Submission2

2025-07-18

Build a function to create the plots you made for Presentation 1, incorporating any feedback you received on your submission. Your functions should take the following input: (1) the name of the data frame, (2) a list of 1 or more gene names, (3) 1 continuous covariate, and (4) two categorical covariates (10 pts)

read data

```
library(tidyverse)
```

```
## -- 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   1.5.0
## v ggplot2   3.5.2     v tibble    3.3.0
## v lubridate 1.9.2     v tidyr     1.3.1
## v purrr     1.0.1
## -- 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 errors
```

```
#read data
gene_data <- read.csv("/Users/yuxinyi/Dartmouth/Data Science/QBS103_GSE157103_genes.csv", row.names = 1)
meta_data <- read.csv("/Users/yuxinyi/Dartmouth/Data Science/QBS103_GSE157103_series_matrix-1.csv", row

# Extract necessary columns and preprocess the metadata
meta_data_cleaned <- meta_data %>%
  rownames_to_column(var = "SampleID") %>%
  select(SampleID, age, sex, disease_status) %>%
  mutate(
    age = as.character(age),
    sex = as.character(sex),
    disease_status = as.character(disease_status)
  )

#age column: set non-numeric entries to NA, then convert to numeric
meta_data_cleaned$age[!grepl("^\\d+$", meta_data_cleaned$age)] <- NA
meta_data_cleaned$age <- as.numeric(meta_data_cleaned$age)

#Replace missing values in age with the median age
median_age <- median(meta_data_cleaned$age, na.rm = TRUE)
meta_data_cleaned$age[is.na(meta_data_cleaned$age)] <- median_age

#Replace missing or "unknown" values in sex with the mode (most frequent value)
mode_sex <- names(sort(table(meta_data_cleaned$sex[meta_data_cleaned$sex != "unknown" & !is.na(meta_data
meta_data_cleaned$sex[meta_data_cleaned$sex == "unknown" | is.na(meta_data_cleaned$sex)] <- mode_sex
```

```r
#Replace missing or "unknown" values in disease_status with the mode
mode_disease <- names(sort(table(meta_data_cleaned$disease_status[meta_data_cleaned$disease_status != "
meta_data_cleaned$disease_status[meta_data_cleaned$disease_status == "unknown" | is.na(meta_data_cleane

#Convert cleaned columns to appropriate types: numeric and factor
variable_df <- meta_data_cleaned %>%
  mutate(
    age = as.numeric(age),
    sex = as.factor(sex),
    disease_status = as.factor(disease_status)
  )
```

create function

```r
plot_gene_analysis <- function(data_frame, gene_name, continuous_var,
                               cat_var1, cat_var2) {
  #Set a standard name for expression column
  expr_col <- "expression"
  #Extract gene expression values
  gene_vector <- gene_data[gene_name, ]
  #Convert gene expression into a tidy data frame
  gene_df <- data.frame(
    SampleID = names(gene_vector),
    expression = as.numeric(gene_vector),
    stringsAsFactors = FALSE
  )
  #Merge gene expression with metadata using SampleID
  combined_df <- gene_df %>%
    left_join(data_frame, by = "SampleID") %>%
    filter(
      !is.na(age),
      !is.na(sex),
      !is.na(disease_status),
      sex != "unknown",
      disease_status != "unknown"
    )
  #Define a clean ggplot theme to apply to all plots
  newBlankTheme <- theme(
    panel.border = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line = element_line(colour = "black", linewidth = rel(1)),
    plot.background = element_rect(fill = "white"),
    panel.background = element_blank(),
    legend.key = element_rect(fill = 'white'),
    legend.position = 'top'
  )
  #Histogram
  p1 <- ggplot(combined_df, aes(x = expression)) +
    geom_histogram(binwidth = 0.1, fill = "skyblue", color = "black") +
    labs(title = paste("Distribution of", gene_name, "Expression"),
         x = paste(gene_name, "Expression"),
         y = "Number of Samples") +
    newBlankTheme
    #Scatterplot
  p2 <- ggplot(combined_df, aes_string(x = continuous_var, y = "expression")) +
    geom_point() +
    labs(
      title = paste(gene_name, "Expression vs.", continuous_var),
      x = continuous_var,
      y = paste(gene_name, "Expression (Log2 intensity)")
    ) +
    newBlankTheme
  #Boxplot
  p3 <- ggplot(combined_df, aes_string(x = cat_var1, y = "expression",
                                        fill = cat_var2)) +
    geom_boxplot() +
    labs(
      title = paste(gene_name, "Expression by", cat_var1, "and", cat_var2),
      x = cat_var1,
      y = paste(gene_name, "Expression (Log2 intensity)"),
      fill = cat_var2
    ) +
    newBlankTheme

  #Show all three plots
  print(p1)
  print(p2)
  print(p3)
}
```

Select variables from metadata

```
variable_df <- meta_data %>%
  rownames_to_column(var = "SampleID") %>% #convert row names to column'SampleID'
  select(SampleID, age, sex, disease_status) %>%
  mutate(
    age = as.numeric(as.character(age)), #convert age to numeric
    sex = as.factor(sex), # convert sex to factor
    disease_status = as.factor(disease_status) #convert disease_status to factor
  )
```

