main

2024-07-20

```r
# importing necessary libraries

library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3

## Warning: package 'ggplot2' was built under R version 4.3.3

## Warning: package 'tibble' was built under R version 4.3.3

## Warning: package 'tidyr' was built under R version 4.3.3

## Warning: package 'readr' was built under R version 4.3.3

## Warning: package 'purrr' was built under R version 4.3.3

## Warning: package 'dplyr' was built under R version 4.3.3

## Warning: package 'stringr' was built under R version 4.3.3

## Warning: package 'forcats' was built under R version 4.3.3

## Warning: package 'lubridate' was built under R version 4.3.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- 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
```

```r
library(rlang)
```

```
## Warning: package 'rlang' was built under R version 4.3.3

##
## Attaching package: 'rlang'
##
## The following objects are masked from 'package:purrr':
##
##     %@%, flatten, flatten_chr, flatten_dbl, flatten_int, flatten_lgl,
##     flatten_raw, invoke, splice
```

```r
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 4.3.3
```

```
library(pheatmap)
```

```
## Warning: package 'pheatmap' was built under R version 4.3.3
```

```
# import the csv
series_matrix<- read.csv('./data/QBS103_GSE157103_series_matrix.csv')
genes <- read.csv('./data/QBS103_GSE157103_genes.csv')

# example code for finding the package version and citations

# packageVersion('tidyverse')
# packageVersion('rlang')
# packageVersion('pheatmap')
# citation('ggpubr')
```

```
# function to transpose/prepare the data from series_matrix and genes

prepare_dataframe <- function(series_matrix, genes){
  unique(series_matrix$age) # this one is covariate

  # let's merge these two data sets together

  # transpose the genes dataframe
  transposed_genes <- as.data.frame(t(genes))
  colnames(transposed_genes) <- genes$X
  transposed_genes <- transposed_genes[2:length(genes), ]


  for (i in colnames(transposed_genes)){
    series_matrix[[i]] <- as.numeric(transposed_genes[[i]]) # convert the transposed data to numeric
  }
  return(series_matrix)
}

dataframe <- prepare_dataframe(series_matrix, genes)
dataframe <- subset(dataframe, sex != " unknown")

# discrete
# sex, icu_status, disease_status

# continuous
# age, ferritin.ng.ml, charlson_score

# Select the chosen discrete and continious variables
selected_dataframe <- dataframe[c('sex', 'icu_status', 'disease_status', 'age', 'ferritin.ng.ml.', 'crp

# ensure that they are treated as numeric
selected_dataframe$age <- as.numeric(selected_dataframe$age)
```

```
## Warning: NAs introduced by coercion
```

```
selected_dataframe$ferritin.ng.ml. <- as.numeric(selected_dataframe$ferritin.ng.ml.)
```

```
## Warning: NAs introduced by coercion
```

```r
selected_dataframe$crp.mg.l. <- as.numeric(selected_dataframe$crp.mg.l.)
```

```
## Warning: NAs introduced by coercion
```

```r
table(selected_dataframe$sex)
```

```
##
##  female    male
##      51      74
```

```r
# find the divider for calculation of n%
divider <- as.numeric(sum(table(selected_dataframe$disease_status)))

# generate the summary table
summary_table <- selected_dataframe %>%
  dplyr::group_by(sex) %>%
  dplyr::summarise(`Age (mean) (years)` = mean(age, na.rm = TRUE),
                   `Age (sd) (years)` = sd(age, na.rm = TRUE),
                   `Ferritin (mean) (ng/ml)` = mean(ferritin.ng.ml., na.rm =  TRUE),
                   `Ferritin (sd) (ng/ml)` = sd(ferritin.ng.ml., na.rm = TRUE),
                   `CRP (mean) (mg/l)` = mean(crp.mg.l., na.rm = TRUE),
                   `CRP (sd) (mg/l)` = sd(crp.mg.l., na.rm = TRUE),
                   `Disease Status (n%)` = table(disease_status)/divider * 100,
                   `Sex (n%)` = table(sex)/divider * 100,
                   `ICU Status (n%)` = table(icu_status)/divider * 100
                   )
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

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

```r
# In order to calculate summary statistics of n% for male and female seperately, the following code was

a <- subset(dataframe, sex == "Male")
table(a$disease_status)/sum(table(a$disease_status))
```

