Final Submission

2024-08-27

```
library(knitr)
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   1.5.1
## 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 errors
```

```
library(dplyr)
library(table1)
```

```
##
## Attaching package: 'table1'
##
## The following objects are masked from 'package:base':
##
##     units, units<-
```

```
library(pheatmap)
```

1.Generate a table formatted in LaTeX of summary statistics for all the covariates you looked at and 2 additional continuous (3 total) and 1 additional categorical variable (3 total). (5 pts) 1)Stratifying by one of your categorical variables 2)Tables should report n (%) for categorical variables 3)Tables should report mean (sd) or median [IQR] for continuous variables

```
my_table_css <- "
.table1 { color: black !important; }
"

#Set the working directory to the path of given dataset.
setwd("E:/55555Dartmouth_QBS/QBS_103_Foundation_of_DS_R_/Project")

#Using read.csv to read in the gene expression data and metadata
genes_expression<-read.csv(file="QBS103_GSE157103_genes.csv",header=T,stringsAsFactors=F,row.names = 1)
genes_expression<-t(genes_expression)
metadata<-read.csv(file="QBS103_GSE157103_series_matrix.csv",header=T,stringsAsFactors=F)
```

```r
linked_data<- merge(genes_expression,metadata, by.x = "row.names", by.y = "participant_id", all = TRUE)
linked_data1<-na.omit(linked_data)
linked_data_filter <- linked_data1 %>%
  filter(!is.na(sex)&!is.na(icu_status)&!is.na(mechanical_ventilation)&sex!='unknown')

#change the gene expression of the linked data frame into numeric type
linked_data_filter$ddimer.mg.l_feu.<-as.numeric(linked_data_filter$ddimer.mg.l_feu.)
```

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

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

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

```r
linked_data_filter$lactate.mmol.l.<-as.numeric(linked_data_filter$lactate.mmol.l.)
```

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

```r
linked_data_filter$sex <- factor(linked_data_filter$sex)
linked_data_filter$icu_status <- factor(linked_data_filter$icu_status)
linked_data_filter$mechanical_ventilation <- factor(linked_data_filter$mechanical_ventilation)

#filter the data to exclude the NA value
linked_data_filter <- linked_data_filter  %>%
  filter(!is.na(ddimer.mg.l_feu.))%>%
  filter(!is.na(crp.mg.l.))%>%
  filter(!is.na(lactate.mmol.l.))
```

```r
#set the labels
label(linked_data_filter$icu_status) <- "ICU Status"
label(linked_data_filter$mechanical_ventilation) <- "Mechanical Ventilation"
label(linked_data_filter$ddimer.mg.l_feu. ) <- "DDimer(mg/L_feu)"
label(linked_data_filter$crp.mg.l.) <- "CRP(mg/L)"
label(linked_data_filter$lactate.mmol.l.) <-"Lactate(mmol/L)"

IQR_format <- c("Median [IQR]" = "MEDIAN [IQR]")

#generate the summary table
tb1<-table1(~ icu_status + mechanical_ventilation +
       ddimer.mg.l_feu. + crp.mg.l. + lactate.mmol.l. | sex,
       data=linked_data_filter,
       overall=c(left="Total"),
       render.continuous = IQR_format,
       css = my_table_css
)


# Convert the table to LaTeX format
latex_table <- kable(tb1, format = "latex")

# Save the LaTeX table to a .tex file
```

```
writeLines(latex_table, "output_table.tex")

tb1
```

```
## Get nicer `table1` LaTeX output by simply installing the `kableExtra` package
```

	Total	female	male
	(N=66)	(N=19)	(N=47)
ICU Status			
no	21 (31.8%)	7 (36.8%)	14 (29.8%)
yes	45 (68.2%)	12 (63.2%)	33 (70.2%)
Mechanical Ventilation			
no	31 (47.0%)	12 (63.2%)	19 (40.4%)
yes	35 (53.0%)	7 (36.8%)	28 (59.6%)
DDimer(mg/L_feu)			
Median [IQR]	2.33 [11.3]	1.87 [4.37]	3.59 [13.3]
CRP(mg/L)			
Median [IQR]	125 [162]	51.1 [117]	133 [161]
Lactate(mmol/L)			
Median [IQR]	1.20 [0.618]	1.17 [0.600]	1.22 [0.610]