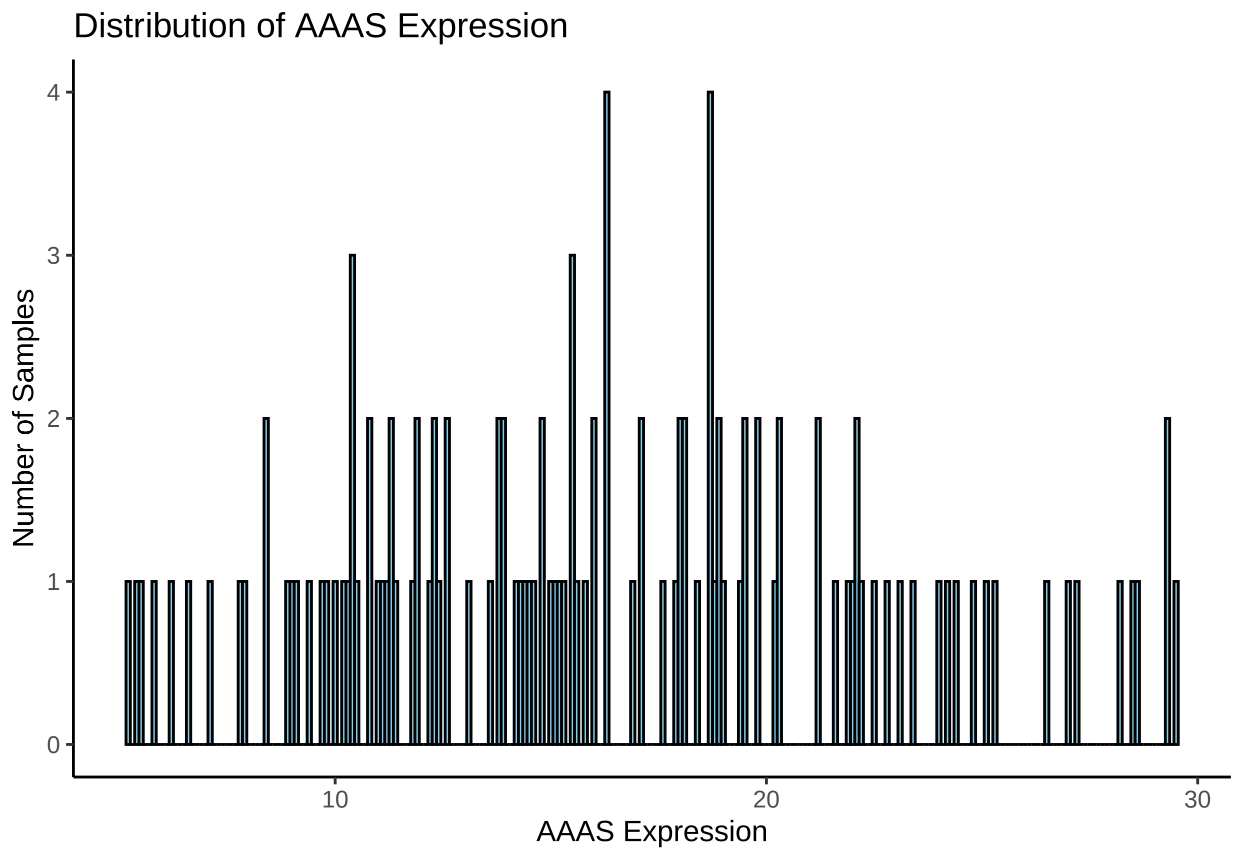
```
## Warning: There was 1 warning in ‘mutate()‘.
## i In argument: ‘age = as.numeric(as.character(age))‘.
## Caused by warning:
## ! NAs introduced by coercion
```

