final_submission

2024-08-29

```
# Load necessary libraries
library(ggplot2)
library(dplyr)
library(readr)
library(data.table)
library(systemfonts)
library(tidyverse)
library(pheatmap)
```

```
# Load the datasets
meta_data_path = 'dataset/QBS103_GSE157103_series_matrix.csv'
gene_data_path = 'dataset/QBS103_GSE157103_genes.csv'
genes_data <- read_csv(gene_data_path, show_col_types = FALSE)
```

```
## New names:
## * `` -> `...1`
```

```
metadata <- read_csv(meta_data_path, show_col_types = FALSE)

# First, in order to simplify the remaining work
#we will first clean and merge two datasets completely
### step one, clean the column name of gene data ##
# Rename the first column to "Genes" if needed
if ("...1" %in% colnames(genes_data)) {
    genes_data <- genes_data %>% rename(Genes = `...1`)
} else {
    print("All columns are correctly named")
}
### step two, transpose the dataset ###
genes_df <- genes_data
# Set gene names as rownames
genes_df = genes_df[,-1]
rownames(genes_df) <- genes_data$Genes
# Transpose the data and convert to a data frame
transposed_genes_data <- as.data.frame(t(genes_df))
# Assign meaningful column names (genes) after transposition
transposed_genes_data <- cbind(participant_id = rownames(transposed_genes_data), transposed_genes_data)
### step three we merge the gene data with meta data
data_frame <- merge(metadata, transposed_genes_data, by = "participant_id")
rownames(data_frame) <- metadata$participant_id
### step four, we want to clean the meta data (NAs/types/...)
data_frame <- data_frame %>%
  mutate(
```

```
    age = as.numeric(age),  # Convert age to numeric
    ferritin = as.numeric(`ferritin(ng/ml)`),  # Convert ferritin to numeric
    crp = as.numeric(`crp(mg/l)`),  # Convert CRP to numeric
    # Handle categorical variables: filter out rows where any of the specified categorical variables ar
    sex = ifelse(tolower(sex) == "unknown", NA, sex),
    icu_status = ifelse(tolower(icu_status) == "unknown", NA, icu_status),
    disease_status = ifelse(tolower(disease_status) == "unknown", NA, disease_status)
  ) %>%
  # Remove rows with NA values in any of the categorical columns after filtering "unknown"
  filter(!is.na(sex), !is.na(icu_status), !is.na(disease_status))
```

```
# Create a function that calculate the mean and std
summary_continuous <- function(var, group) {
  data_frame %>%
    group_by(!!sym(group)) %>%
    summarise(
      Mean = mean(!!sym(var), na.rm = TRUE),
      SD = sd(!!sym(var), na.rm = TRUE),
    )
}

age_summary <- summary_continuous("age", "icu_status")
ferritin_summary <- summary_continuous("ferritin", "icu_status")
crp_summary <- summary_continuous("crp", "icu_status")
age_summary
```

```
## # A tibble: 2 x 3
##   icu_status  Mean    SD
##   <chr>      <dbl> <dbl>
## 1 no          58.7  17.8
## 2 yes         63.2  13.9
```

```
ferritin_summary
```

```
## # A tibble: 2 x 3
##   icu_status  Mean    SD
##   <chr>      <dbl> <dbl>
## 1 no          716. 1068.
## 2 yes         935. 1019.
```

```
crp_summary
```

```
## # A tibble: 2 x 3
##   icu_status  Mean    SD
##   <chr>      <dbl> <dbl>
## 1 no          109.  94.4
## 2 yes         150. 106.
```

```
# Create a function that calculate the count and percentage
summary_categorical <- function(var, group) {
  data_frame %>%
```

```
    group_by(!!sym(group), !!sym(var)) %>%
    summarise(n = n()) %>%
    mutate(percentage = n / sum(n) * 100)
}
sex_summary <- summary_categorical("sex", "icu_status")
```

```
## `summarise()` has grouped output by 'icu_status'. You can override using the
## `.groups` argument.
```

```
charlson_summary <- summary_categorical("disease_status", "icu_status")
```

