive	Overall
Female	300 (60.0%)	900 (22.0%)	1200 (26.1%)
Male	200 (40.0%)	3200 (78.0%)	3400 (73.9%)
Ferritin (ng/mL)			
Median [IQR]	211 [184]	800 [906]	722 [899]
Lactate (mmol/L)			
Median [IQR]	3.68 [3.07]	1.18 [0.540]	1.20 [0.580]
D-Dimer (mg/L FEU)			
Median [IQR]	3.68 [3.07]	1.18 [0.540]	1.20 [0.580]

Publication Quality Histogram, Scatterplot, Boxplot

The code below generates a publication quality histogram, scatter plot, and boxplot from submission 1 for your the gene AASDHPPT.

Building a new dataframe to use specifically for these following plots. I will subset the dataframe so that it only contains columns that I want to perform further analysis on.

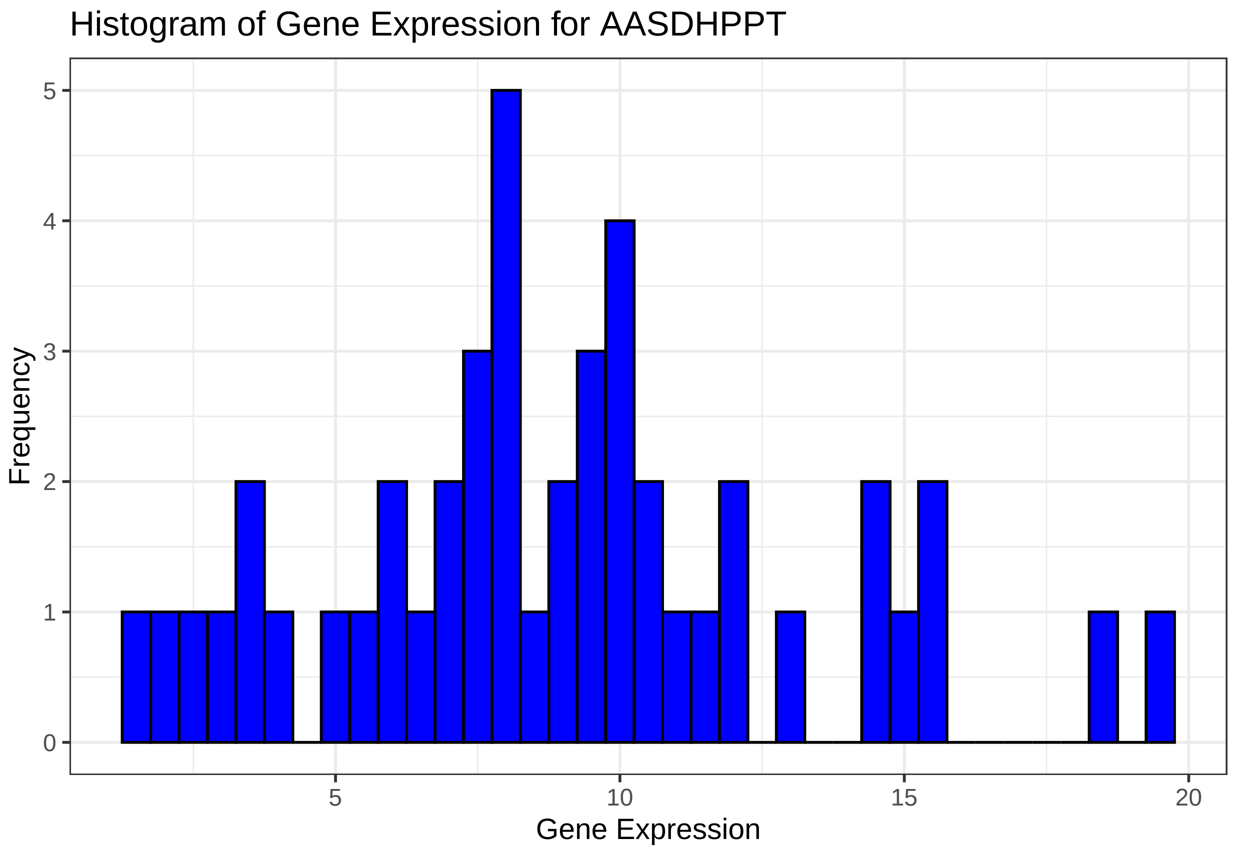
```r
# Creating new DF that contains columns of interest for my plots
covid_data <- all_data %>% select(participant_id,
                                  gene,
                                  expression,
                                  ferritin.ng.ml., # Continuous Covariate
                                  sex, # Categorical Covariate
                                  disease_status # Categorical Covariate
                                  )
```