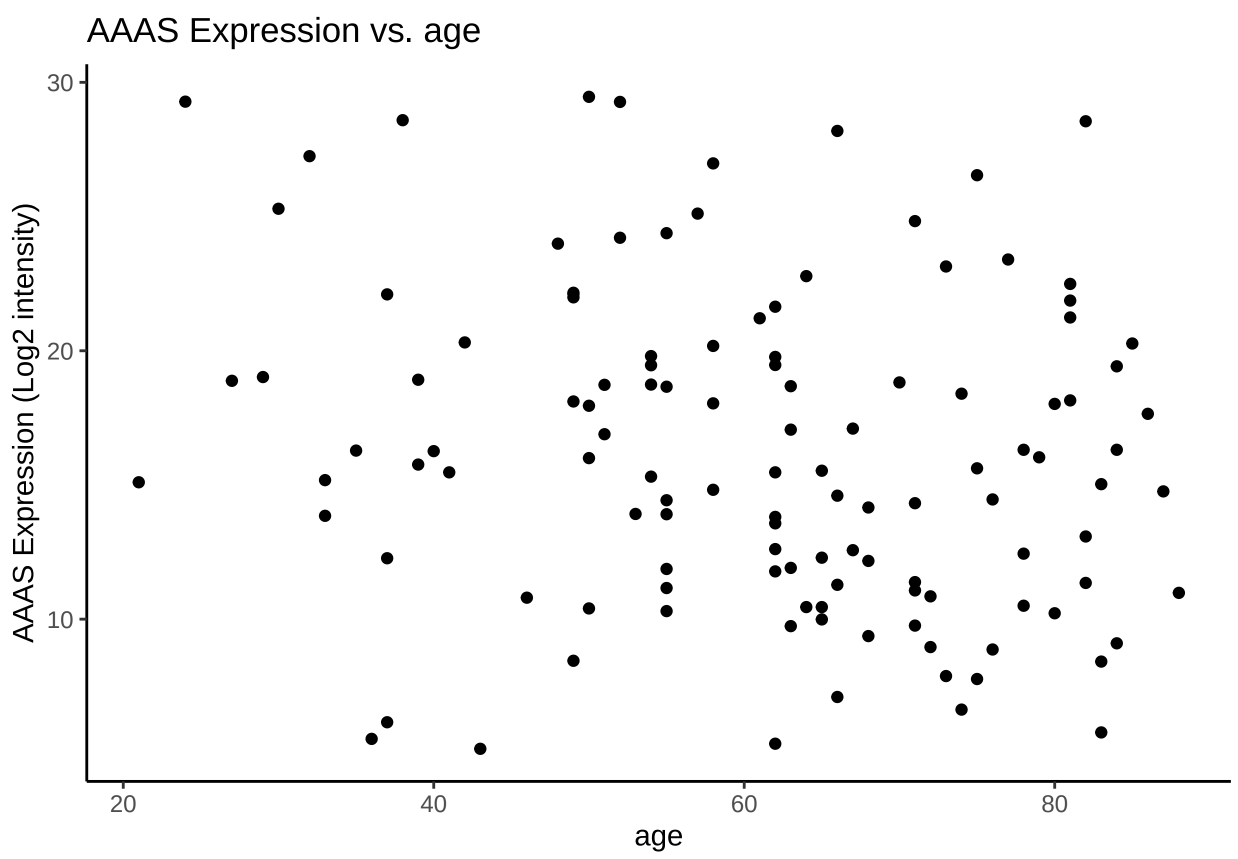
```
#head(variable_df)
```

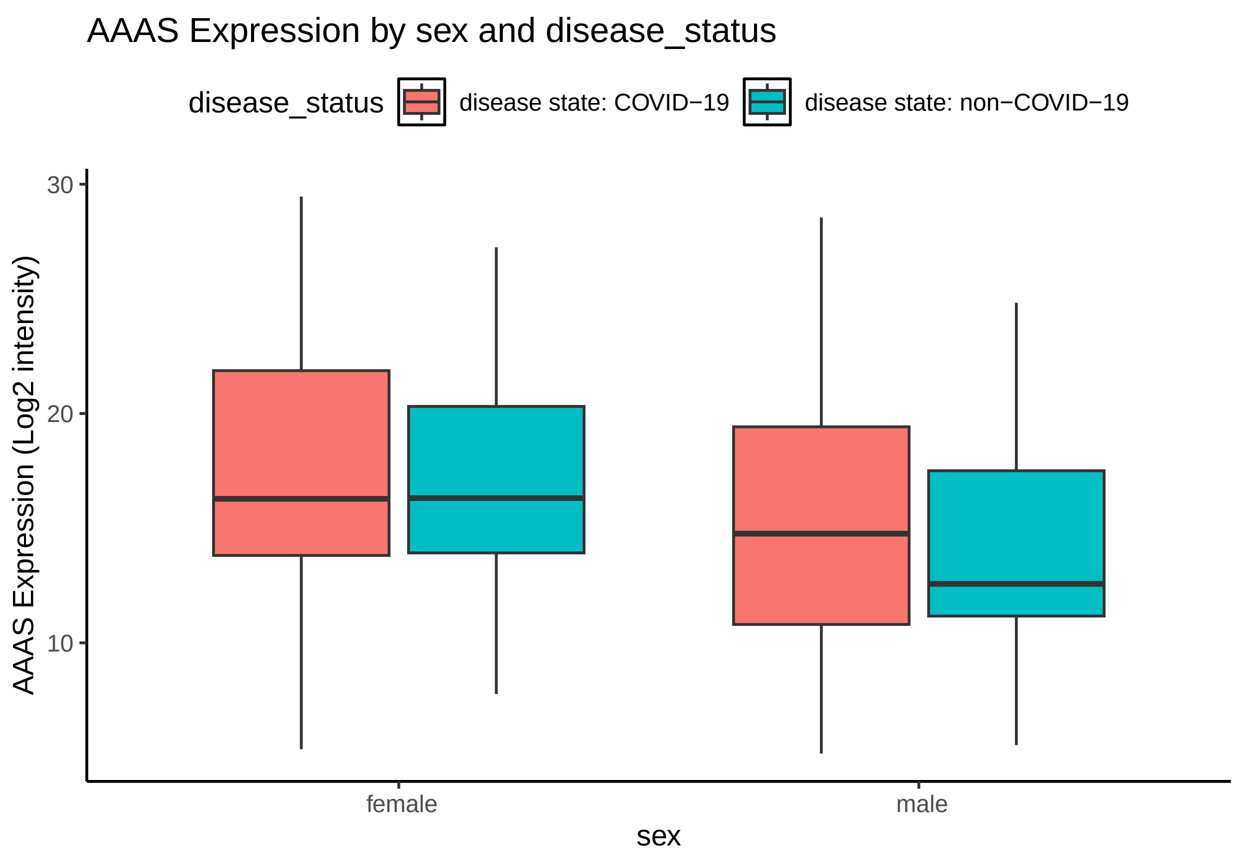
show plot

```
#Use the function to generate plots for the gene AAAS
plot_gene_analysis(
  data_frame = variable_df,
  gene_name = "AAAS",
  continuous_var = "age",
  cat_var1 = "sex",
  cat_var2 = "disease_status"
)
```

```
## 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.
```