```
## `summarise()` has grouped output by 'icu_status'. You can override using the
## `.groups` argument.
```

```
sex_summary
```

```
## # A tibble: 4 x 4
## # Groups:   icu_status [2]
##   icu_status sex        n percentage
##   <chr>      <chr>  <int>      <dbl>
## 1 no         female    27       45
## 2 no         male      33       55
## 3 yes        female    24       36.9
## 4 yes        male      41       63.1
```

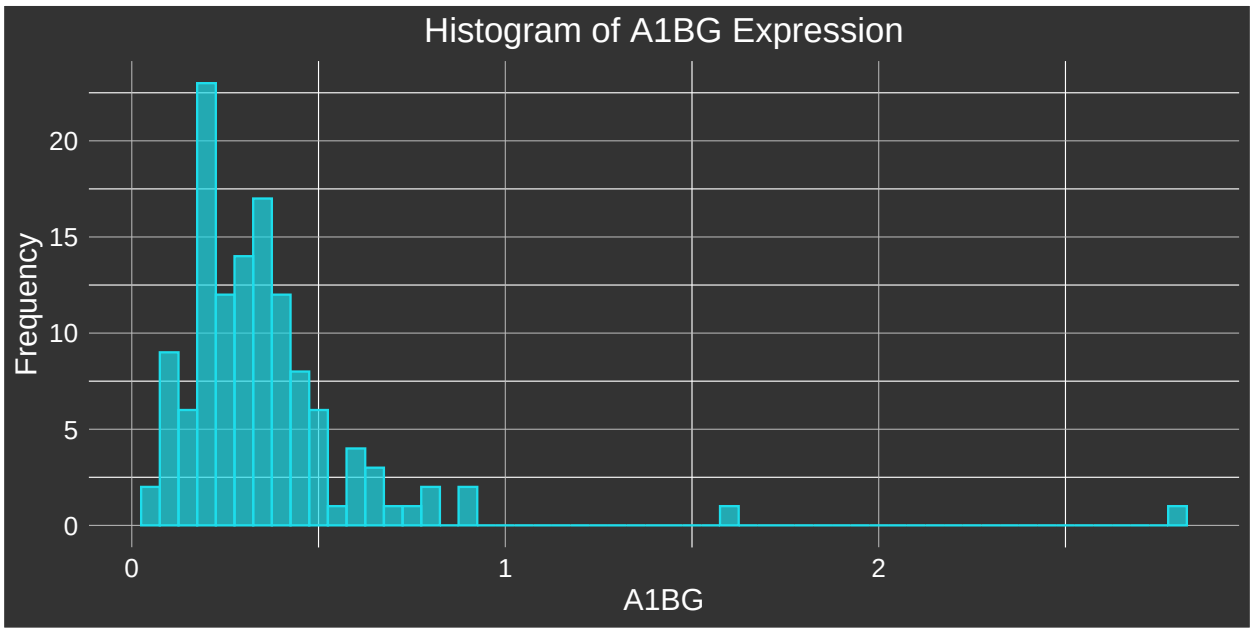
```
charlson_summary
```

```
## # A tibble: 4 x 4
## # Groups:   icu_status [2]
##   icu_status disease_status                 n percentage
##   <chr>      <chr>                      <int>      <dbl>
## 1 no         disease state: COVID-19       50       83.3
## 2 no         disease state: non-COVID-19   10       16.7
## 3 yes        disease state: COVID-19       50       76.9
## 4 yes        disease state: non-COVID-19   15       23.1
```

```
# Define a custom theme
custom_theme <- theme(
    panel.background = element_rect(fill = '#333333'),
    plot.title = element_text(hjust = 0.5, color = 'white', size = 16),
    axis.title = element_text(size = 14, color = 'white'),
    axis.text = element_text(size = 12, color = 'white'),
    plot.background = element_rect(fill = '#333333'),
