

Project 1: Function to Summarize and Plot Gene Expression

2024-08-07

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)
```

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", "ferritin.ng.ml.",  
                                   "ddimer.mg.l_feu.", "procalcitonin.ng.ml..",  
                                   "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" #reassigning value to match metadata
```

Combining metadata and genes dataframe

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

Write 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  
  
    #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,  
                                                                    '\n sd:', 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')
title_scatter <- paste(toupper(cont_cov), ' vs ', gene, ' Expression')
x_scatter <- paste(toupper(cont_cov), '(mg/L)')
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)

#creat titles for box plot
title_box <- paste(gene, ' Expression across ', cat_cov[1], ' and ', cat_cov[2])
x_lab_box <- cat_cov[1]
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(paste('no ', cat_cov[2]), cat_cov[2])

#generate box plot using same parameters as previous assignment
box <- one_gene %>% 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) +

```

```

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)

#just an idea I had for putting it all together
#all_plots <- ggpubr::ggarrange(hist, ggarrange(scatter, box, ncol = 2,
#labels = c('B', 'C')), nrow = 2, labels = c('A'))
#plotlist = list(box, scatter, hist), ncol = 3, nrow = 1, widths = c(5,5,5))
#print(all_plots)
}
}

```

Observing expression patterns across three genes, one continuous covariate and two categorical covariates

Chosen genes: ABI1, AAMP, ABHD17A

Continuous Covariate: Crp (mg/L)

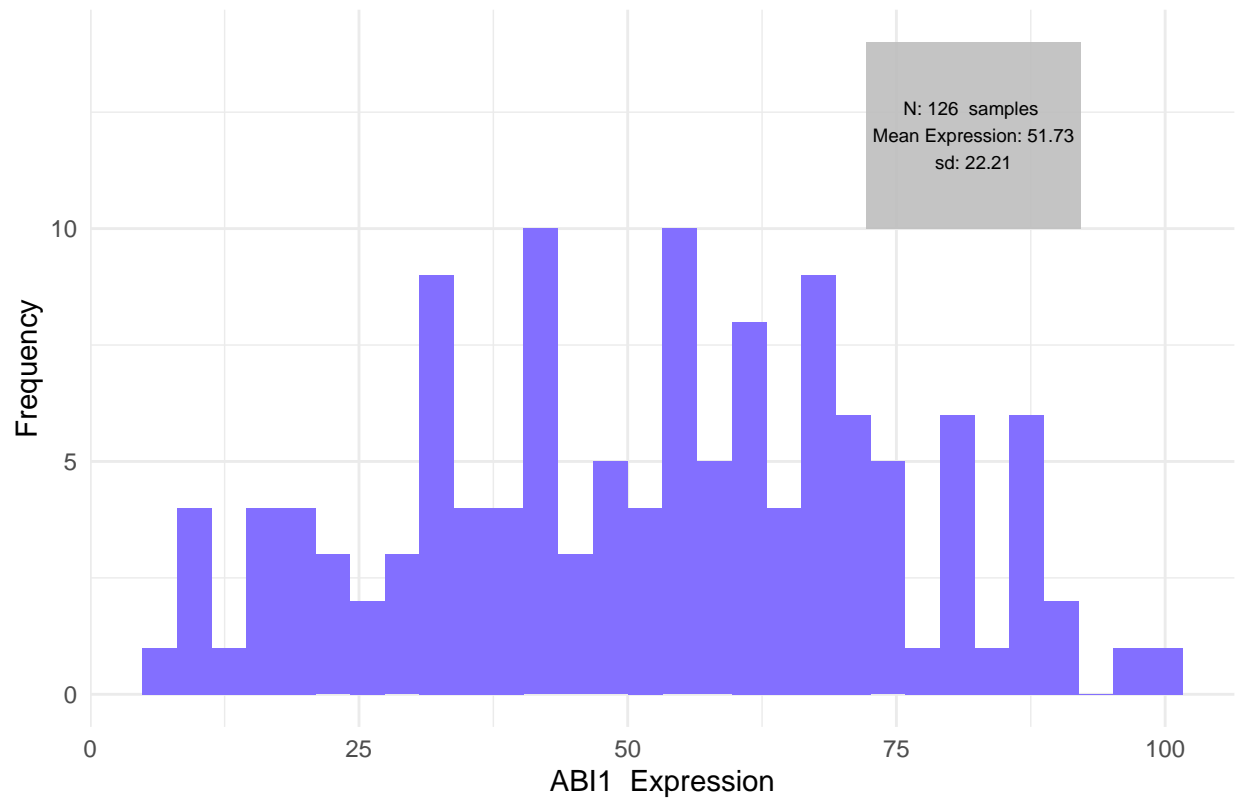
Categorical Covariates: Disease Status and Mechanical Ventilation

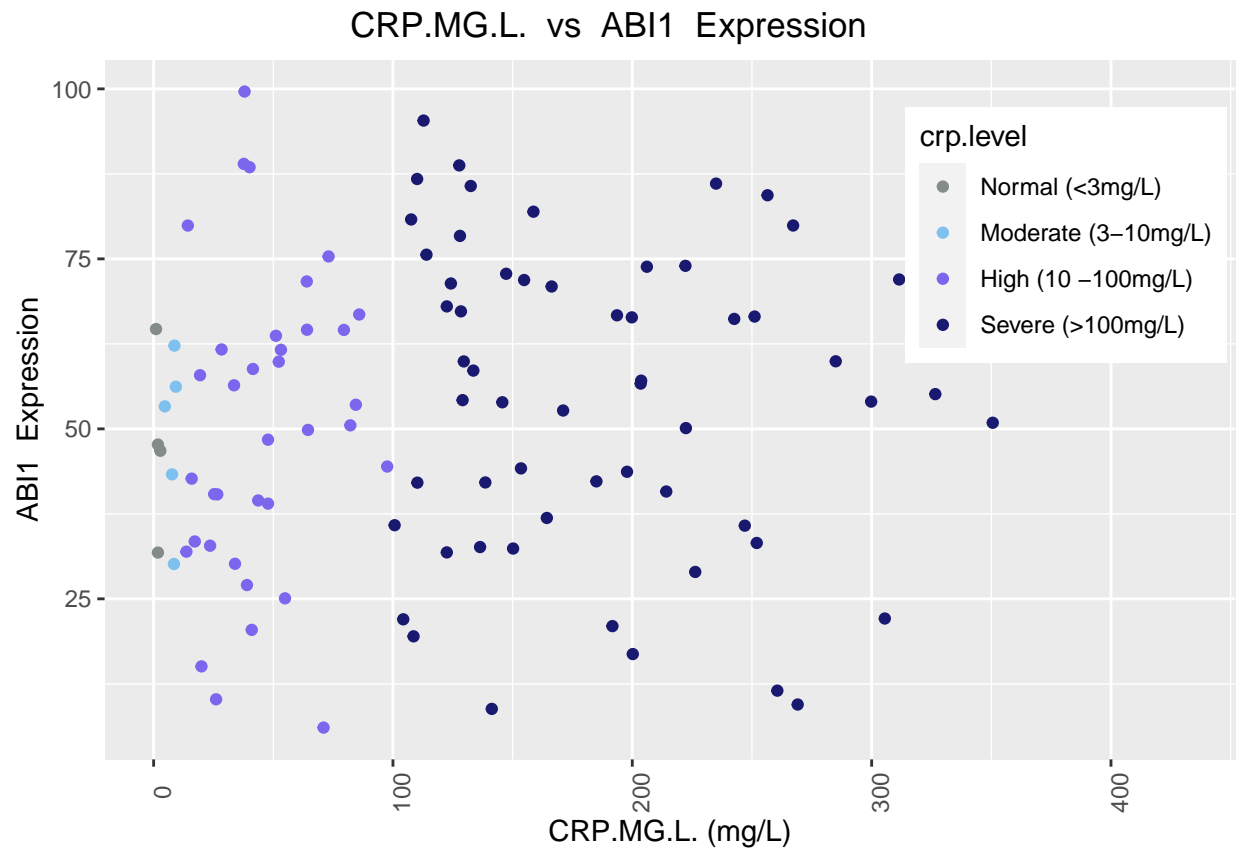
```