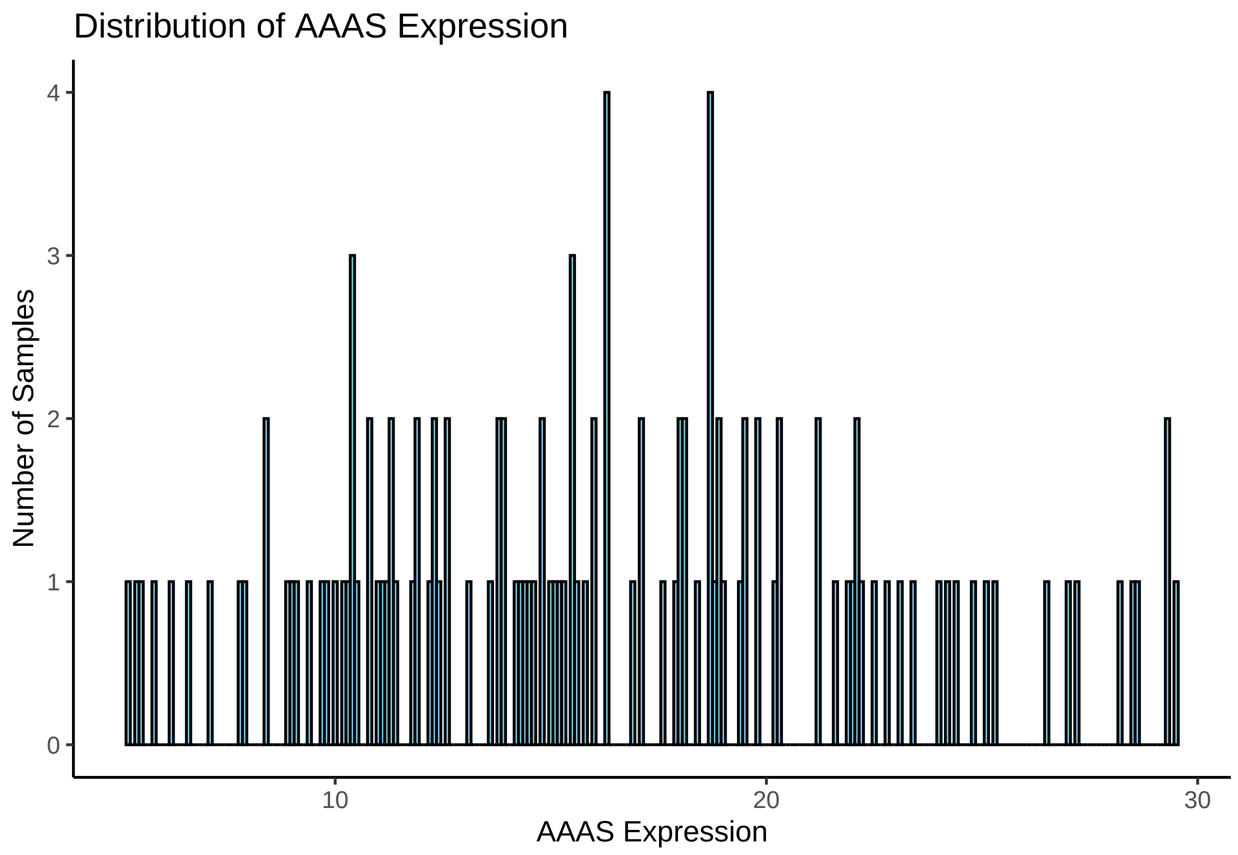


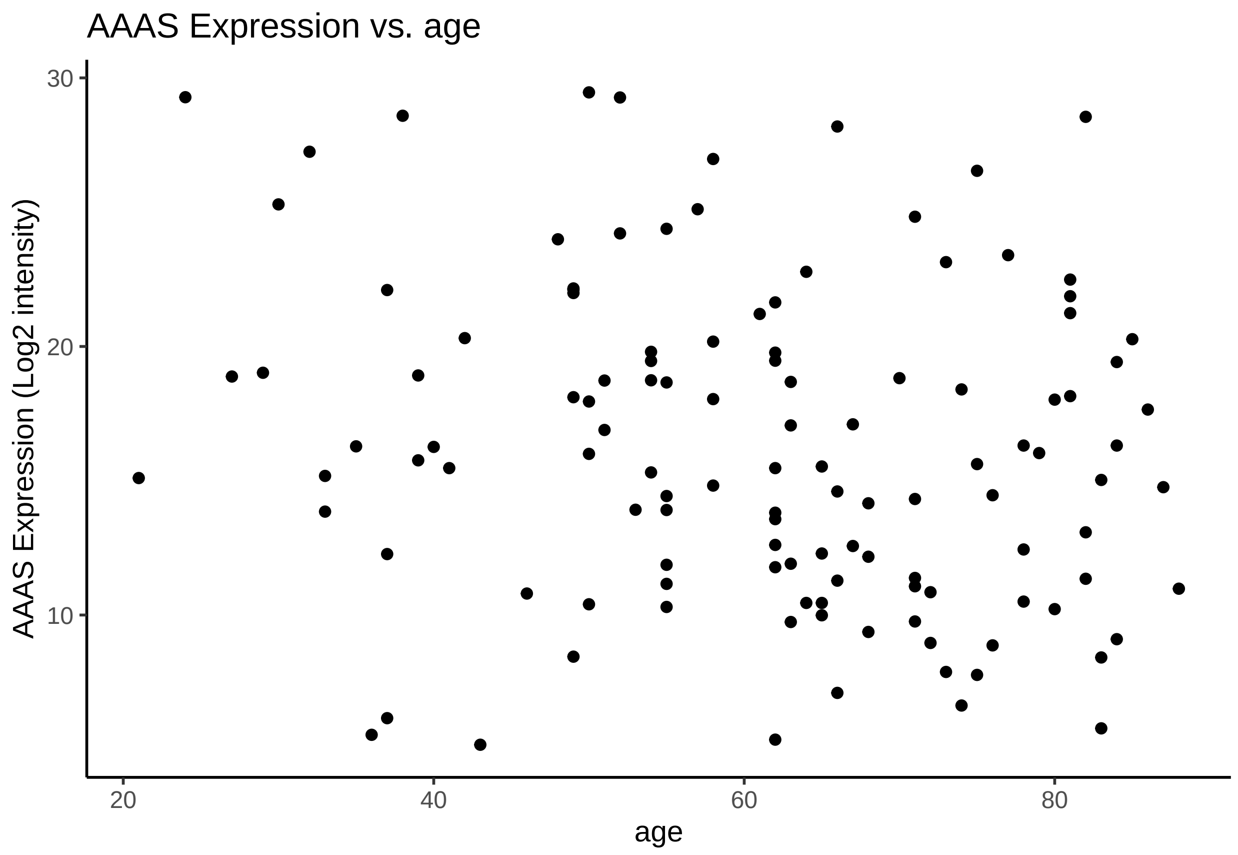

AAAS Expression vs. age
AAAS Expression (Log2 intensity)
30
20
10
20
40
60
80
age

