QBS103 Final Project

2024-08-19

Import data

```
genes <- read.csv('/Users/isabellabaldacci/Desktop/QBS103/Project/QBS103_GSE157103_genes.csv',
                  sep = ',', header = TRUE)
metadata <- read.csv('/Users/isabellabaldacci/Desktop/QBS103/Project/QBS103_GSE157103_series_matrix.csv
                     header = TRUE, stringsAsFactors = TRUE)
```

Clean up metadata

```
## cleaning up metadata
# metadata were removed if they were not helpful or if they had too many unknowns
metadata_clean <- dplyr::select(metadata,
                                -c("X.Sample_submission_date","channel_count",
                                   "status","last_update_date","type",
                                   "channel_count","source_name_ch1",
                                   "organism_ch1","apacheii",
                                   "ddimer.mg.l_feu.",
                                   "lactate.mmol.l.","fibrinogen","sofa"))
```

Transpose Genes dataframe

```
#transposing the genes x samples matrix to samples x genes
rownames(genes) <- genes$X #assigning row names
genes_t <- as.data.frame(t(dplyr::select(genes, -c('X')))) #transpose as a dataframe

genes_t$participant_id <- rownames(genes_t) #assigning participant id column for later joining
#which(genes_t$participant_id == "COVID_06_.y_male_NonICU") #finding column with mismatch to metadata
genes_t$participant_id[6] <- "COVID_06_:y_male_NonICU" #reassinging value to match metadata
```

```
#combining the samples x genes matrix to the metadata
all_data <- dplyr::inner_join(genes_t, metadata_clean, by = 'participant_id')

#finding all values that would cause issues later
findNAs <- function(value) {
    new_val <- ifelse(((value == 'unknown') |
                           (value == ' unknown') |
                           (value == ' :') |
                           (value == " >89")),
```

```
                      NA,
                      value)
    return(new_val)
}

clean_data <- as.data.frame(apply(all_data, 2, FUN = findNAs))
```

```
# USED FROM CLASS NOTES
# Define a function to calculate a mean or a median
contSummary <- function(x,normal = T) {

  #x <- as.numeric(x, na.rm = TRUE)
  # Calculate mean (sd) if normally distributed (the default)
  if (normal == T) {
      # Calculate individual values
    myMean <- round(mean(x),2)
    mySD <- round(sd(x),2)
    # Combine values
    paste0(myMean,' (',mySD,')')
  }
  # Calculate median (IQR) if non-normally distributed
  else {
    # Calculate individual values
    myMedian <- round(median(x, na.rm = TRUE), 2)
    myIQR <- round(IQR(x), 2)
    #myIQR1 <- round(quantile(x,1/4),digits = 2)
    #myIQR2 <- round(quantile(x,3/4),digits = 2)
    # Combine values
    paste0(myMedian,' [',myIQR,']')
  }
}
```

Summary Statistics

```
#omitted NAs, which made the dim 97 instead of 126
table_data <- na.omit(clean_data %>%
  dplyr::select('mechanical_ventilation',
                'disease_status',
                'sex',
                'crp.mg.l.',
                'ferritin.ng.ml.',
                'procalcitonin.ng.ml..'))

#ensure categorial are factors
table_data$disease_status <-
  factor(table_data$disease_status,
         levels = unique(table_data$disease_status))

table_data$mechanical_ventilation <-
  factor(table_data$mechanical_ventilation,
         levels = unique(table_data$mechanical_ventilation),
```

```
         labels = c('Mechanical Ventilator',
                    'No Mechanical Ventilator'))

table_data$sex <- factor(table_data$sex,
                         levels = unique(table_data$sex),
                         labels = c('Male', 'Female'))

#ensure continuous are numeric
table_data$crp.mg.l. <- as.numeric(table_data$crp.mg.l.)
table_data$ferritin.ng.ml. <- as.numeric(table_data$ferritin.ng.ml.)
table_data$procalcitonin.ng.ml.. <-
  as.numeric(table_data$procalcitonin.ng.ml..)
```

