Final_Submission_AC

Angelique Cortez

2024-08-25

```
set.seed(1001001)
```

```
library(ggplot2)
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(ggpubr)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
library(tidyr)
library(pheatmap)
library(tibble)
```

```
#read the downloaded csvs for the gene expression and metadata
gene_expression <- read.csv("~/Dropbox (Dartmouth College)/Dartmouth Classes/Summer2024/Intro to data a
metadata <- read.csv("~/Dropbox (Dartmouth College)/Dartmouth Classes/Summer2024/Intro to data analysis/
```

```
combined_data <- read.csv("~/Dropbox (Dartmouth College)/Dartmouth Classes/Summer2024/Intro to data anal
```

#Use function to create histogram and

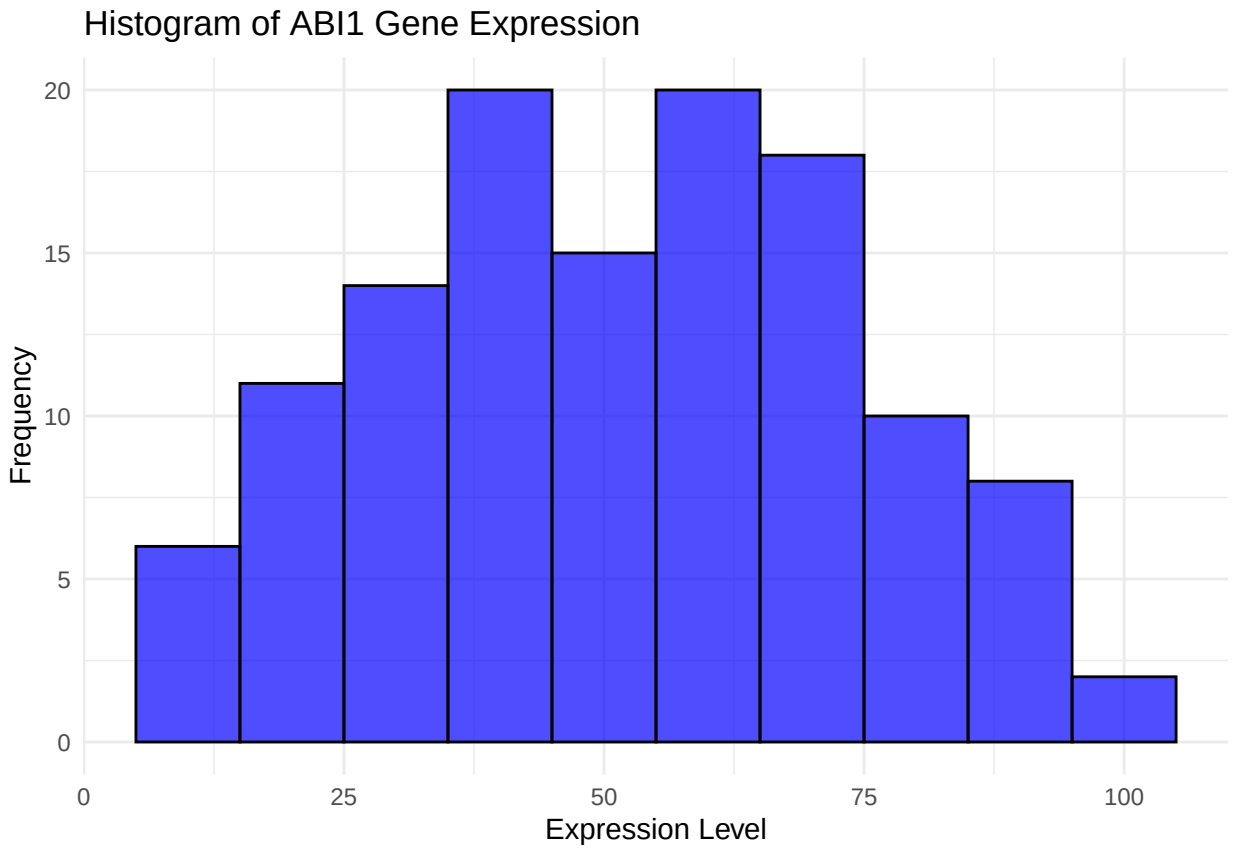
```r
# Write a function that can recreate the plots from submission one and two but fix the unknown variable
generate_plots <- function(data, genes, continuous_covariate, categorical_covariate1, categorical_covari

  for (gene in genes) {
    # Preprocess the data for the gene of interest
    gene_data <- data %>%
      filter(gene == !!gene) %>%
      mutate(
        # Extract age using a regular expression and convert to numeric
        !!continuous_covariate := as.numeric(sub(".*_(\\d{2})y_.*", "\\1", participant_id)),
        # Extract sex, identifying male or female
        !!categorical_covariate1 := sub(".*_(male|female)_.*", "\\1", participant_id)
      ) %>%
      # Filter out participants labeled as "unknown"
      filter(!grepl("unknown", participant_id)) %>%
      select(expression, !!sym(continuous_covariate), !!sym(categorical_covariate1), !!sym(categorical_
      distinct()  # Ensure no duplicate rows

    # Check if the gene_data has the correct number of data points
    print(paste("Number of data points for gene", gene, ":", nrow(gene_data)))

    # Histogram: distribution of expression values
    histogram_plot <- ggplot(gene_data, aes(x = expression)) +
      geom_histogram(binwidth = 10, fill = "blue", color = "black", alpha = 0.7) +
      labs(title = paste("Histogram of", gene, "Gene Expression"),
           x = "Expression Level",
           y = "Frequency") +
      theme_minimal()

    print(histogram_plot)

    # Scatterplot: expression vs continuous covariate, colored by categorical covariate1
    # Calculate the Pearson correlation coefficient and p-value
correlation_test <- cor.test(gene_data[[continuous_covariate]], gene_data$expression, method = "pearson
correlation_value <- round(correlation_test$estimate, 2)
p_value <- round(correlation_test$p.value, 4)

# Create the scatter plot
scatter_plot <- ggplot(gene_data, aes_string(x = continuous_covariate, y = "expression", color = categor
  geom_point(size = 3, alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE, color = "black", linetype = "dashed") +  # Add the correlation
  annotate("text", x = 5, y = max(gene_data$expression) * 0.9,
           label = paste("R =", correlation_value, "\np =", p_value),
           color = "black", size = 5, hjust = 0) +  # Display the correlation coefficient and p-value
  scale_x_continuous(breaks = seq(0, 100, by = 10)) +  # Customize x-axis breaks if needed
  labs(title = paste("Scatterplot of", gene, "Expression vs", continuous_covariate),
       x = "Age (yrs)",  # Update x-axis title here
       y = "Expression Level",
       color = categorical_covariate1) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    axis.title = element_text(size = 12),
```

```
    axis.text = element_text(size = 10),
    legend.position = "right"
  )


    print(scatter_plot)

    # Boxplot: expression by categorical_covariate1, separated by categorical_covariate2
    box_plot <- ggplot(gene_data, aes_string(x = categorical_covariate1, y = "expression", fill = categ
      geom_boxplot() +
      labs(title = paste("Boxplot of", gene, "Expression by", categorical_covariate1, "and", categorica
           x = categorical_covariate1,
           y = "Expression Level",
           fill = categorical_covariate2) +
      theme_minimal() +
      facet_wrap(as.formula(paste("~", categorical_covariate2))) +
      theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))

    print(box_plot)
  }
}
```