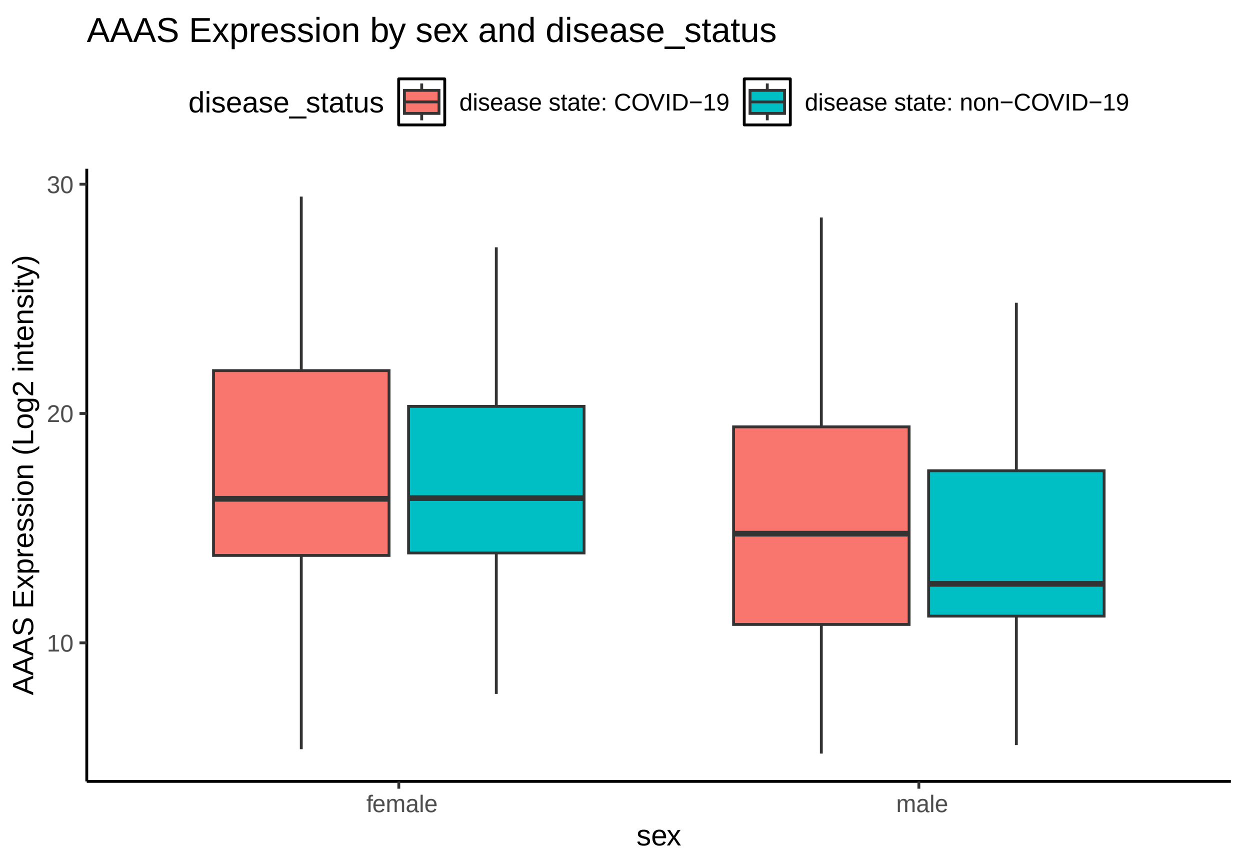


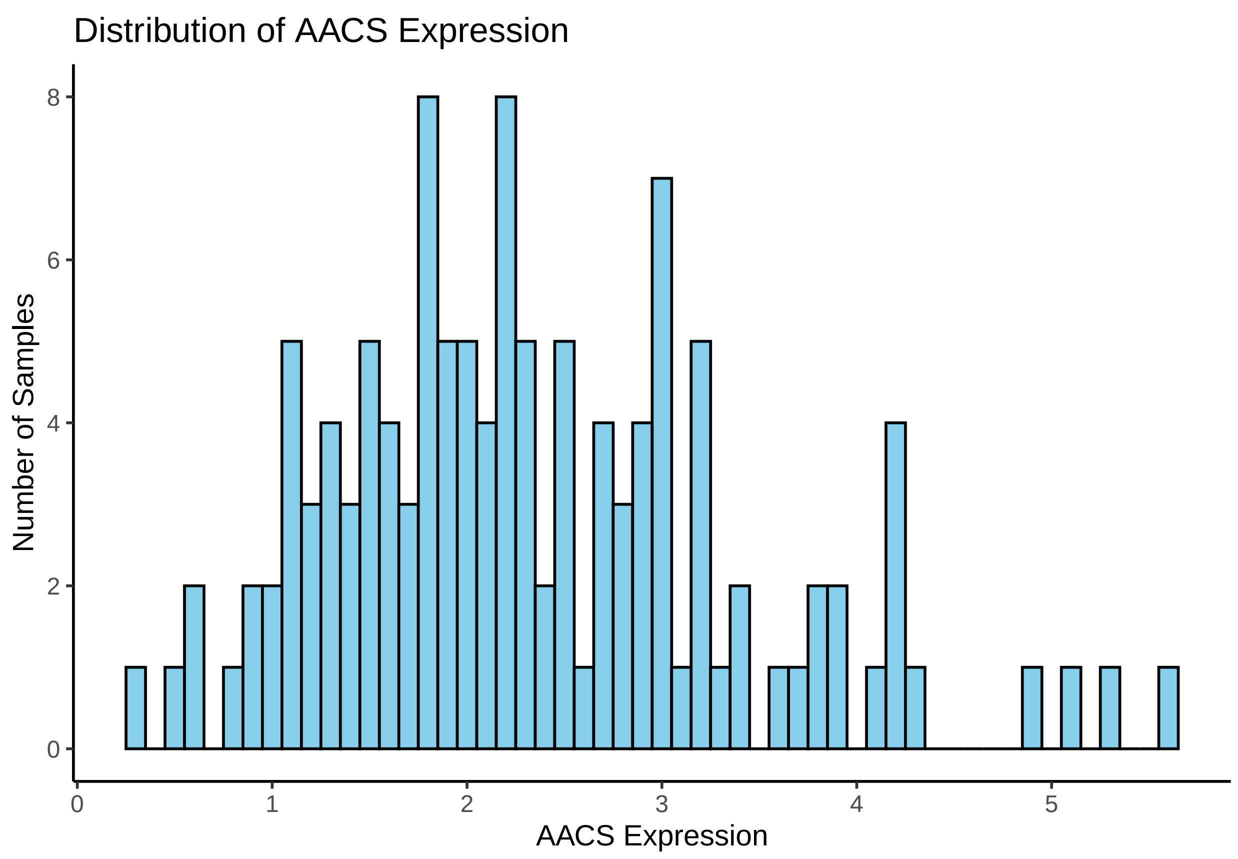
Select 2 additional genes (for a total of 3 genes) to look at and implement a loop to generate your figures using the function you created (10 pts)