Categorical Variables

```
#calculate n and p by using table and prop.table for all categorial variables
sn <- as.data.frame(table(table_data$sex,
                          table_data$disease_status))
sp <- as.data.frame(prop.table(table(table_data$sex,
                                     table_data$disease_status))) %>%
  dplyr::rename('Percent' = 'Freq') #rename for later joining

sex <- dplyr::inner_join(sn, sp, by = join_by(Var1, Var2))

mn <- as.data.frame(table(table_data$mechanical_ventilation,
                          table_data$disease_status))
mp <- as.data.frame(prop.table(table(table_data$mechanical_ventilation,
                                     table_data$disease_status))) %>%
  dplyr::rename('Percent' = 'Freq')

mech <- dplyr::inner_join(mn, mp, by = join_by(Var1, Var2))

dsn <- as.data.frame(table(table_data$disease_status))
dsp <- as.data.frame(prop.table(table(table_data$disease_status))) %>%
  dplyr::rename('Percent' = 'Freq')

ds <- dplyr::inner_join(dsn, dsp, by = join_by(Var1))
ds$Var2 <- ds$Var1
ds$Var1 <- c('Disease State', 'Disease State') #rename for later joining
ds <- ds[,c(1,4,2,3)] #reordering

#combine all cat vars, create column as_str with nice format
#pivot wider to get right format for kable
all_cat <- dplyr::bind_rows(sex, mech) %>%
  dplyr::bind_rows(ds) %>%
  dplyr::mutate(as_str = paste0(Freq,' (',
                                round(Percent*100,1),')')) %>%
  dplyr::select(Var1, Var2, as_str) %>%
  tidyr::pivot_wider(names_from = Var2, values_from = as_str)

#rename to get right format for kable
```

```r
all_cat_final <- all_cat %>%
  dplyr::rename('Variables' = 'Var1') %>%
  dplyr::rename('Covid.Positive' = 'disease state: COVID-19',
                'Covid.Negative' = 'disease state: non-COVID-19') %>%
  dplyr::select(Variables, Covid.Positive, Covid.Negative)
```

Continuous Variables

```r
#calculate values and get good format for kable for continous vars
cont_vars_pos <- table_data %>%
  dplyr::filter(disease_status == 'disease state: COVID-19') %>%
  dplyr::select('crp.mg.l.', 'ferritin.ng.ml.', 'procalcitonin.ng.ml..') %>%
  apply(MARGIN  = 2, FUN = function(x) {contSummary(x, normal = F)})

cont_vars_neg <- table_data %>%
  dplyr::filter(disease_status == 'disease state: non-COVID-19') %>%
  dplyr::select('crp.mg.l.', 'ferritin.ng.ml.', 'procalcitonin.ng.ml..') %>%
  apply(MARGIN  = 2, FUN = function(x) {contSummary(x, normal = F)})

#make sure they are dfs
pos <- as.data.frame(cont_vars_pos)
neg <- as.data.frame(cont_vars_neg)
```

```r
#merge the dfs by row names
continuous_variables_table <- merge(cont_vars_pos,
                                    cont_vars_neg,
                                    by= 'row.names')

#create continuous variables table with nice names for latex
cvt <- continuous_variables_table %>%
  dplyr::rename(Variables = Row.names,
                Covid.Positive = x, Covid.Negative = y)
cvt$Variables <- c('CRP (mg/L) Median [IQR]',
                   'Ferritin (ng/mL) Median[IQR]',
                   'Procalitonin (ng/mL) Median[IQR]')
```

Final Latex Table Generation

```r
#combine categorical and continuous vars for kable func
tableLatex <- dplyr::bind_rows(cvt, all_cat_final)
#add spacer rows
tableLatex <- tableLatex %>% dplyr::add_row(Variables = 'Sex N(%)',
                                Covid.Positive = ' ',
                                Covid.Negative = ' ',
                                .before = 4) %>%
  dplyr::add_row(Variables = 'Mechanical Ventilation N(%)',
                                Covid.Positive = ' ',
                                Covid.Negative = ' ',
                                .before = 7)
```

```r
#reorder rows
tableLatex <- tableLatex[c(10, 4,5,6,7,8,9, 1,2,3),]
```

```r
#generate tab2
tab2 <- kable(x = tableLatex, caption = 'Summary Table',
              format = 'latex',booktabs = T,
              row.names = FALSE,
              col.names = c('Variable','Covid Positive', 'Covid Negative'),
              align = c('l','r'),escape = T) %>%
  add_indent(c(3,4,6,7), all_cols = FALSE, target_cols = 1 )
```