2.Generate final histogram, scatter plot, and boxplot from submission 1 (i.e. only for your first gene of interest) incorporating all feedback from your presentations (5 pts)

```
makePlot<-function(dataSet,geneName,Cont,Cat1,Cat2){
  metadata<-read.csv(file="QBS103_GSE157103_series_matrix.csv",header=T,stringsAsFactors=F)

  #initialize an empty list to store the plots
  plots<-list()

  for (gene in geneName){

    #using 'which' to select the chosen gene and identify it
    gene_selected<-dataSet[which(dataSet[,1]==gene),]

    #use pipe and merge to convert the gene expression to required format and link two dataframes
    gene_selected<-gene_selected %>%
      gather(key=participant_id,value=expression)
    linked_data<-merge(metadata,gene_selected)

    #change the gene expression of the linked data frame into numeric type
    linked_data$expression<-as.numeric(linked_data$expression)
    linked_data$ddimer.mg.l_feu.<-as.numeric(linked_data$ddimer.mg.l_feu.)

    #generate the histogram plot
    p1<-ggplot(linked_data,aes(expression))+
      geom_histogram(fill='lightblue',color='#694F8E')+
      scale_x_continuous(breaks = seq(0,100,by=1)) +
      labs(title=paste0("Gene Expression of ",gene),x=gene,y='Count')+
      theme_classic()+
      theme(
```