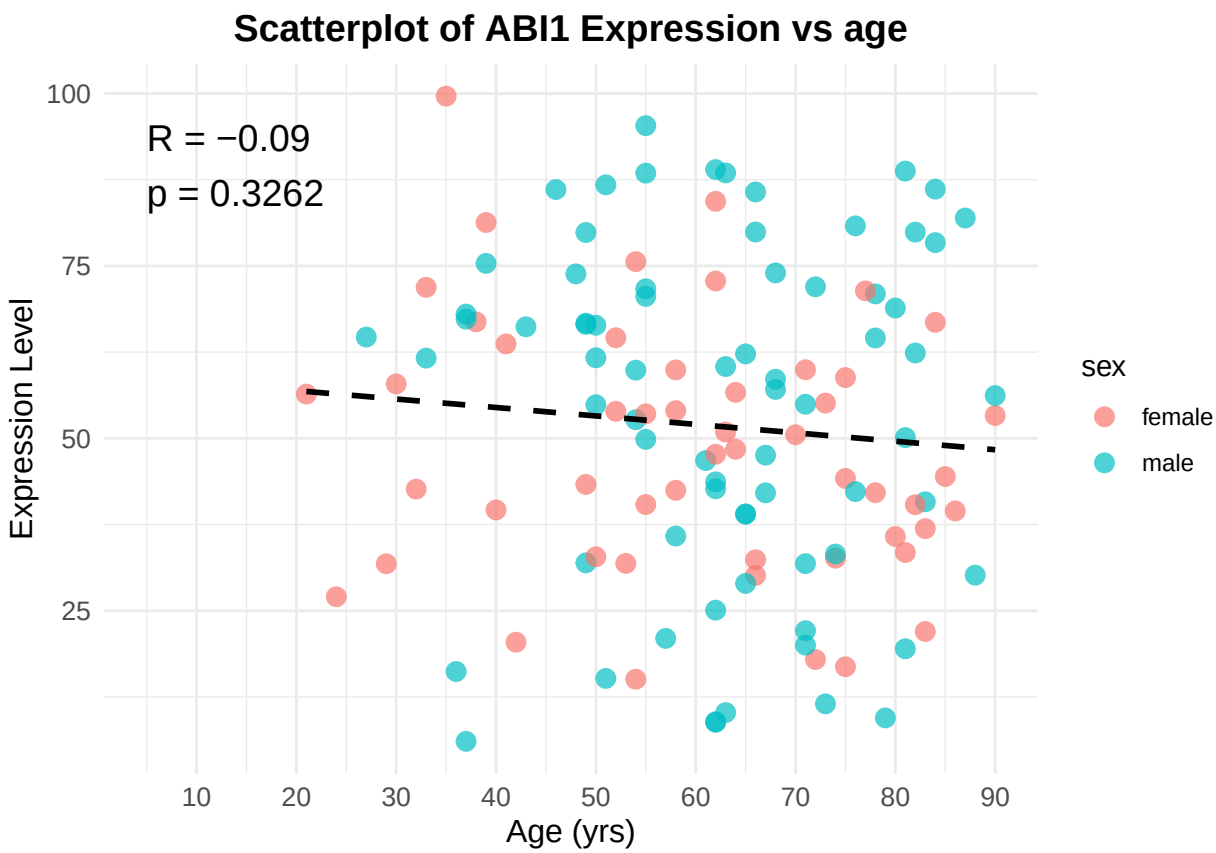
```
generate_plots(
  data = combined_data,
  genes = c("ABI1"),
  continuous_covariate = "age",
  categorical_covariate1 = "sex",
  categorical_covariate2 = "mechanical_ventilation"
)
```