Plots: Histogram, Scatterplot, Boxplot

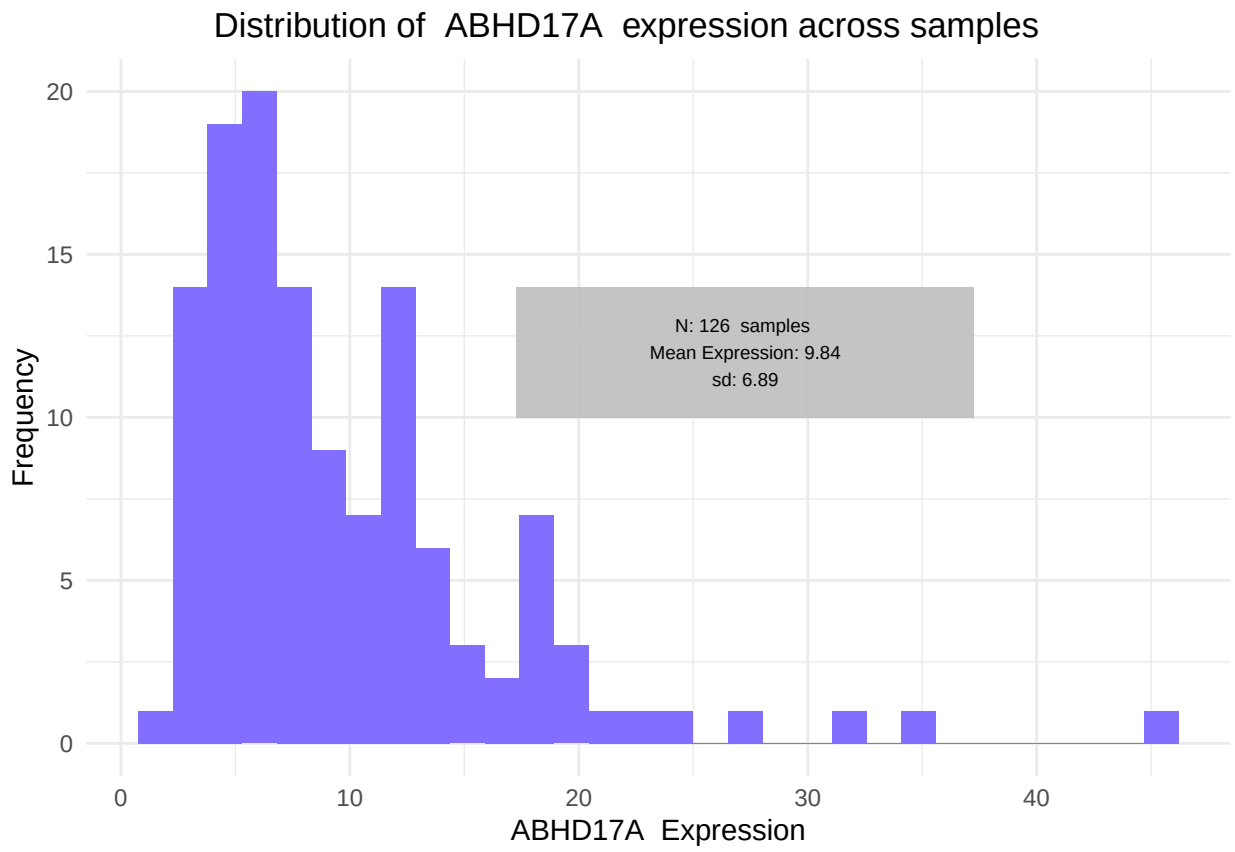
```r
#define function for plotting
plots <- function(df, genes_list, cont_cov, cat_cov) {
  #filter dataframe (received error message indicating to use "all of")
  df_filtered <- df %>% dplyr::select(participant_id,
                                      all_of(genes_list),
                                      all_of(cont_cov),
                                      all_of(cat_cov))

  #cast to long
  df_filtered_long <- df_filtered %>%
    tidyr::pivot_longer(cols = genes_list, names_to = 'Gene',
                        values_to = 'Expression')

  for (gene in genes_list){ #for gene in genes list

    one_gene <- df_filtered_long %>%
      dplyr::filter(Gene == gene) #get dataframe for one gene
    one_gene$Expression <- as.numeric(na.omit(one_gene$Expression))

    #get data for plotting, need mean, sd, and the
    #75th quantile for positioning annotation box
    mean_expression <- round(mean(one_gene$Expression), 2)
    sd_expression <- round(sd(one_gene$Expression), 2)
    x_min_box <- quantile(one_gene$Expression, 3/4)
    n <- length(one_gene$Expression)

    #create labels
    hist_title <- paste('Distribution of ',
                        gene, ' expression across samples')
    hist_x <- paste(gene, ' Expression')

    #plot histogram using theme parameters from previous assignment
    hist <- one_gene %>% ggplot(aes(Expression)) +
      geom_histogram(bins = 30, fill = 'slateblue1') +
      theme_minimal() +
      theme(plot.title = element_text(hjust = .4)) +
      labs(title = hist_title,
       x = hist_x,
```