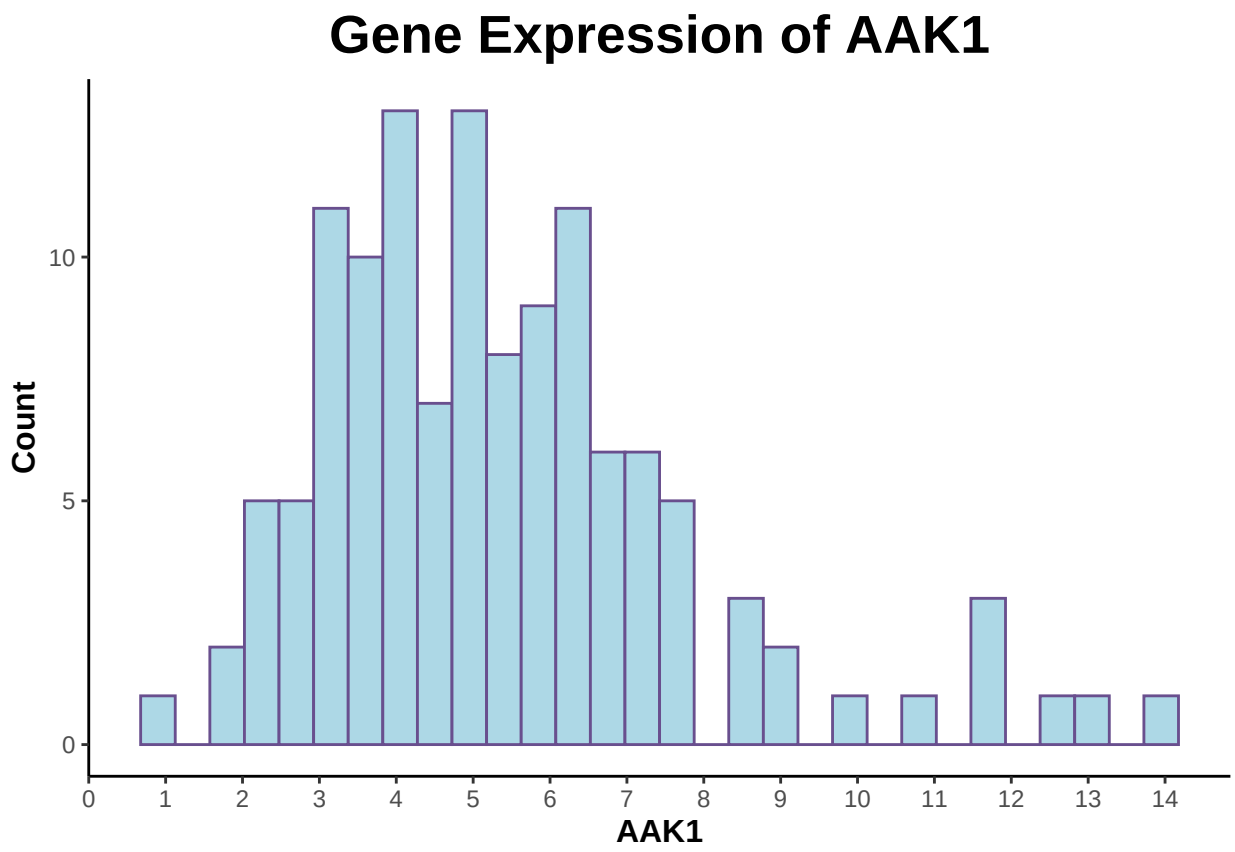
```
        plot.title=element_text(hjust=0.5,size=20,face='bold'),
        axis.title.x = element_text(size = 12, face = "bold"),
        axis.title.y = element_text(size = 12, face = "bold")
      )

    # Set x-axis to display specified ticks
    breaks <- seq(0, 200, by = 5)  # Display every 5 ticks

    #generate the scatterplot and do customization
    p2<-ggplot(linked_data,aes(y=expression,x=ddimer.mg.l_feu.))+
      geom_point(color='darkblue')+
      scale_x_continuous(breaks = breaks) +
      labs(title=paste0('Scatterplot for ',gene,' and ddimer'),y=gene,x='ddimer(mg/L_feu)')+
      theme_classic()+
      theme(
        plot.title = element_text(hjust = 0.5, size = 20, face = "bold"),
        axis.title.x = element_text(size = 12, face = "bold"),
        axis.title.y = element_text(size = 12, face = "bold"),
        #turn a angle to avoid overlapping
        axis.text.x = element_text(size = 8, angle = 45, hjust = 1),
        axis.text.y = element_text(size = 10)
        )

    #generate the boxplot
    p3<-ggplot(linked_data,aes(x=sex,y=expression,fill=icu_status))+
      geom_boxplot(outlier.shape = NA)+
      labs(title=paste0("Boxplot of ",gene," Expression separated by ",Cat1," and ",Cat2),
       x=Cat1,
       y=gene)+
      stat_summary(fun = mean, geom = "point", shape = 20, size = 3, color = "#A94438",
                   position = position_dodge(width = 0.75))+
      scale_fill_manual(values=c(" yes"='#EED3D9'," no"='#B5C0D0'))+
      theme_classic()+
      theme(
        plot.title = element_text(hjust = 0.5, size = 15, face = "bold"),
        axis.title.x = element_text(size = 12, face = "bold"),
        axis.title.y = element_text(size = 12, face = "bold"),
        axis.text.x = element_text(size = 10),
        axis.text.y = element_text(size = 10),
        legend.title = element_text(size = 12),
        legend.text = element_text(size = 10),
        legend.position ="top"
      )

    #add the three plots into the list according to its input gene name
    plots[[gene]] <- list(histogram = p1, scatter = p2, boxplot = p3)
    }

    return(plots)
}

setwd("E:/55555Dartmouth_QBS/QBS_103_Foundation_of_DS_R_/Project")
genes_expression<-read.csv(file="QBS103_GSE157103_genes.csv",header=T,stringsAsFactors=F)
```

```r
# Specify genes of interest
genes <- c('AAK1')

# Generate plots
plots <- makePlot(dataSet=genes_expression, geneName=genes, Cont='ddimer.mg.l_feu.',
                  Cat1='sex',
                  Cat2='icu_status')
```