Publication Quality Histogram

The code below creates a histogram of AASDHPPT expression.

```r
# Creating new DF which only contains rows where the gene is AASDHPPT
AASDHPPT_data <- covid_data %>%
  filter(gene == "AASDHPPT")

# Ensuring columns with numeric values are of numeric type
AASDHPPT_data$expression <- as.numeric(AASDHPPT_data$expression)
AASDHPPT_data$ferritin.ng.ml. <- as.numeric(AASDHPPT_data$ferritin.ng.ml. )

# Creating new DF only containing cols of interest for histogram
AASDHPPT_hist <- AASDHPPT_data %>%
  select(expression)

# Creating histogram using ggplot
gene_hist <- ggplot(AASDHPPT_data, aes(x =as.numeric(expression))) +
  geom_histogram(binwidth = 0.5, color = "black", fill = "blue") +
  labs(title = "Histogram of Gene Expression for AASDHPPT",
       x = "Gene Expression",
       y = "Frequency") +
  theme_bw() # Choosing preferred theme
```

```
# Displaying graph
gene_hist
```

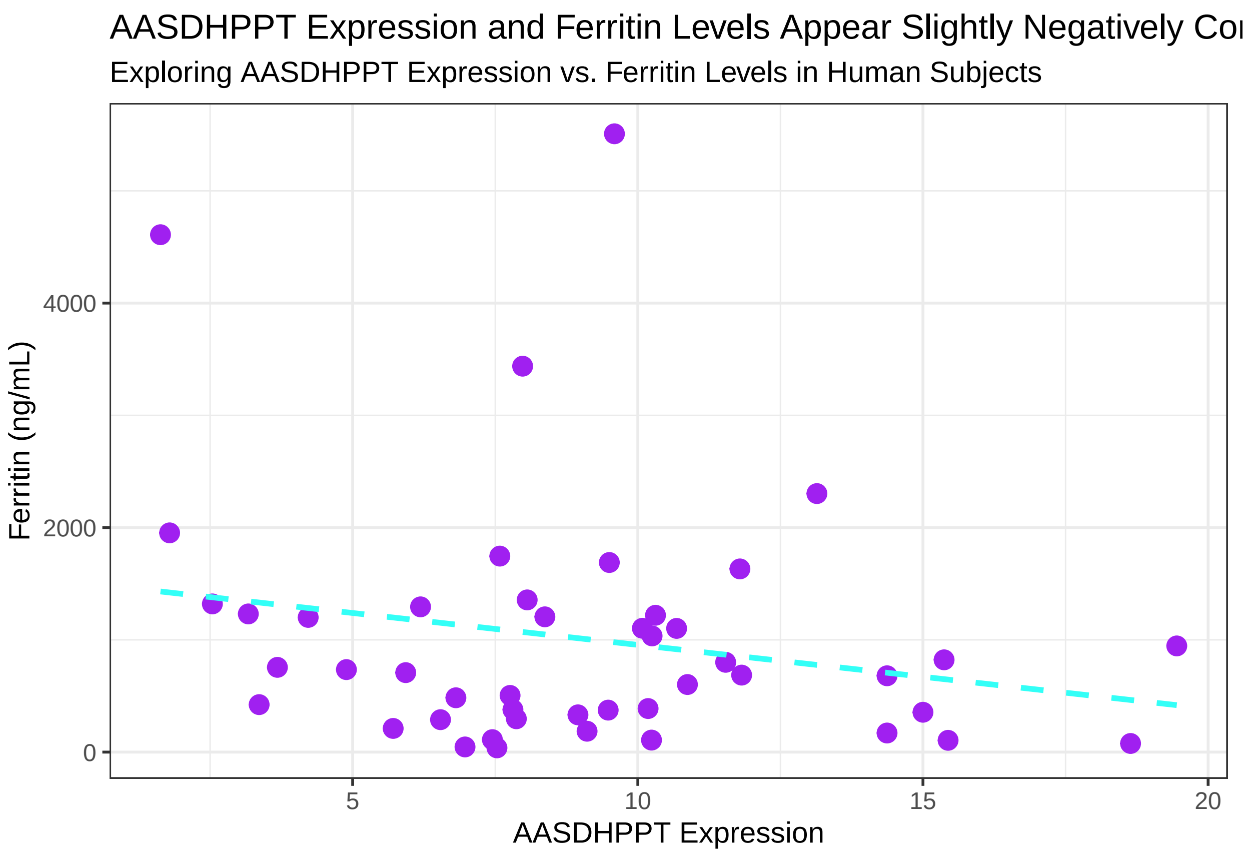


Publication Quality Scatterplot

The code below creates a scatterplot observing of AASDHPPT expression vs. Ferritin Levels.

```
# Creating scatterplot using ggplot
scatter_plot <- ggplot(AASDHPPT_data, aes(x = expression, y = ferritin.ng.ml.)) +
    geom_point(color = "purple",size = 3) + # Incr size of the data pts
    geom_smooth(method = "lm", se = FALSE, color = "#33FFF7",linetype = "dashed") +
  labs(title = "AASDHPPT Expression and Ferritin Levels Appear Slightly Negatively Correlated",
      subtitle = "Exploring AASDHPPT Expression vs. Ferritin Levels in Human Subjects",
       x= "AASDHPPT Expression" ,
       y = "Ferritin (ng/mL)") +
  theme_bw() # Changing to preferred theme

# Displaying graph
scatter_plot
```

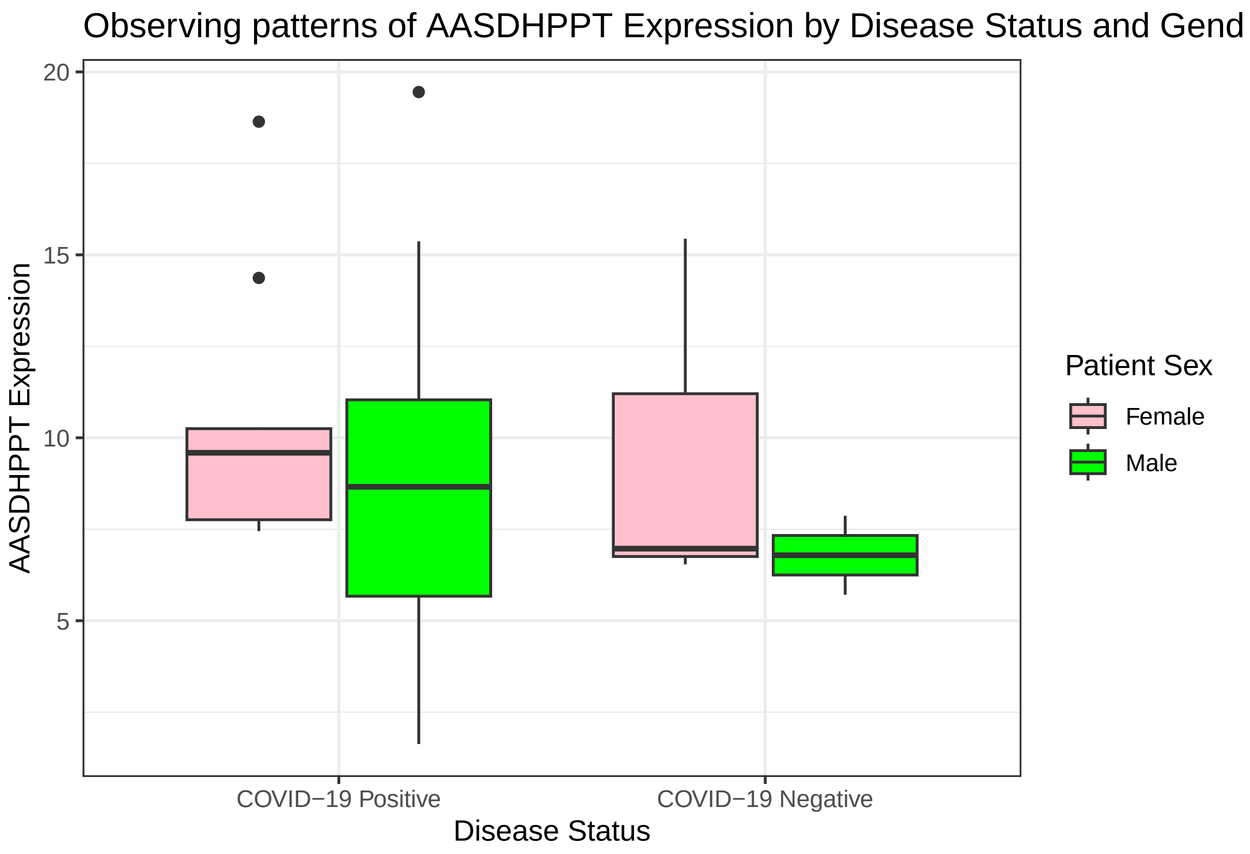
```
## `geom_smooth()` using formula = 'y ~ x'
```