```
## numeric(0)
```

```r
table(a$icu_status)/sum(table(a$icu_status))
```

```
## numeric(0)
```

```r
b <- subset(dataframe, sex == "Female")
table(b$disease_status)/sum(table(b$disease_status))
```

```
## numeric(0)
```

```r
table(b$icu_status)/sum(table(b$icu_status))
```

```
## numeric(0)
```

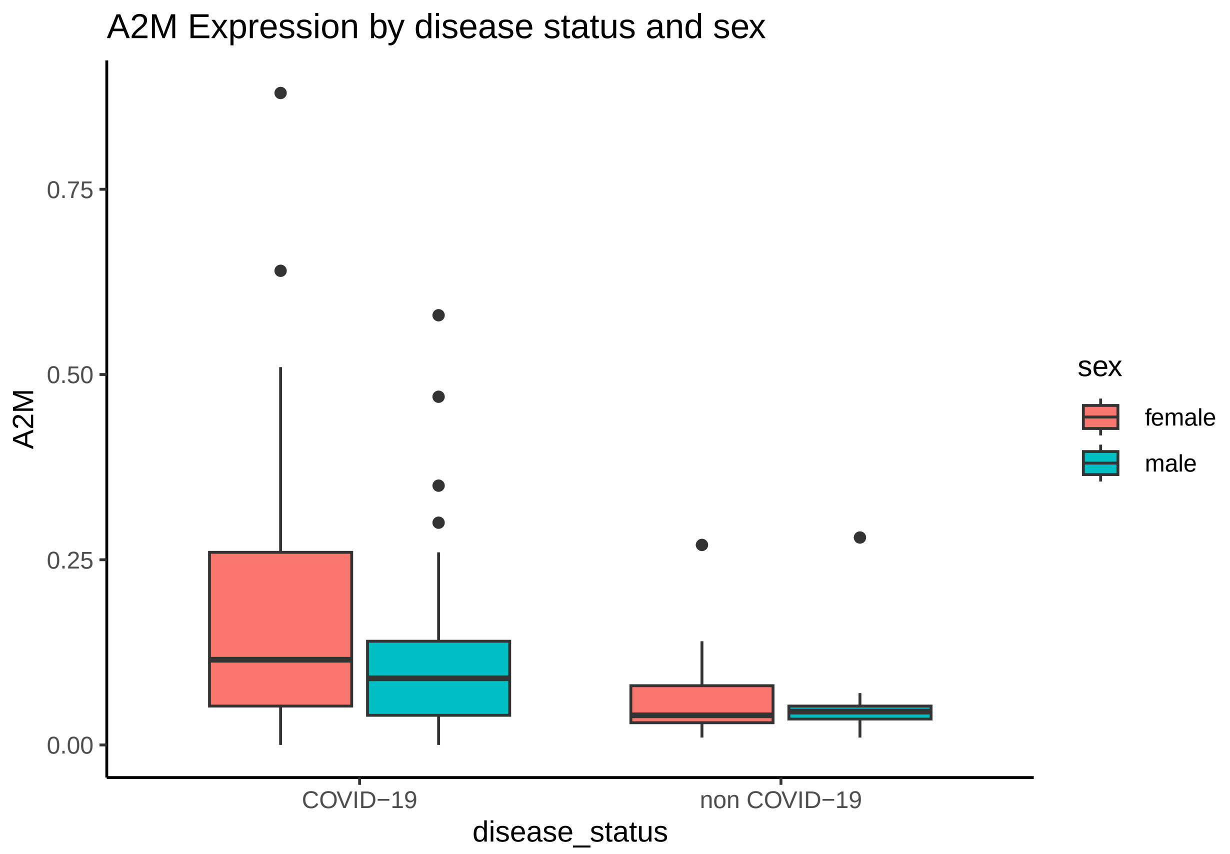
```r
# https://stackoverflow.com/questions/55132771/standard-eval-with-ggplot2-without-aes-string/55133909#5
# source used form !!sym (to pass a string as a symbol in an aes environment)


generate_plots <- function(dataframe, list_of_genes, continuous_variable, categorical_covariates)
  {
  # This the continuous variables
  # plots scatter

  for (gene in list_of_genes){
    print(gene)
    print(continuous_variable)
    scatter <- ggplot(dataframe, aes(x = !!sym(continuous_variable)
                                     , y = !!sym(gene), color = sex)) +
    geom_point() +
    xlab('Age (years)') +
    ylab(paste(gene, ' Expression')) +
    scale_x_discrete(breaks = seq(0, 100, by = 10)) +
    ggtitle(paste( "Expression of ", gene, " vs Age")) +
    labs(color = "Sex") +
    theme_classic()