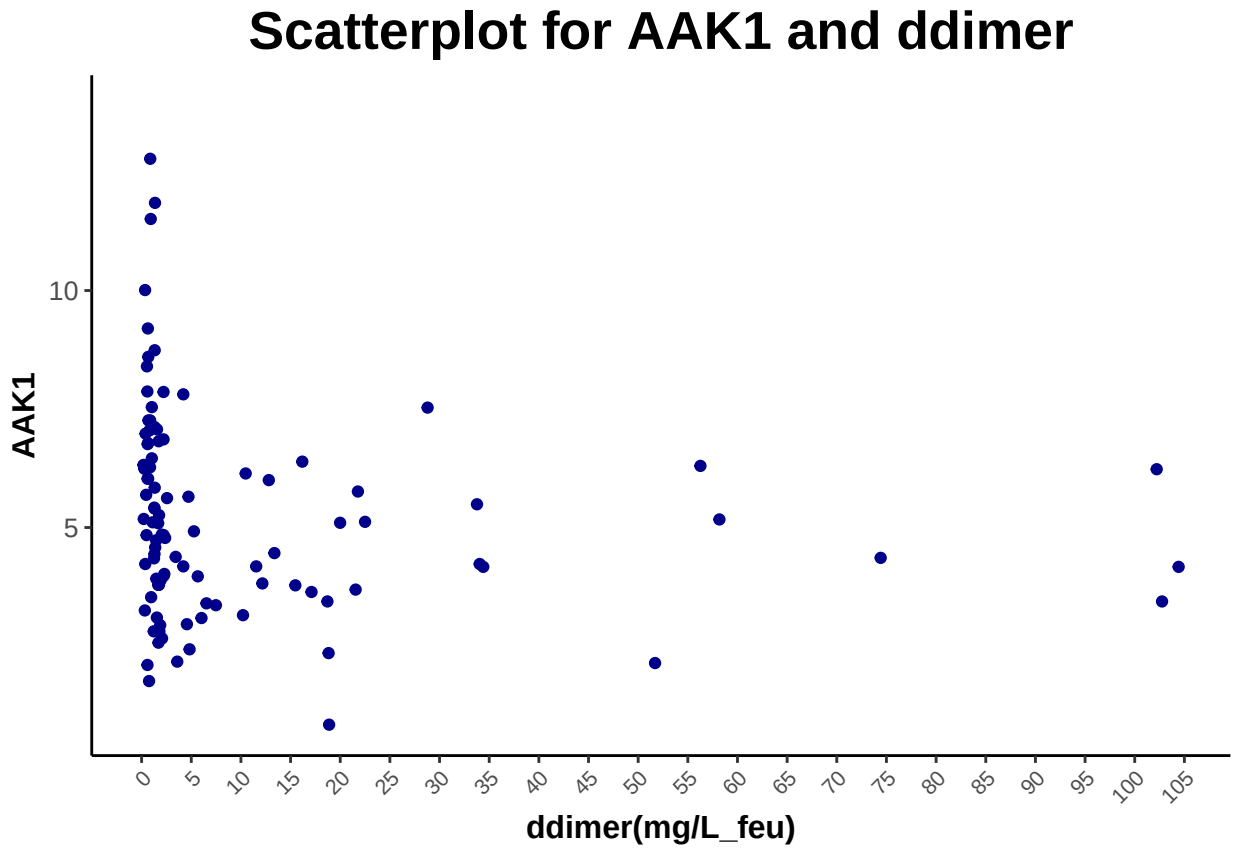
```
## Warning in makePlot(dataSet = genes_expression, geneName = genes, Cont =
## "ddimer.mg.l_feu.", : NAs introduced by coercion
```