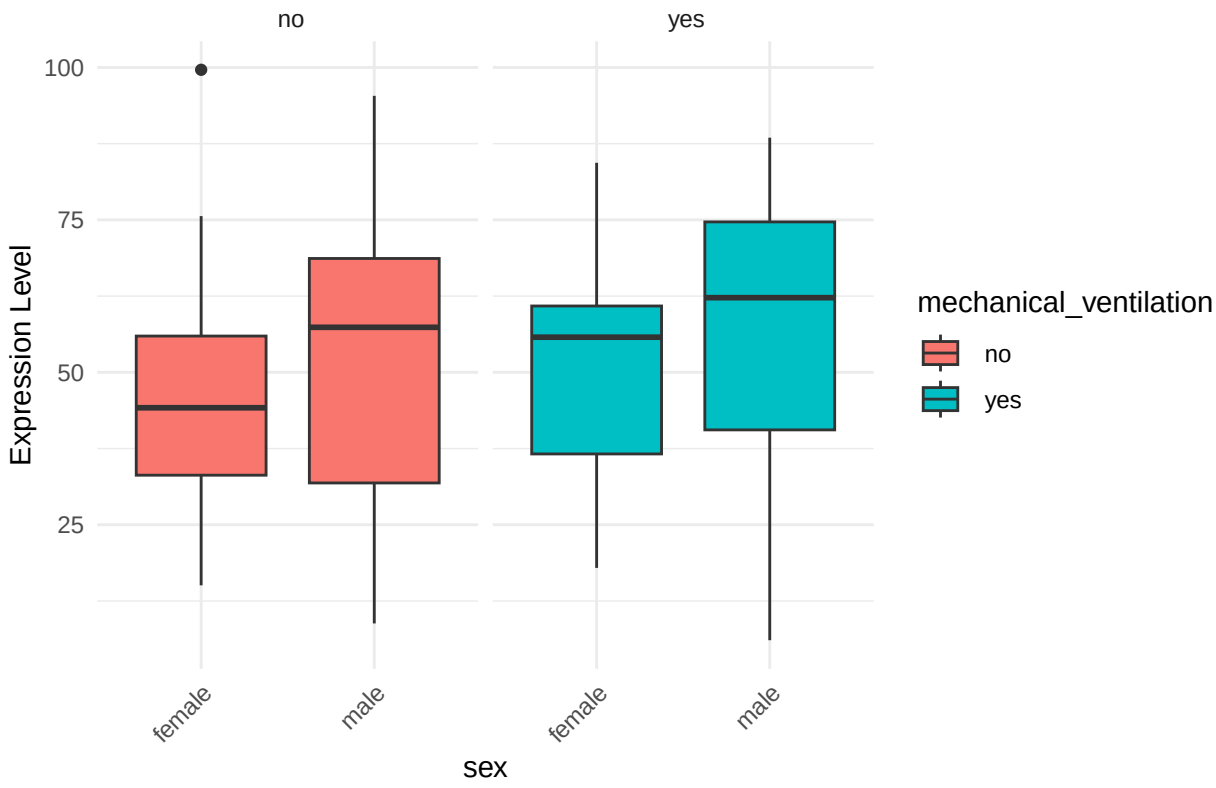
```
## [1] "Number of data points for gene ABI1 : 124"
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




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


Scatterplot of ABI1 Expression vs age
R = −0.09
p = 0.3262
Expression Level
Age (yrs)
sex
female
male


Boxplot of ABI1 Expression by sex and mechanical_ventilation
no
yes
Expression Level
sex
mechanical_ventilation
no
yes