    # this is for the two categorical variables
    # plot the box plot
    box <- ggplot(dataframe, aes(y = !!sym(gene),
                                 x = !!sym(categorical_covariates[1]),
                                 fill = !!sym(categorical_covariates[2]))) +
      geom_boxplot(names(c('COVID', 'NON-COVID'))) +
      labs(x = categorical_covariates[1], y = gene) +
      scale_x_discrete(labels = c("disease state: COVID-19" = "COVID-19",
                                  "disease state: non-COVID-19" = "non COVID-19")) +
      theme(axis.text.x = element_text(angle = 0, hjust = 0.5, size = 8))  +
      ggtitle(paste(gene, "Expression by disease status", "and", categorical_covariates[2])) + theme_cla

    # plot the histogram
    print(gene)
    histo <- ggplot(dataframe, aes(x = !!sym(gene))) +
    geom_histogram() +
    theme_classic() +
    ggtitle(paste( "Expression of ", gene))

    # save the plots
    ggsave("boxplot.png", plot = box)
    ggsave("scatterplot.png", plot = scatter)
    ggsave("histogram.png", plot = histo)

    return (list(box, histo, scatter))
  }

}

plots <- generate_plots(dataframe, c('A2M'), 'age', c('disease_status', 'sex') )
```

```
## [1] "A2M"
## [1] "age"
## [1] "A2M"
```

```
## Saving 6.5 x 4.5 in image
## Saving 6.5 x 4.5 in image
## Saving 6.5 x 4.5 in image
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
plots[1]
```

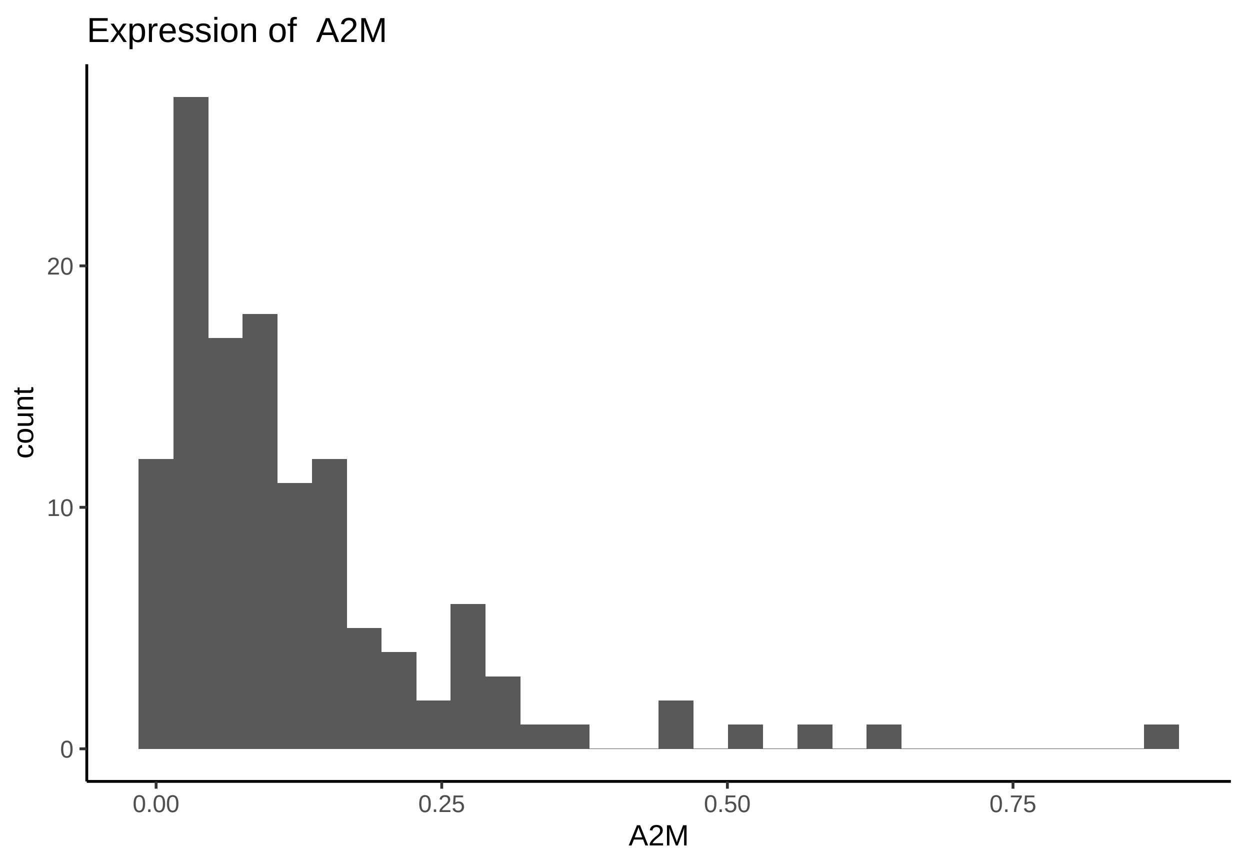
```
## [[1]]
```