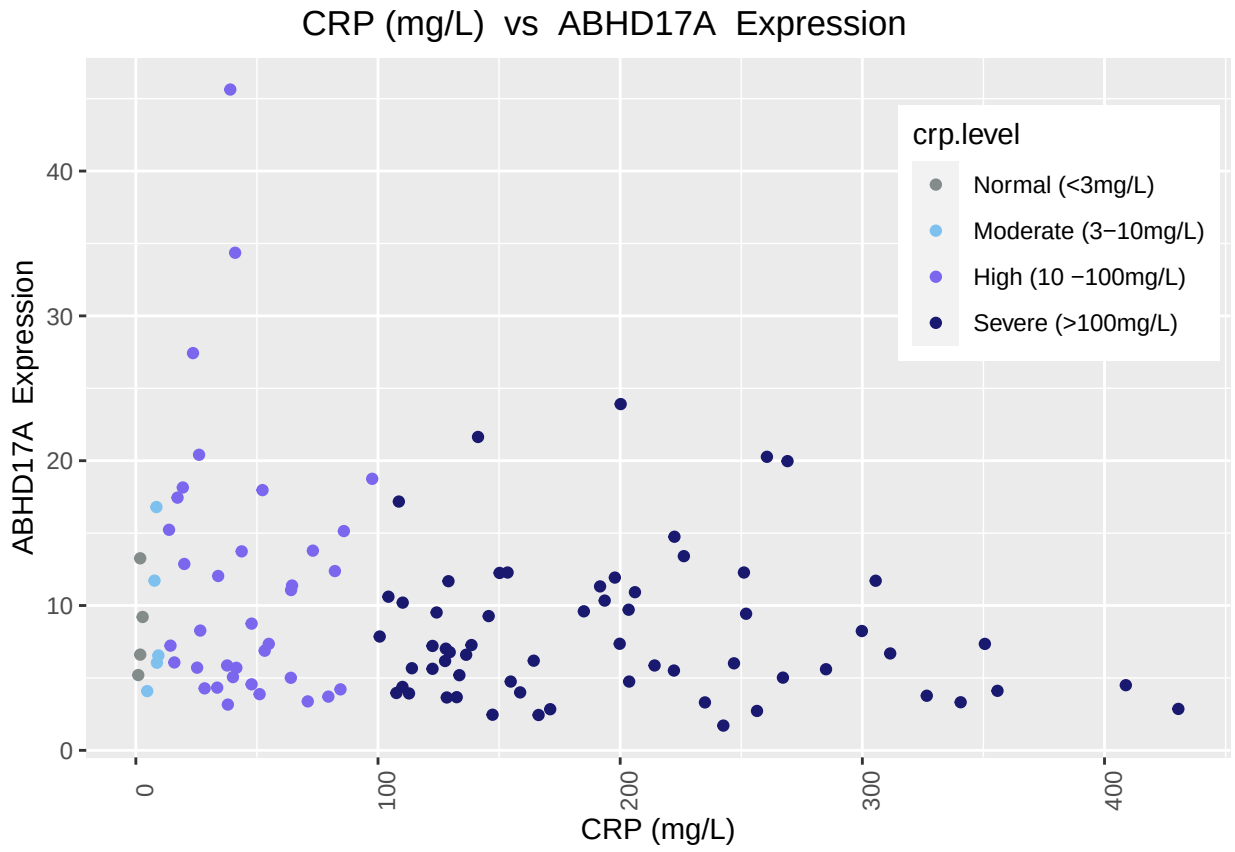
```
      y = 'Frequency') +
    scale_color_manual(values = c('slateblue1')) +
    annotate('rect',
         xmin = x_min_box + 5,
         xmax = x_min_box + 25,
         ymin = 10,
         ymax = 14, fill = 'grey', alpha = .9)+
annotate(geom = 'text',
         x = x_min_box + 15,
         y = 12,
         label = paste('N:', n,' samples',
                       '\n Mean Expression:',
                       mean_expression,
                       '\nsd:', sd_expression), size = 2.5)
  #print to display the histogram
  print(hist)


  #filter the data to create the scatter plot
  #to avoid warnings of coerced NAs and to ensure cont_cov is numeric
  scatter_filtered <- one_gene %>%
    filter(one_gene[[cont_cov]] != ' unknown')
  scatter_filtered[cont_cov] <-
    as.numeric(scatter_filtered[[cont_cov]])
  #assign the levels for crp
  if (cont_cov == 'crp.mg.l.') {
    scatter_filtered$crp.level <-
      cut(scatter_filtered$crp.mg.l.,
          breaks = c(0, 3, 10, 100, 500000),
          labels = c('normal', 'moderate', 'high', 'severe'))
  }

  #set color palette and titles
  colorPalette_scatter <- c('azure4', 'skyblue2', 'slateblue2', 'midnightblue')

  x_scatter_split <- strsplit(x = cont_cov, split = '\\.')[[1]][1]
  x_scatter <- paste(toupper(x_scatter_split), '(mg/L)')
  title_scatter <- paste(x_scatter, ' vs ', gene, ' Expression')
  y_scatter <- paste(gene, ' Expression')

  #create scatterplot with parameters from previous assignment
  scatter <- scatter_filtered %>% ggplot(aes(x = .data[[cont_cov]],
                                             y = Expression,
                                             color = crp.level)) +
    geom_point(na.rm = TRUE) +
    theme(
      axis.text.x = element_text(angle=90),
      plot.title = element_text(hjust = .4),
      legend.position = c(.85,.75)) +
    labs(title = title_scatter, x = x_scatter, y = y_scatter)  +
    scale_color_manual(labels =
                         c('Normal (<3mg/L)', 'Moderate (3-10mg/L)',
                           'High (10 -100mg/L)', 'Severe (>100mg/L)'),
```