```r
# Remove the row with participant_id 'NONCOVID_15_83y_unknown_ICU'
filtered_data <- combined_data %>%
  filter(participant_id != "NONCOVID_15_83y_unknown_ICU")

# Check the result
#print(filtered_data)
```

#Used inspiration to make summry table from https://stackoverflow.com/questions/59214500/summary-table-of-numeric-and-categorical-data-in-r

```r
calcs <- filtered_data %>%
  mutate(
    sex = forcats::fct_relevel(sex, "male"),  # Adjusted to match the actual levels
    icu_status = forcats::fct_relevel(icu_status, "yes", "no")
  ) %>%
  group_by(icu_status, sex) %>%
  summarise(
     n = n(),# number of participants per row
    mean_age = round(mean(age, na.rm = TRUE), digits = 2),
    sd_age = round(sd(age, na.rm = TRUE), digits = 2),
    mean_ferritin = round(mean(ferritin.ng.ml., na.rm = TRUE), digits = 2),
    sd_ferritin = round(sd(ferritin.ng.ml., na.rm = TRUE), digits = 2),
    mean_crp = round(mean(crp.mg.l., na.rm = TRUE), digits = 2),
    sd_crp = round(sd(crp.mg.l., na.rm = TRUE), digits = 2),
    .groups = "drop"
  ) %>%
  ungroup() %>%
  tidyr::replace_na(list(sd_age = 0, sd_ferritin = 0, sd_crp = 0))
```

```
## Warning: There were 2 warnings in `mutate()`.
## The first warning was:
## i In argument: `sex = forcats::fct_relevel(sex, "male")`.
## Caused by warning:
## ! 1 unknown level in `f`: male
## i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

```
## Warning: There were 22 warnings in `summarise()`.
## The first warning was:
## i In argument: `mean_age = round(mean(age, na.rm = TRUE), digits = 2)`.
## i In group 1: `icu_status = no` and `sex = female`.
## Caused by warning in `mean.default()`:
## ! argument is not numeric or logical: returning NA
## i Run `dplyr::last_dplyr_warnings()` to see the 21 remaining warnings.
```

```r
calcs
```

```
## # A tibble: 4 x 9
##   icu_status sex           n mean_age sd_age mean_ferritin sd_ferritin mean_crp
##   <fct>      <fct>     <int>    <dbl>  <dbl>         <dbl>       <dbl>    <dbl>
## 1 " no"      " female"  2700       NA   18.2            NA        983.       NA
## 2 " no"      " male"    3200       NA   16.6            NA       1116.       NA
## 3 " yes"     " female"  2400       NA   16              NA       1100.       NA
```

```
## 4 " yes"     " male"    4100       NA   12.3            NA        928.       NA
## # i 1 more variable: sd_crp <dbl>
```

```
# Format the table for LaTeX output using kableExtra
latex_table <- calcs %>%
  kbl(format = "latex", booktabs = TRUE, caption = "Summary Statistics of Covariates") %>%
  kable_styling(latex_options = "hold_position")
# Print the table (this can be copy-pasted into your LaTeX document)
latex_table
```