```
plots[2]
```

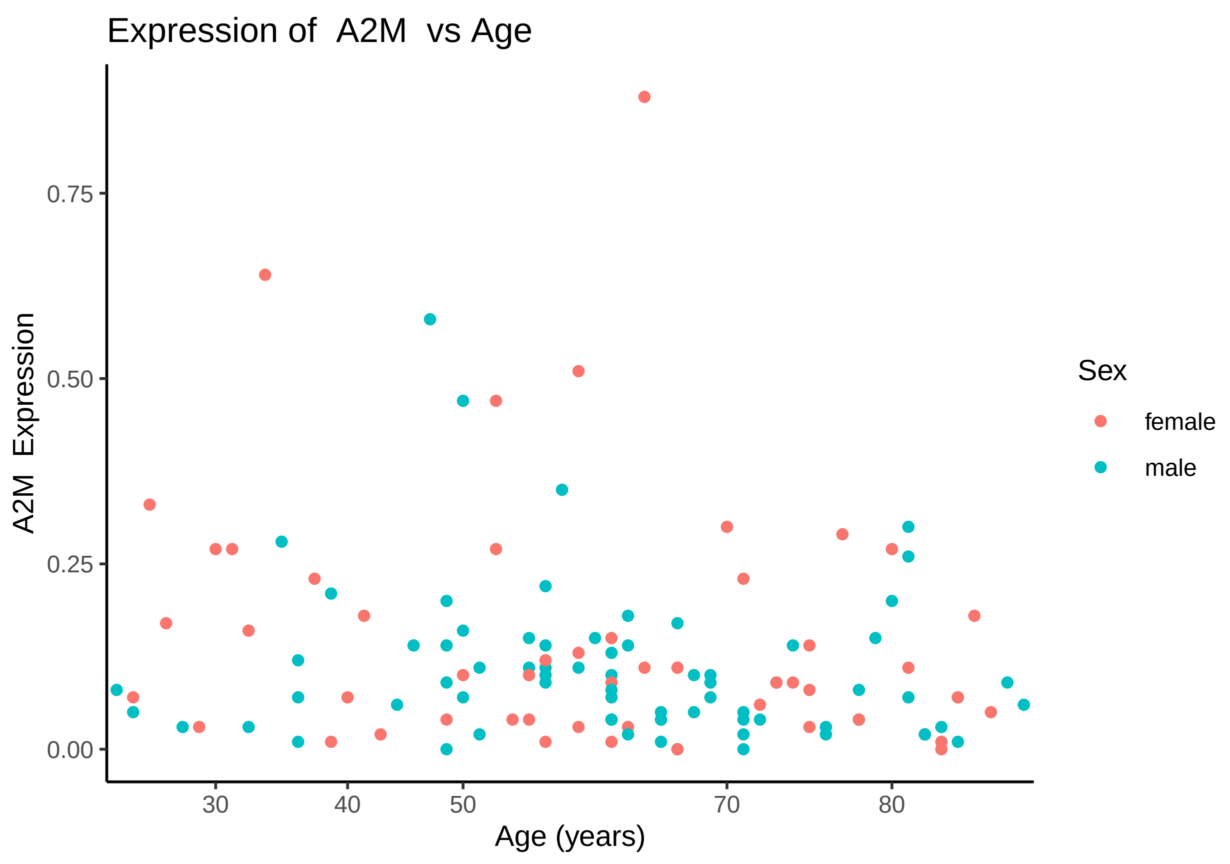
```
## [[1]]
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