    panel.grid.major = element_line(color = "gray", size = 0.1),
    legend.position = 'top',
    legend.background = element_rect(fill = "#333333"),
    legend.text = element_text(size = 12, color = 'white'),
    legend.title = element_text(size = 12, color = 'white'),
    legend.key = element_rect(fill = "#333333")
)
```

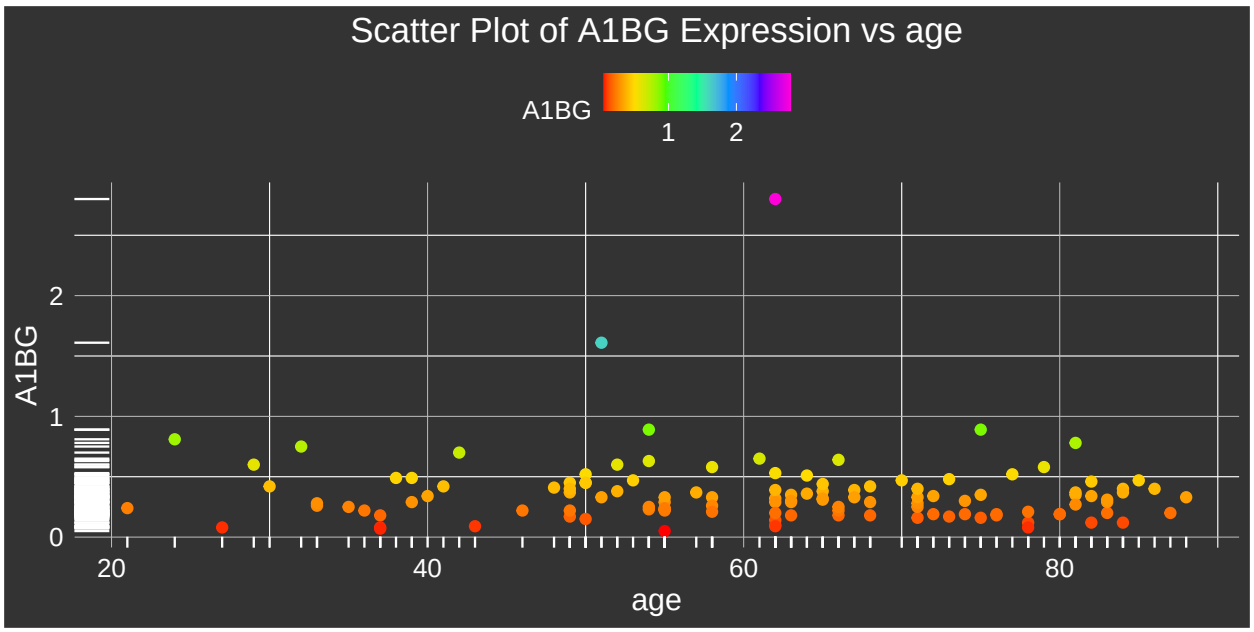
```r
# Define the function to create histogram plots
gene_histogram <- ggplot(data_frame, aes(x = A1BG)) +
    geom_histogram(binwidth = 0.05, color = "#1dddeb", fill = "#1dddeb", alpha = 0.7) +
    labs(x = 'A1BG',
         y = "Frequency",
         title = paste("Histogram of A1BG Expression")) +
    custom_theme

# Save the histogram
ggsave(paste0("plots/plots_A1BG_histogram.png"),
       plot = gene_histogram, width = 10, height = 6)
# Print the histogram
print(gene_histogram)
```



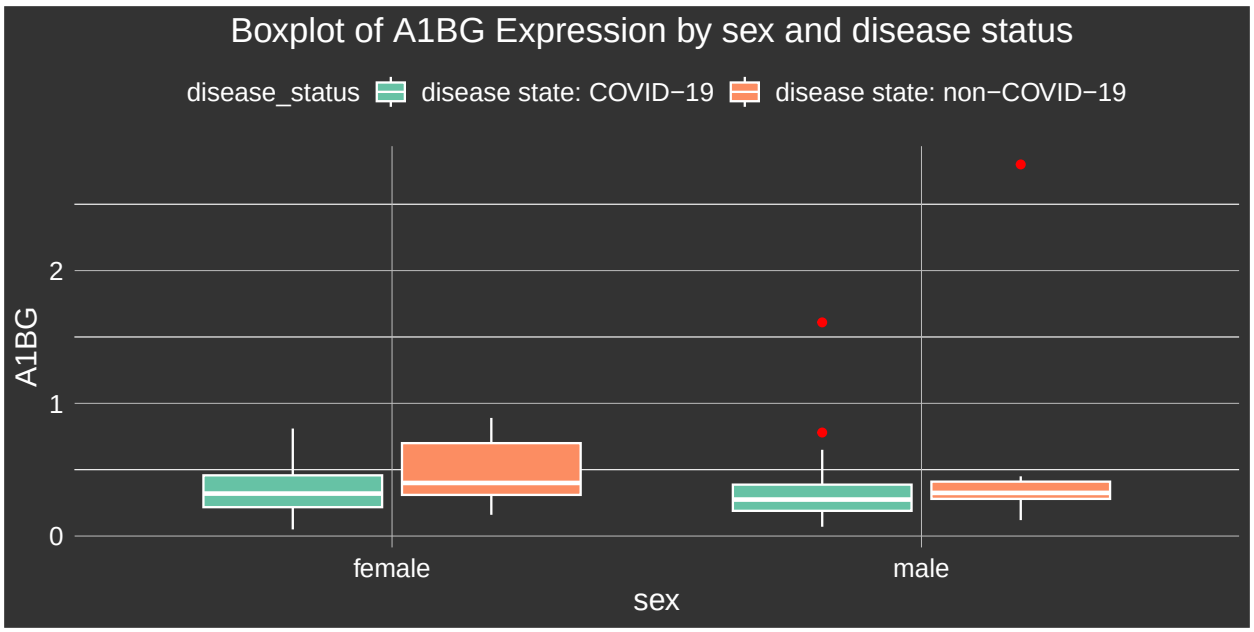

```r
# Scatter Plot with rainbow colors
scatter_plot <- ggplot(data_frame,
                       aes(x = age,
                           y = A1BG,
                           color = A1BG)) +
    geom_point(size = 2) +
    geom_rug(color = "white") +  # Add a rug plot
    scale_color_gradientn(colors = rainbow(7)) +  # Apply rainbow colors