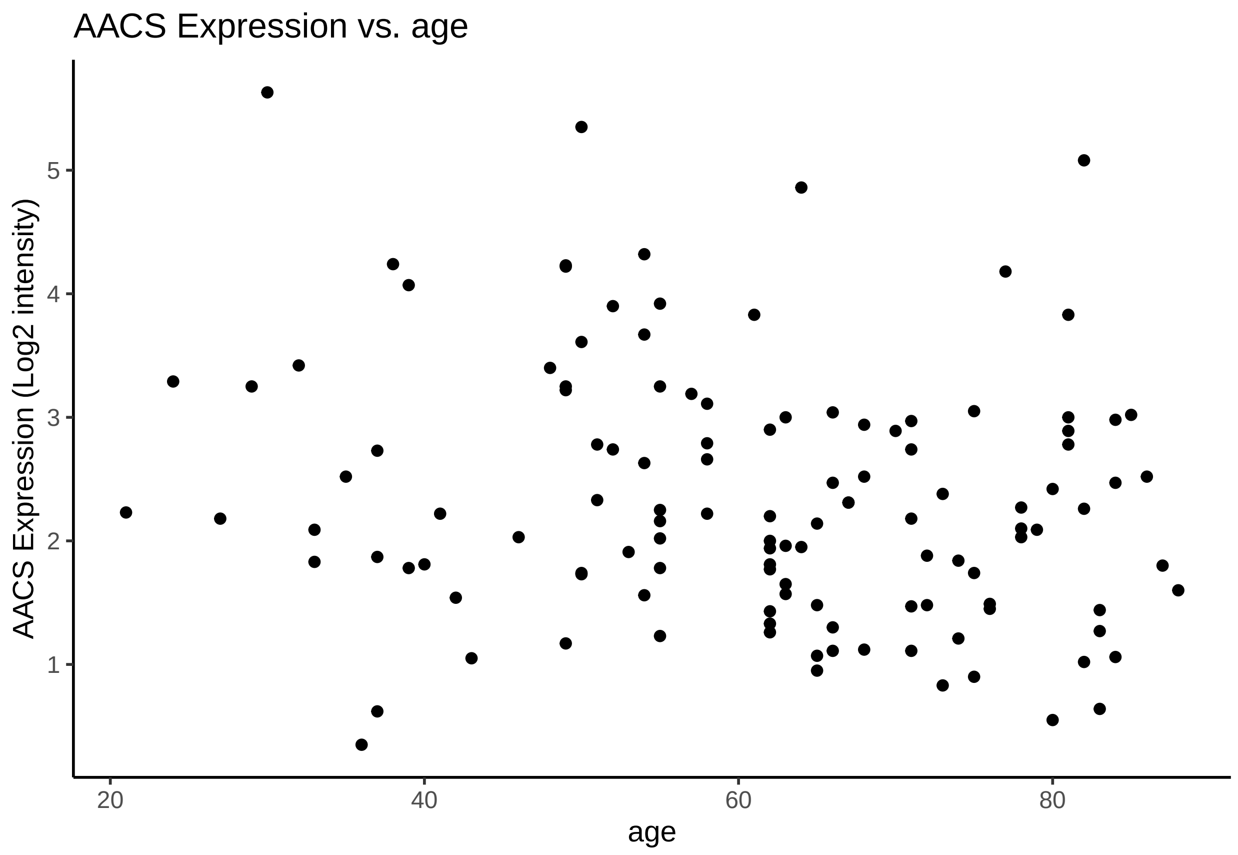
```r
#Define a list of 3 genes
gene_list <- c("AAAS", "AACS", "AATF")
#Loop over each gene in the list and call the plot function
for (gene in gene_list) {
  plot_gene_analysis(
    data_frame = variable_df,
    gene_name = gene,
    continuous_var = "age",
    cat_var1 = "sex",
    cat_var2 = "disease_status"
  )
}
```

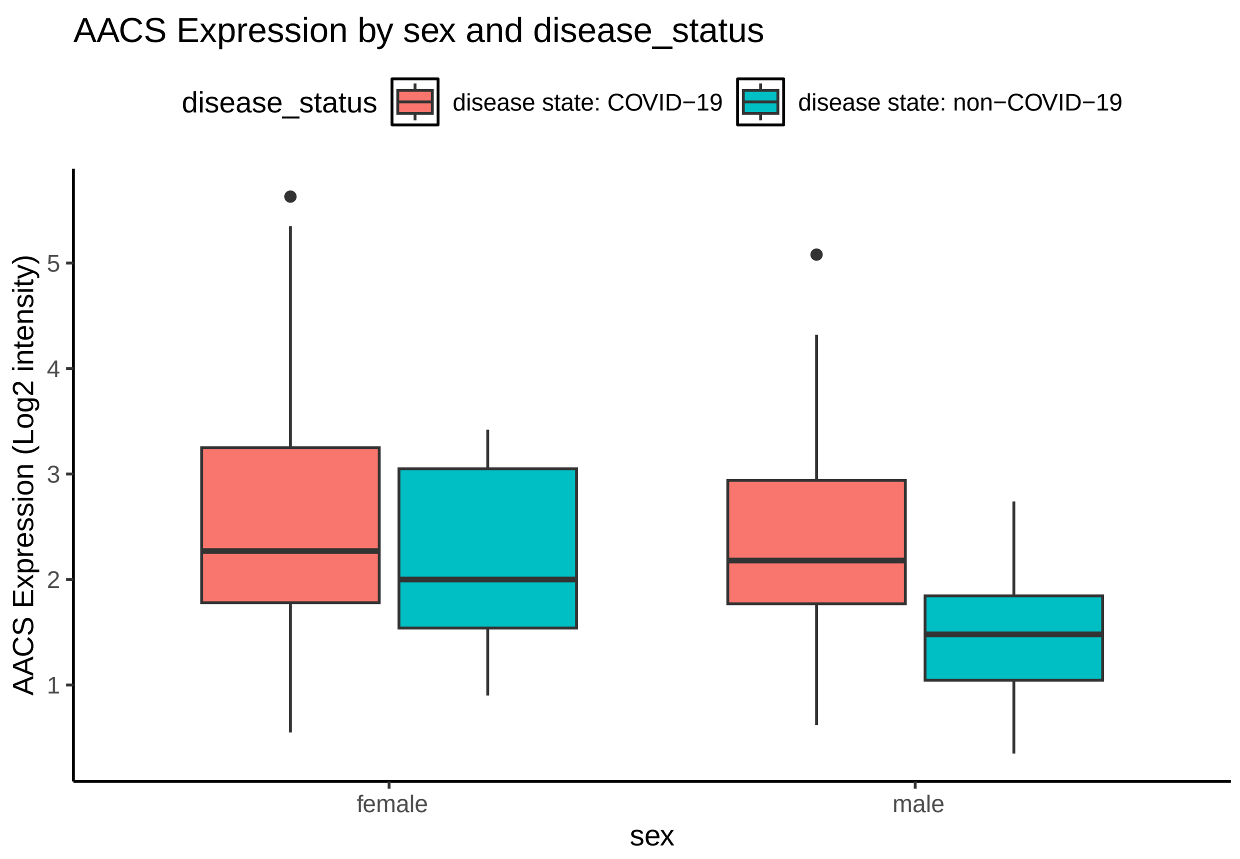

Distribution of AAAS Expression
Number of Samples
0
1
2
3
4
10
20
30
AAAS Expression