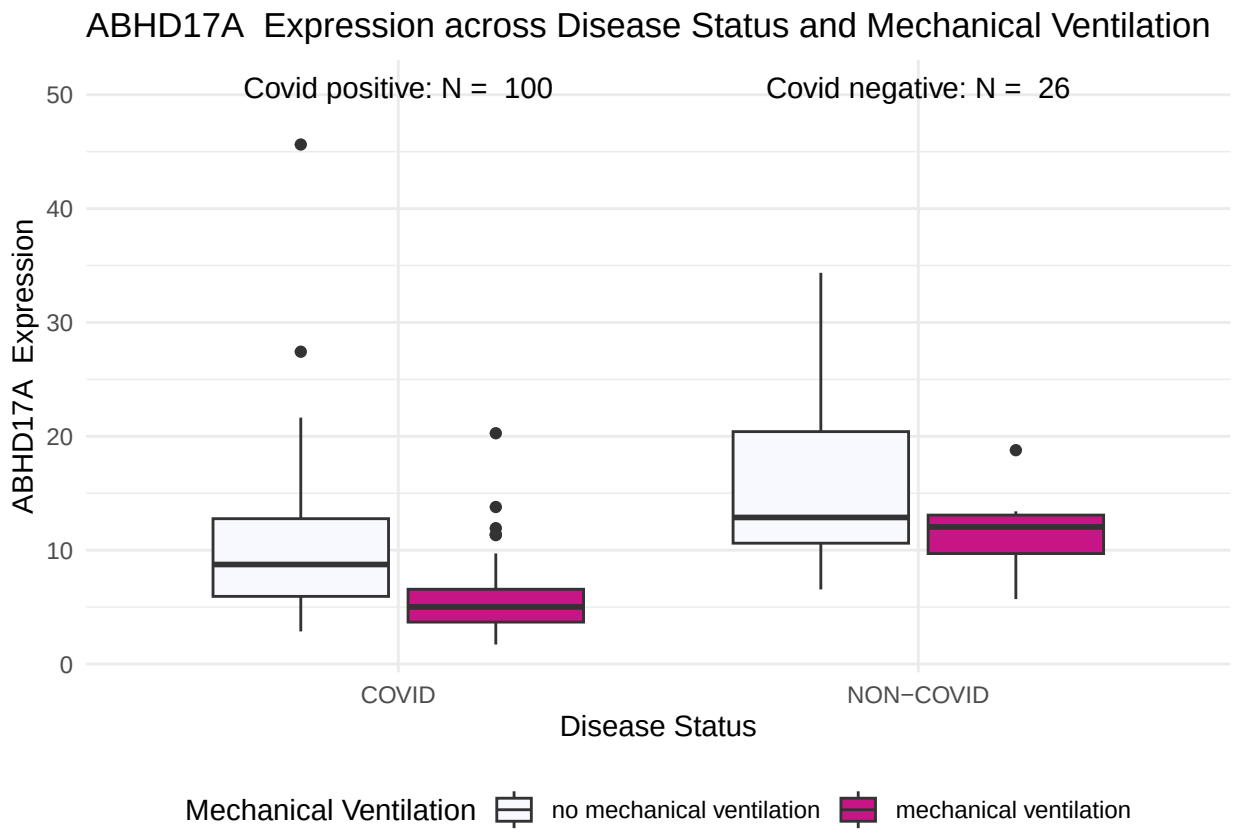
```
                        values = colorPalette_scatter)

    #print scatterplot
    print(scatter)


    #create titles for box plot
    title_box <- paste(gene,
                       ' Expression across Disease Status and Mechanical Ventilation')
    x_lab_box <- 'Disease Status'
    y_lab_box  <- paste(gene, ' Expression')
    #get num_covid positive and negative for annotations
    n_covid_pos <- length(dplyr::filter(
      one_gene,
      disease_status == 'disease state: COVID-19')$Expression)
    n_covid_neg <- length(dplyr::filter(
      one_gene,
      disease_status != 'disease state: COVID-19')$Expression)
    #set the y value for the annotations as
    #the max of expression so that it will not cover the data
    y_annot <- max(one_gene$Expression)
    labels_legend <- c('no mechanical ventilation',
                       'mechanical ventilation')

    #generate box plot using same parameters as previous assignment
    box_filtered <- one_gene %>%
      dplyr::filter(one_gene[[cat_cov[2]]] != ' unknown')
    box <- box_filtered %>%
      ggplot(aes_string(x = cat_cov[1], y = 'Expression', fill = cat_cov[2])) +
      geom_boxplot() +
      theme_minimal() +
      theme(legend.position = 'bottom') +
      labs(title = title_box,
           x = x_lab_box,
           y = y_lab_box,
           colour = 'Mechanical Ventilation') +
      guides(fill=guide_legend(title="Mechanical Ventilation")) +
      scale_fill_manual(labels = labels_legend,
                        values = c('ghostwhite', 'mediumvioletred'))+
      scale_x_discrete(labels = c('COVID', 'NON-COVID')) +
      annotate(geom = 'text', x = 1, y = y_annot + 5,
               label = paste('Covid positive: N = ', n_covid_pos),
               color = 'black')+
      annotate(geom = 'text', x = 2, y = y_annot + 5,
               label = paste('Covid negative: N = ', n_covid_neg),
               color = 'black')
    print(box)

  }
  }
```

```
#a few warnings about deprecated methods, did not cause
#problems so decided to suppress for final output
```

```
suppressWarnings({ plots(clean_data, c('ABHD17A'),
      c('crp.mg.l.'),
      c('disease_status', 'mechanical_ventilation')) } )
```