    labs(x = 'age', y = 'A1BG' ,
         title = paste("Scatter Plot of A1BG Expression vs age")) +
    custom_theme

# Save the scatter plot
ggsave(paste0("plots/plots_A1BG_scatter_plot.png"),
       plot = scatter_plot, width = 10, height = 6)
# Print the scatter plot
print(scatter_plot)
```

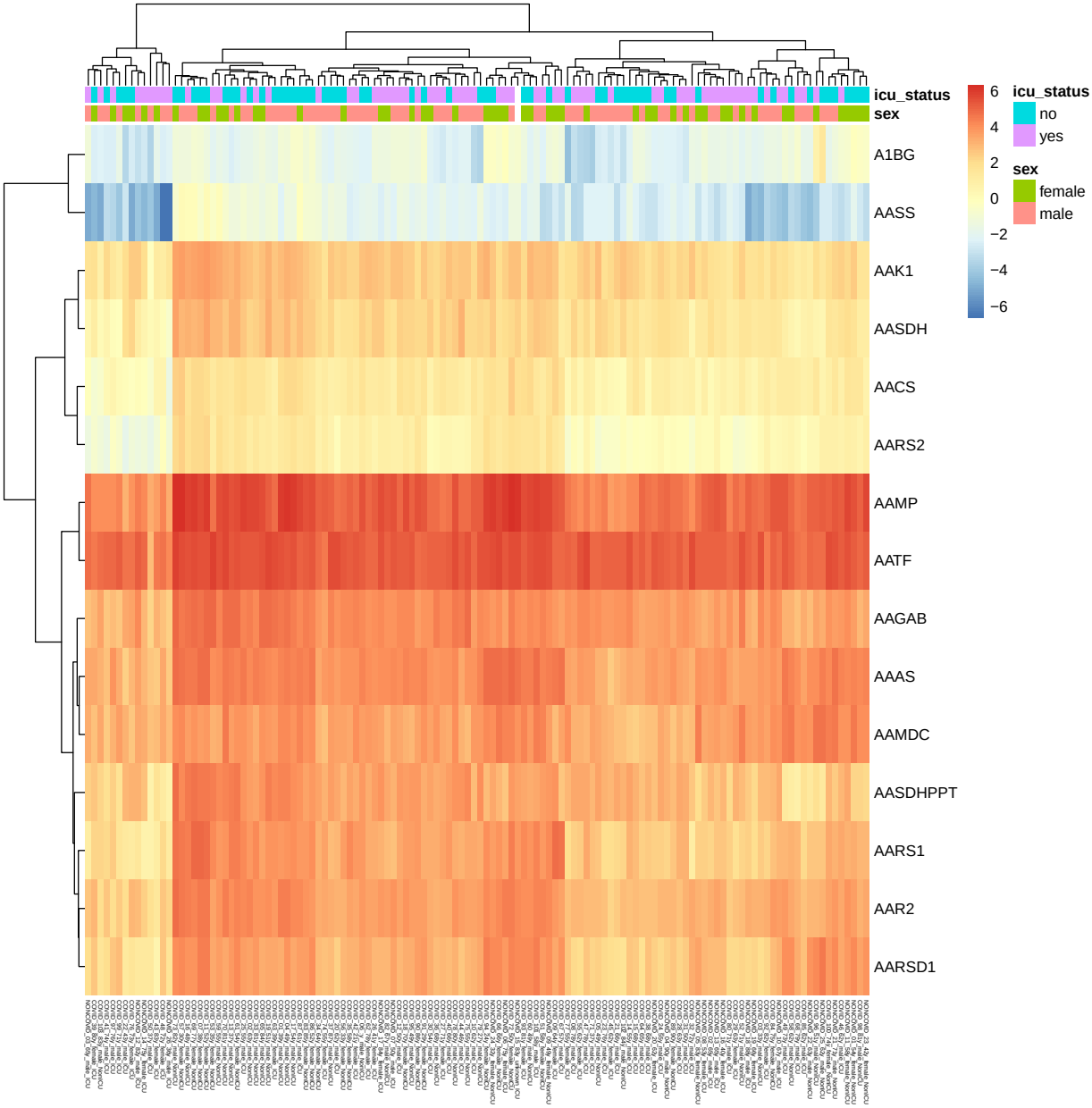



```
# Define labels for the legend
labels <- unique(data_frame[["disease_status"]])
# Create the palette dynamically based on the labels
color <- c("#66c2a5", "#fc8d62")
palette <- setNames(color, labels)
# https://www.statology.org/setnames-r/
# Boxplot
boxplot <- ggplot(data_frame, aes(x = sex,
                                  y = A1BG,
                                  fill = disease_status)) +
    geom_boxplot(outlier.color = "red", color = "white") +
    labs(x = "sex", y = "A1BG", fill = "disease_status" ,
         title = paste("Boxplot of A1BG Expression by sex and disease status ")) +
    scale_fill_manual(values = palette, labels = labels) +
    custom_theme

# Save the boxplot
ggsave(paste0("plots/plots_A1BG_boxplot.png"),
       plot = boxplot, width = 10, height = 6)
# Print the boxplot
print(boxplot)
```




```
genes_heat = as.data.frame(genes_df)