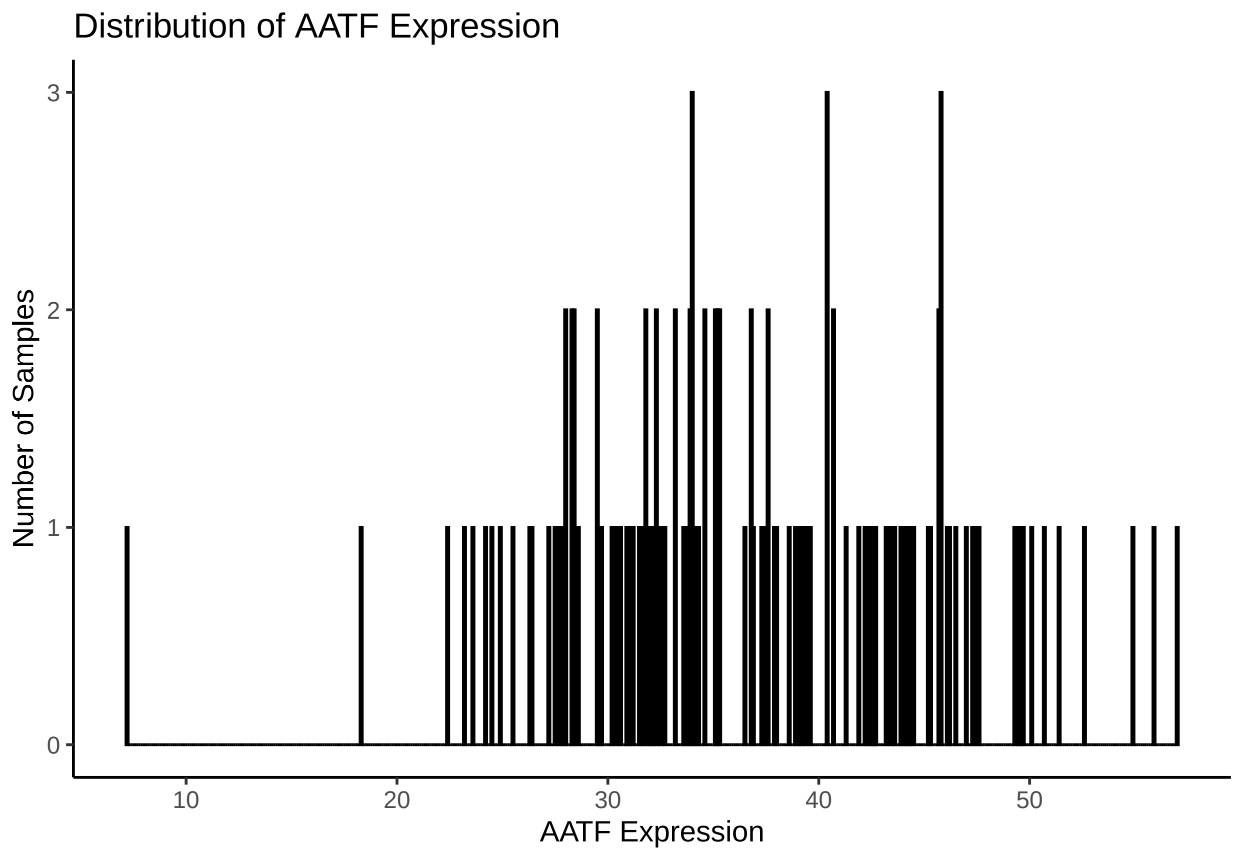

AAAS Expression vs. age
AAAS Expression (Log2 intensity)
30
20
10
20
40
60
80
age