Table 1: Summary Statistics of Covariates

icu_status	sex	n	mean_age	sd_age	mean_ferritin	sd_ferritin	mean_crp	sd_crp
no	female	2700	NA	18.15	NA	983.14	NA	68.76
no	male	3200	NA	16.62	NA	1116.34	NA	106.34
yes	female	2400	NA	16.00	NA	1100.11	NA	112.64
yes	male	4100	NA	12.27	NA	928.44	NA	99.82

#Create heatmap

```
# Check if column names of gene_expression match participant IDs in metadata
identical(colnames(gene_expression), metadata$participant_id)
```

```
## [1] FALSE
```

```
# Set the gene column as row names
rownames(gene_expression) <- gene_expression$gene

# Remove the gene column as it's now redundant
gene_expression <- gene_expression[, !names(gene_expression) %in% "gene"]
```

```
# Check the first few rows to confirm
#head(gene_expression)

# Check the row names
#head(rownames(gene_expression))
```

```
# IDs in gene expression but not in metadata
missing_in_metadata <- setdiff(colnames(gene_expression), metadata$participant_id)
# IDs in metadata but not in gene expression
missing_in_gene_expression <- setdiff(metadata$participant_id, colnames(gene_expression))

# Print the results
missing_in_metadata
```

```
## [1] "X"                       "COVID_06_.y_male_NonICU"
```

```
missing_in_gene_expression
```

```
## [1] "COVID_06_:y_male_NonICU"
```