plots[3]
```

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

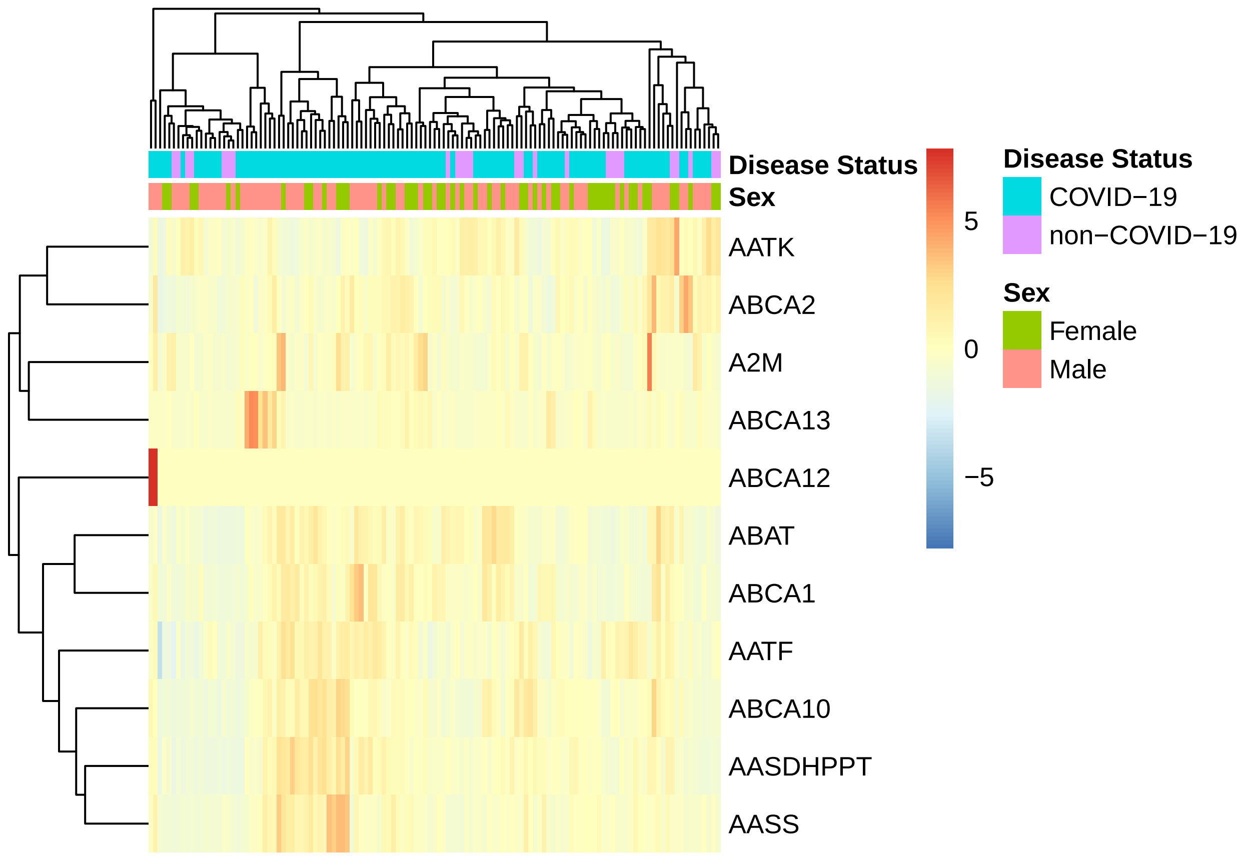