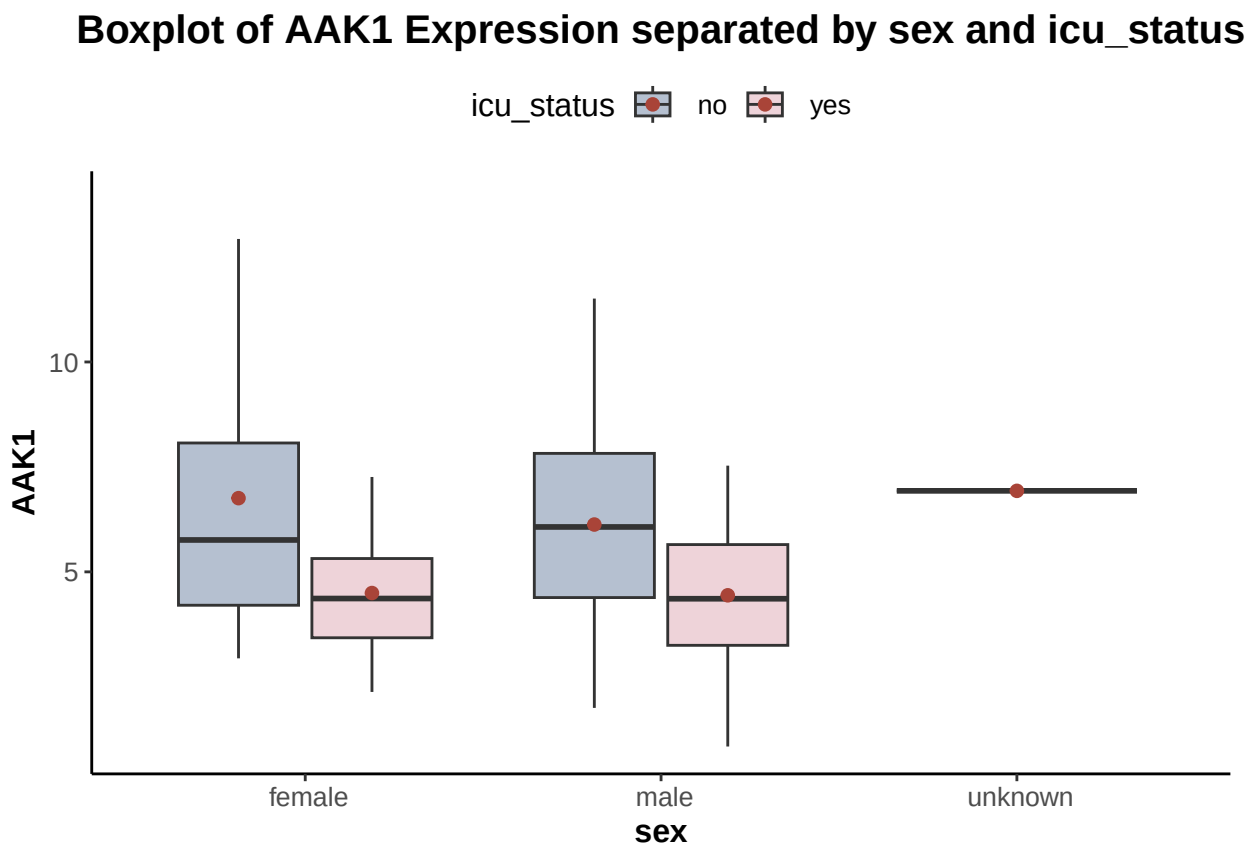
```r
# Display the plots
for (gene in names(plots)) {
  print(plots[[gene]]$histogram)
  print(plots[[gene]]$scatter)
  print(plots[[gene]]$boxplot)
}
```