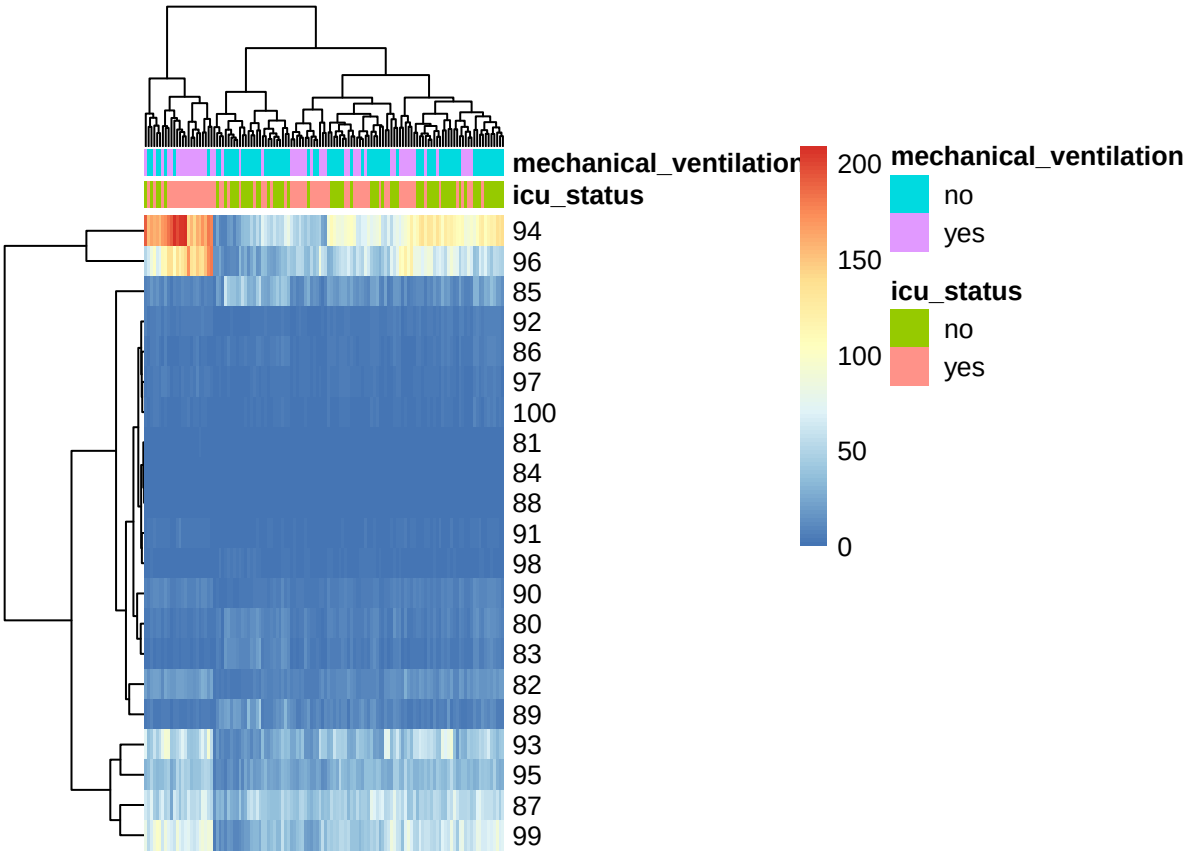
```r
# Normalize column names in gene_expression
colnames(gene_expression) <- gsub("[[:punct:]]", "_", tolower(trimws(colnames(gene_expression))))

# Normalize participant_id in metadata
metadata$participant_id <- gsub("[[:punct:]]", "_", tolower(trimws(metadata$participant_id)))

# Check again for discrepancies
missing_in_metadata <- setdiff(colnames(gene_expression), metadata$participant_id)
missing_in_gene_expression <- setdiff(metadata$participant_id, colnames(gene_expression))

# Print the results
missing_in_metadata
```

```
## [1] "x"
```

```r
missing_in_gene_expression
```

```
## character(0)
```

```r
# Prepare metadata for annotations
annotation_data <- metadata %>%
  select(participant_id, icu_status, mechanical_ventilation) %>%
  column_to_rownames("participant_id")

# Ensure gene_expression columns match metadata participant IDs
gene_expression <- gene_expression[, colnames(gene_expression) %in% metadata$participant_id]

# Plot the heatmap without specifying annotation colors
pheatmap(
  gene_expression[80:100, ],
  annotation_col = annotation_data,
  clustering_distance_cols = "euclidean",
  clustering_distance_rows = "euclidean",
  show_rownames = TRUE,
  show_colnames = FALSE
)
```



#PCA plot by

```
# Select numeric columns for PCA
pca_data <- filtered_data %>%
  select(where(is.numeric)) %>%
  na.omit()  # Remove any rows with NA values
# Check for NA values
if(any(is.na(pca_data))) {
    pca_data <- na.omit(pca_data)
    message("NA values found and removed from the data.")
}

# Identify and remove columns with zero variance
constant_columns <- sapply(pca_data, function(x) var(x, na.rm = TRUE) == 0)
pca_data_clean <- pca_data[, !constant_columns]

# Perform PCA on the cleaned data
pca_result <- prcomp(pca_data_clean, scale. = TRUE)

# Create a dataframe with the PCA results and the categorical variable
pca_df <- data.frame(PC1 = pca_result$x[,1],
                     PC2 = pca_result$x[,2],
                     CategoricalVariable = factor(filtered_data$icu_status, levels = unique(filtered_da
```

```
# Plot the PCA
ggplot(pca_df, aes(x = PC1, y = PC2, color = CategoricalVariable)) +
    geom_point(size = 3, alpha = 0.8) +
```

```
    stat_ellipse(type = "norm", linetype = 2) +  # Add ellipses around groups
    labs(title = "PCA Plot",
         x = paste0("Principal Component 1 (", round(pca_result$sdev[1]^2 / sum(pca_result$sdev^2) * 10
         y = paste0("Principal Component 2 (", round(pca_result$sdev[2]^2 / sum(pca_result$sdev^2) * 10
         color = "ICU Status") +  # Change the legend title here
    theme_minimal() +
    theme(legend.position = "bottom")  # Place legend at the bottom
```