rownames(genes_heat) = genes_data$Genes
# filter out genes with 0 value
# such that we dont get -inf during log2 normalization
genes_heat <- genes_heat[rowSums(genes_heat == 0) == 0, ]
# select 15 genes
genes_heat <- genes_heat[1:15, ]
# perform log2 normalization
log2.genes_heat = log2(genes_heat)
# extract sex and icu_status from datast
annotation_data <- data_frame[, c("sex", "icu_status")]
# create heatmap with pheatmap
heat_map <- pheatmap(log2.genes_heat,
clustering_distance_cols = 'euclidean',
clustering_distance_rows = 'euclidean',
annotation_col = annotation_data,
fontsize_col = 4)
```




```
# save the heatmap
ggsave(paste0("plots/plots_heatmap.png"),
       plot = heat_map, width = 10, height = 10)
```

```
library(ggridges)
# create the ridge plot
ridge<- ggplot(data_frame, aes(x = A1BG, y = sex, fill = disease_status)) +
  geom_density_ridges(scale = 1.5, rel_min_height = 0.01) +
  labs(title = "Ridge Plot of Gene Expression by Sex and Disease Status",
       x = "Gene Expression",
       y = "Sex") +
  scale_fill_manual(values = c("#4BBEE3", "#9e0015")) +
  custom_theme
```

```r
# save the plot
ggsave(paste0("plots/plots_Ridge.png"),
       plot = ridge, width = 10, height = 6)
print(ridge)
```