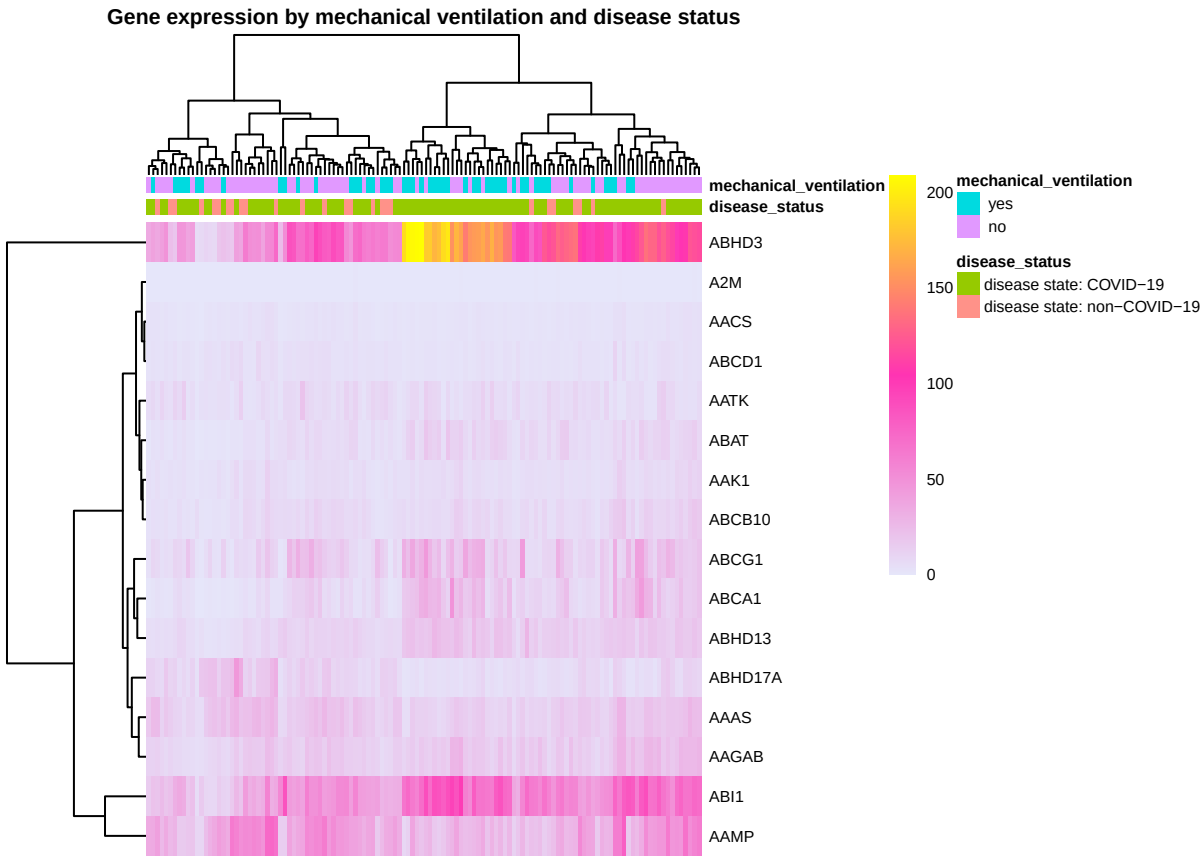

CRP (mg/L) vs ABHD17A Expression
ABHD17A Expression
CRP (mg/L)
crp.level
Normal (<3mg/L)
Moderate (3−10mg/L)
High (10 −100mg/L)
Severe (>100mg/L)
0
10
20
30
40
0
100
200
300
400