```
# select 10 genes
genes_vector <- c("A2M", "AASDHPPT", "AASS", "AATF", "AATK", "ABAT", "ABCA1", "ABCA10", "ABCA12", "ABCA

# create a dataframe with those genes
gene_df <- select(dataframe, genes_vector)
```

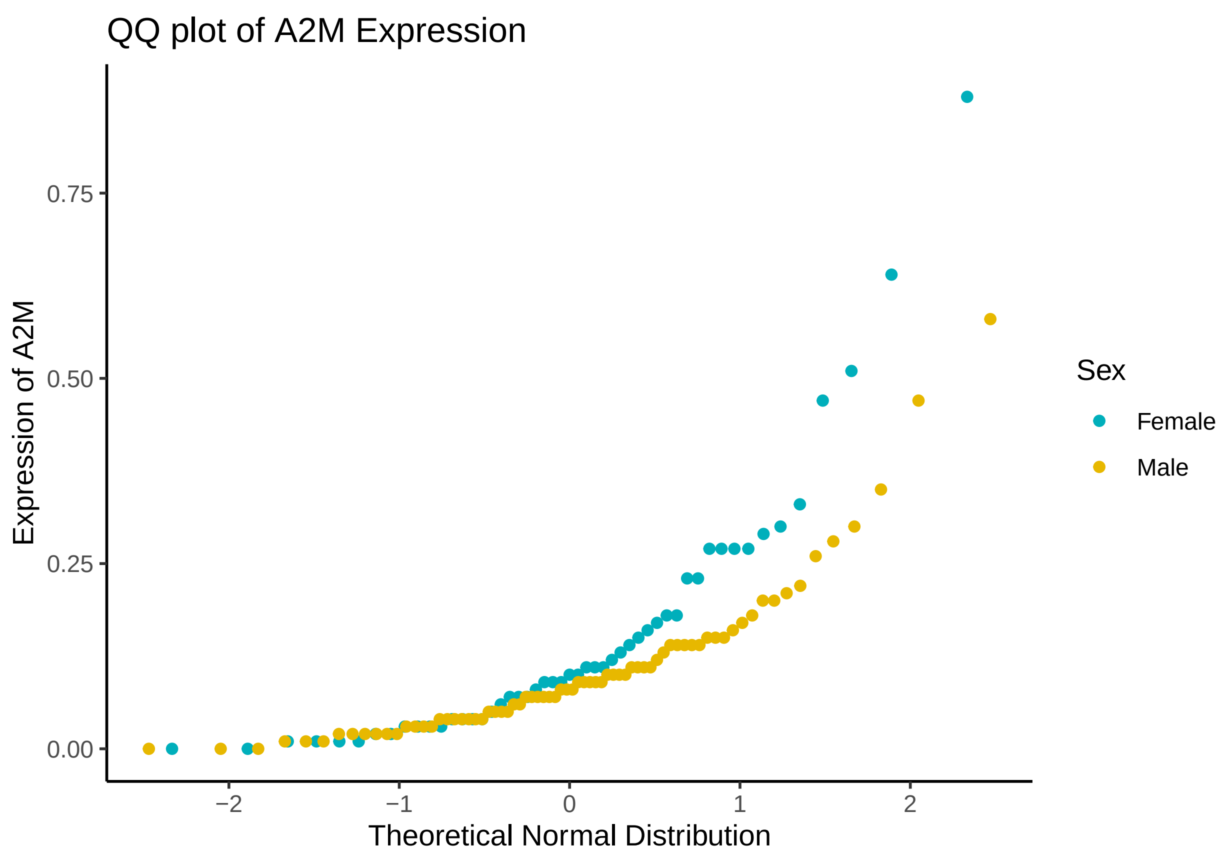
```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(genes_vector)
##
##   # Now:
##   data %>% select(all_of(genes_vector))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# generate the annotation dataframe
annotation <- select(dataframe, c("sex", "disease_status"))  # Include 'sex' and 'disease_status

# rename for clarity
annotation$disease_status <- ifelse(annotation$disease_status=="disease state: COVID-19", 'COVID-19', '
annotation$sex <- ifelse(annotation$sex==" female", 'Female', 'Male')
colnames(annotation) <- c('Sex', 'Disease Status')
```

```r
# transpose gene dataframe for heatmap generation
transposed_ndf <- t(gene_df)

# generate the heatmap using euclidean clustering
pheatmap(transposed_ndf,
         clustering_distance_rows = "euclidean",
         clustering_distance_cols = "euclidean",
         clustering_method = "complete",
         scale = "row",
         annotation_col = annotation,
         show_colnames = FALSE)
```



```r
# Generate a QQ plot
dataframe$sex <- ifelse(dataframe$sex==" female", 'Female', 'Male')

ggplot(dataframe, aes(sample = A2M)) +
  stat_qq(aes(color = sex)) +
  scale_color_manual(values = c("#00AFBB", "#E7B800"))+
  labs(y = "Expression of A2M") +
  theme_classic() +
  ggtitle('QQ plot of A2M Expression') +
  xlab('Theoretical Normal Distribution') +
  labs(color = "Sex")
```



```r
ggsave(('qq_plot.png'))
```

```
## Saving 6.5 x 4.5 in image
```