Publication Quality Boxplot

The code below creates a boxplot observing of AASDHPPT expression by Disease Status and Gender.

```
# Changing sex values to cleaer/more professional ones
AASDHPPT_data <- AASDHPPT_data %>%
  mutate(sex = case_when(
    sex == " female" ~ "Female",
    sex == " male" ~ "Male",
    TRUE ~ sex
  ))
custom_colors <- c("Male" = "green", "Female" = "pink")
# Creating boxplot using ggplot
gene_box <- ggplot(AASDHPPT_data,
                   aes(x = disease_status,
                       y = as.numeric(expression),
                       fill = sex
                       )) +
  geom_boxplot() +
  scale_x_discrete(labels = c("disease state: COVID-19" = "COVID-19 Positive",
                              "disease state: non-COVID-19" = "COVID-19 Negative"
                              ))+ # Changing the x values to visually cleaner ones
  labs( # Adding labels and titles to the graph
    title = "Observing patterns of AASDHPPT Expression by Disease Status and Gender",
    x= "Disease Status",
    y = "AASDHPPT Expression",
    fill = "Patient Sex" ) +
  scale_fill_manual(values = custom_colors)+
  theme_bw() # Changing to preferred theme

# Displaying graph
gene_box
```

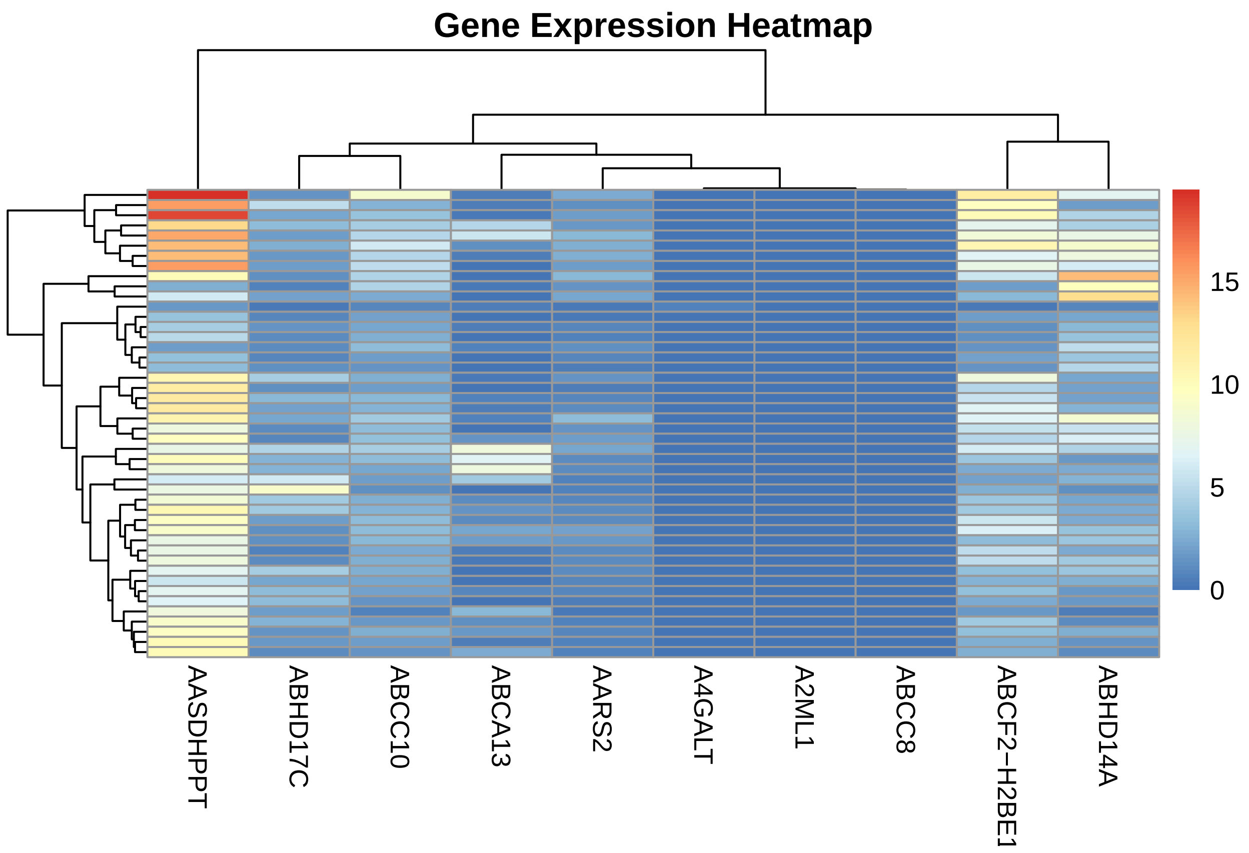


Heatmap

The code below generates a heatmap which includes at least 10 genes along with clustered rows and columns.

```
# Creating a vector containing genes of interest
heat_genes <- c('AASDHPPT',
                'ABCF2-H2BE1',
                'ABHD17C',
                'A2ML1',
                'A4GALT',
                'ABCC10',
                'ABCA13',
                'ABHD14A',
                'ABCC8',
                'AARS2'
                )

# Creating a new dataframe using spread to access the genes and values
# Referred to this site to implement spread function:
# https://tidyr.tidyverse.org/reference/spread.html
newdf <- spread(all_data, key=gene, value=expression)
```

```r
# Accessing dataframe only has rows for genes of interest
newdf <- newdf[,heat_genes]

# Changing all values in the dataframe to numeric
newdf <-  mutate_all(newdf, function(x) as.numeric(as.character(x)))

# Creating the heatmap
heat <- pheatmap(newdf,
         cluster_rows = TRUE,
         cluster_cols = TRUE,
         main = "Gene Expression Heatmap",
         show_rownames = FALSE)
```