Heatmap

```
#define interesting genes
interesting_genes <- c('AAAS', 'AACS', 'A2M',
                       'AAGAB','AAK1','ABCD1',
                       'ABCG1', 'ABHD17A',
                       'ABHD17A','ABI1',
                       'AAMP', 'AATK','ABAT',
                       'ABCA1','ABCB10',
                       'ABHD13', 'ABHD3')

#select genes, ensure they are numeric
genesHM <- as.data.frame(clean_data %>%
  dplyr::select(all_of(interesting_genes)) %>%
  apply(MARGIN = 2, as.numeric))

#add columns for categorical vars, make sure factors
genesHM$mechanical_ventilation <-
  factor(clean_data$mechanical_ventilation,
         levels = unique(clean_data$mechanical_ventilation))

genesHM$disease_status <-
  factor(clean_data$disease_status,
```

```
         levels = unique(clean_data$disease_status))

#transpose gene dataframe
tgenesHM <- as.data.frame(t(genesHM[,1:16]))

#select annotation columns
annotationColumns <- genesHM %>%
  dplyr::select('disease_status', 'mechanical_ventilation')
#set rownames to match colnames of genes mat
row.names(annotationColumns) = colnames(tgenesHM)

#define plotting colors
annotationColors <- list(Disease_status =
                           c('COVID-19' = 'lightsalmon',
                             'non-COVID-19' = 'slateblue4'),
                         Mech_Ventilation =
                           c('yes' = 'red',
                             'no' = 'green'))

#heatmap with clustering
pheatmap(tgenesHM,
         color = colorRampPalette(c('lavender', 'maroon1', 'yellow'))((500)),
         cluster_rows = T,
         cluster_cols = T,
         annotation_col = annotationColumns,
         annotation_colors = annotationColors,
         show_colnames = FALSE,
         main = 'Gene expression by mechanical ventilation and disease status',
         fontsize = 6)
```



Final plot: diverging barchart

```
#plot diverging
#inspired by: https://www.tutorialspoint.com/ggplot2/ggplot2_diverging_charts.htm
plot_diverging <- function(df, gene1) {

  #select gene of interest and ensure numeric
  plotting_df <- df %>%
    dplyr::select(all_of(gene1))
  plotting_df$my_gene <- as.numeric(df[[gene1]])

  #create a participant column
  plotting_df$participant <- seq(1:length(plotting_df$my_gene))

  #add disease status col to plotting df as factor
  plotting_df$disease_status <- factor(df$disease_status,
                                       levels = unique(df$disease_status))
  #add z-score column
  plotting_df <- plotting_df %>%
    dplyr::mutate(z = (my_gene - mean(my_gene))/sd(my_gene)) %>%
    dplyr::arrange(z)

  #factor participants to maintain order
  plotting_df$participant <- factor(plotting_df$participant,
                                    levels = unique(plotting_df$participant))
  #plot final by participant and z score
  final_plot <-
```

```
    ggplot(plotting_df, aes(x = participant, y = z, label = z)) +
    geom_bar(stat='identity', aes(fill = disease_status), width = .9) +
    scale_fill_manual(name = 'Disease State',
                      labels = c('COVID-19', 'NON-COVID-19'),
                      values = c("disease state: COVID-19"="plum2", "disease state: non-COVID-19"="limeg
    labs(title= paste("Z-scored Expression for", gene1),
       y = 'Z-scored expression',
       x = 'Participant') +
    theme(axis.text.y = element_blank()) +
    coord_flip()

  return(final_plot)
}
```

```
#create 4 plots and plot together
a <- plot_diverging(clean_data, 'ABI1')
b <- plot_diverging(clean_data, 'ABHD3')
c <- plot_diverging(clean_data, 'ABHD17A')
d <- plot_diverging(clean_data, 'AAMP')

ggpubr::ggarrange(a,b,c,d, common.legend = T)
```