#a few warnings about deprecated methods, did not cause
#problems so decided to suppress for final output
suppressWarnings({ plots(all_data, c('ABI1', 'AAMP', 'ABHD17A'),
c('crp.mg.l.'),
c('disease_status', 'mechanical_ventilation')) } )

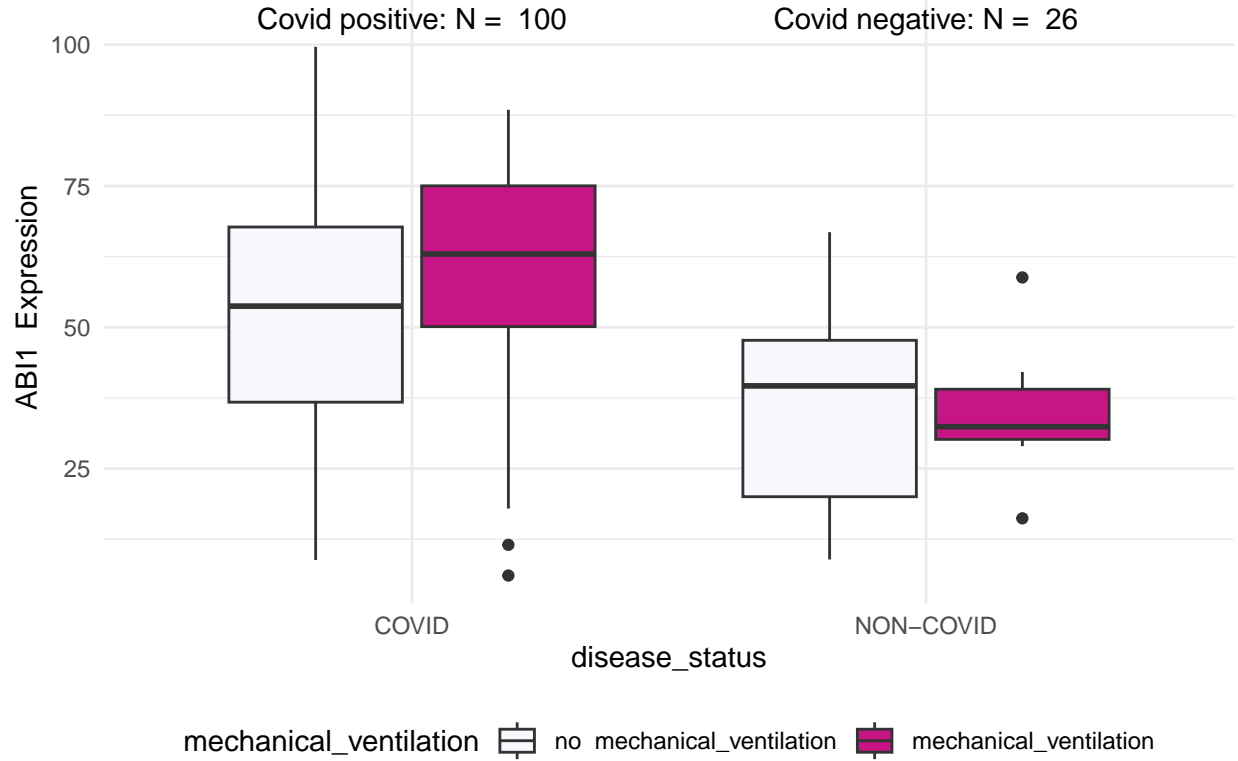
```

Distribution of ABI1 expression across samples





ABI1 Expression across disease_status and mechanical_ventilation



Distribution of AAMP expression across samples