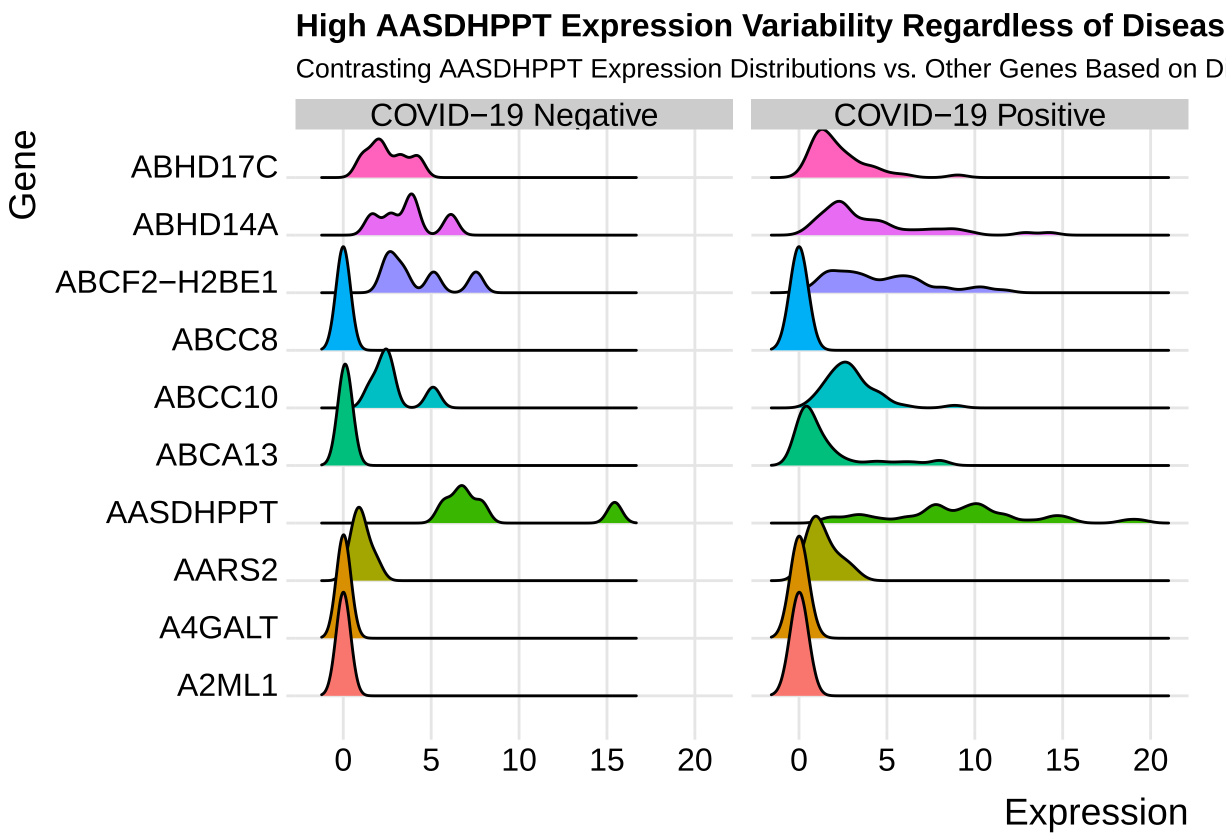


Ridgeplot

The code below creates a ridge plot that looks at the expression of the same 10 genes in the heatmap, and compares the expression between COVID-19 disease status.

```r
# Referred to this code to create my plot:
# https://r-graph-gallery.com/294-basic-ridgeline-plot.html

# Defining my genes of interest for the ridge plot
ridge_genes <- c('AASDHPPT',
                'ABCF2-H2BE1',
                'ABHD17C',
                'A2ML1',
                'A4GALT',
                'ABCC10',
                'ABCA13',
                'ABHD14A',
                'ABCC8',
                'AARS2'
                )

# Filtering covid data for only rows with genes above, creating new DF
ridge_data <- covid_data %>%
  filter(gene %in% ridge_genes)

# Changing disease status labels to more professional ones
ridge_data <- ridge_data %>%
  mutate(disease_status = case_when(
    disease_status == "disease state: COVID-19" ~ "COVID-19 Positive",
    disease_status == "disease state: non-COVID-19" ~ "COVID-19 Negative",
    TRUE ~ disease_status
  ))

# Changing expression values from character to numeric
ridge_data$expression <- as.numeric(ridge_data$expression)