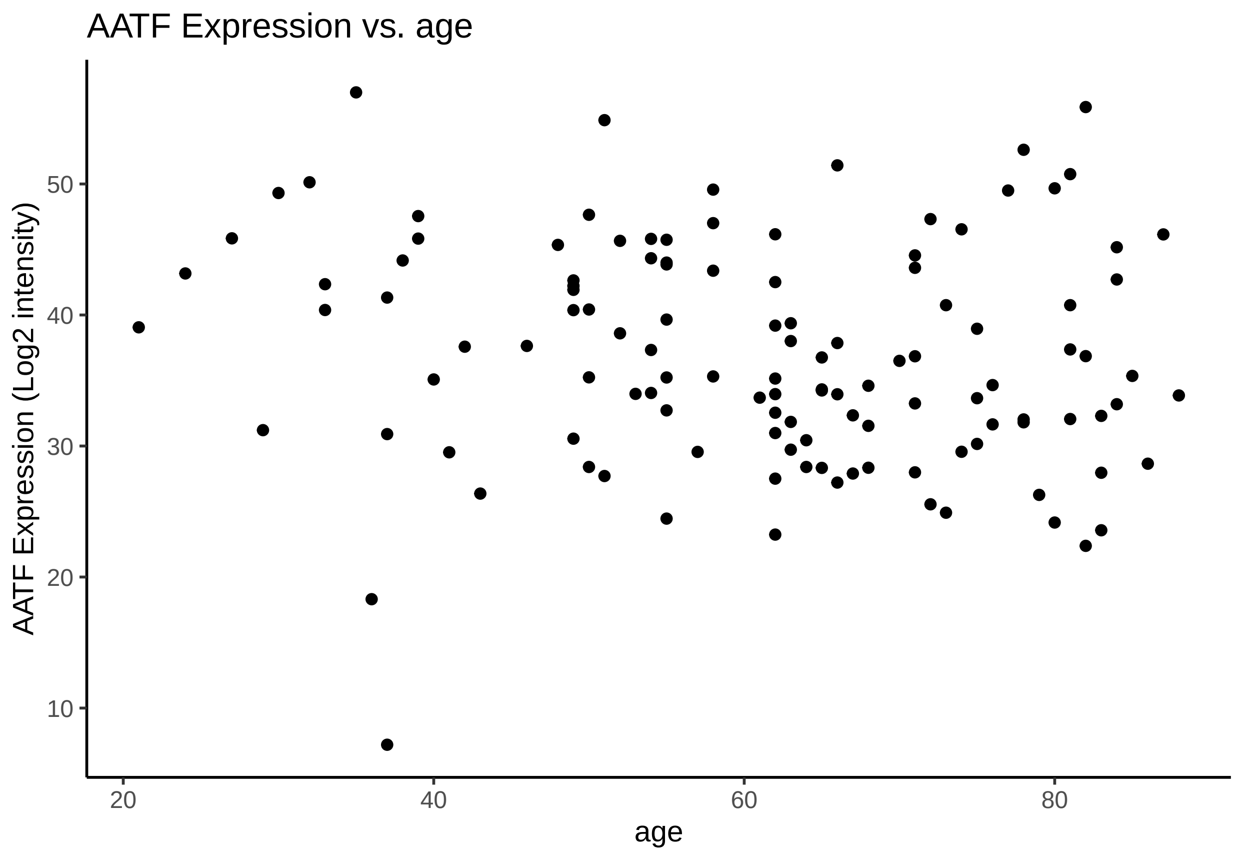

AAAS Expression by sex and disease_status
disease_status
disease state: COVID−19
disease state: non−COVID−19
AAAS Expression (Log2 intensity)
30
20
10
female
male
sex

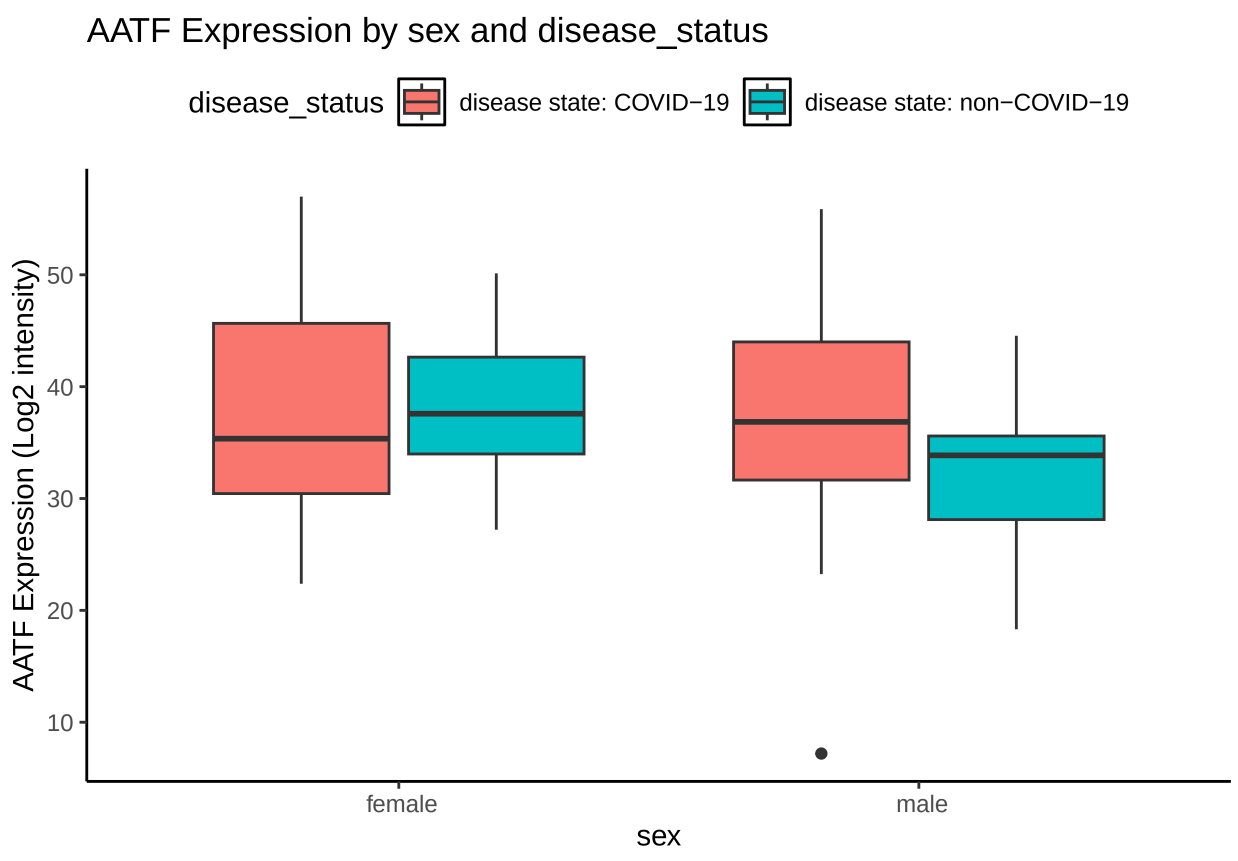

Distribution of AACS Expression
Number of Samples
AACS Expression
0
2
4
6
8
0
1
2
3
4
5


AACS Expression vs. age
AACS Expression (Log2 intensity)
1
2
3
4
5
20
40
60
80
age


AACS Expression by sex and disease_status
disease_status
disease state: COVID−19
disease state: non−COVID−19
AACS Expression (Log2 intensity)
1
2
3
4
5
female
male
sex


Distribution of AATF Expression
Number of Samples
AATF Expression
0
1
2
3
10
20
30
40
50


AATF Expression vs. age
AATF Expression (Log2 intensity)
50
40
30
20
10
20
40
60
80
age


AATF Expression by sex and disease_status
disease_status
disease state: COVID−19
disease state: non−COVID−19
AATF Expression (Log2 intensity)
50
40
30
20
10
female
male
sex