# Creating ridge plot of genetic expression
ridge_plot <- ggplot(ridge_data, aes(x = expression, y = gene, fill = gene)) +
  geom_density_ridges() +
  # Facet Wrapping by disease status
  facet_wrap(~disease_status) +
  labs( #Adding labels and titles to the graph
    title = "High AASDHPPT Expression Variability Regardless of Disease Status",
    subtitle = "Contrasting AASDHPPT Expression Distributions vs. Other Genes Based on Disease Status",
    x = 'Expression',
    y = 'Gene',
    fontsize = 2) +
  theme_ridges() +  # Choosing desired theme
  theme(legend.position = "none") + # Removing legend b/c redundant info
  # Adjusting the size of the titles and subtitles
  theme(
    plot.title = element_text(size = 12, face = "bold"),
    plot.subtitle = element_text(size = 10)
  )

ridge_plot
```

```
## Picking joint bandwidth of 0.411

## Picking joint bandwidth of 0.526
```


High AASDHPPT Expression Variability Regardless of Diseas
Contrasting AASDHPPT Expression Distributions vs. Other Genes Based on D
COVID−19 Negative
COVID−19 Positive
Gene
ABHD17C
ABHD14A
ABCF2−H2BE1
ABCC8
ABCC10
ABCA13
AASDHPPT
AARS2
A4GALT
A2ML1
0
5
10
15
20
Expression