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




```
## Warning: Removed 25 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

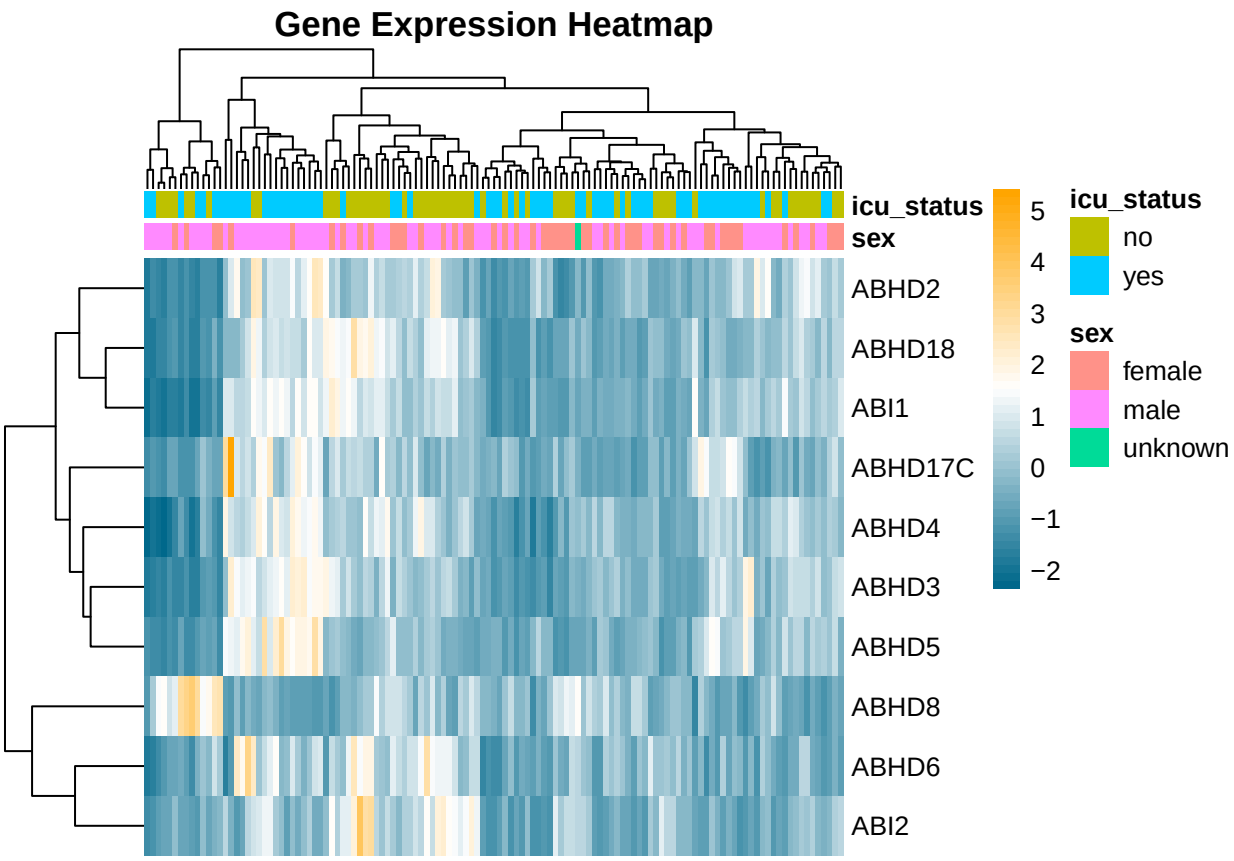

Scatterplot for AAK1 and ddimer
AAK1
10
5
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
ddimer(mg/L_feu)



3.Generate a heatmap (5 pts) 1)Heatmap should include at least 10 genes 2)Include tracking bars for the 2 categorical covariates in your boxplot 3)Heatmaps should include clustered rows and columns

```
# Select at least 10 genes for the heatmap
selected_genes<-c(tail(genes_expression[['X']],10))
metadata <- read.csv("QBS103_GSE157103_series_matrix.csv", header = TRUE, stringsAsFactors = FALSE)

# Filter the dataset to include only the selected genes
gene_data <- genes_expression %>%
  filter(X %in% selected_genes) %>%
  column_to_rownames("X")

# Optionally scale the data
scaled_data <- t(scale(t(gene_data)))

# Merge the metadata for the categorical variables
metadata <- metadata %>%
  select(participant_id, sex, icu_status)


# Prepare annotation data for the heatmap
row_annotation <- metadata %>%
  filter(participant_id %in% colnames(scaled_data)) %>%
  select(participant_id,sex, icu_status) %>%
  as.data.frame()
```

```r
rownames(row_annotation)<-row_annotation[['participant_id']]
row_annotation<-row_annotation[,-1]

# Identify the column names in scaled_data that are not in row_annotation
missing_id <- setdiff(colnames(scaled_data), rownames(row_annotation))

#Remove the column from scaled_data if it's not in the metadata
scaled_data <- scaled_data[, !colnames(scaled_data) %in% missing_id]

#Generate the heatmap
pheatmap(scaled_data,
         annotation_col = row_annotation,
         cluster_rows = TRUE,
         cluster_cols = TRUE,
         display_numbers = FALSE,
         color = colorRampPalette(c("deepskyblue4", "white", "orange"))(50),
         fontsize_row = 10,
         angle_col = 45,
         legend_labels = c("Low", "High"),
         annotation_legend = TRUE,
         annotation_names_col = TRUE,
         annotation_names_row = FALSE,
         border_color = NA,
         treeheight_row = 50,
         treeheight_col = 50,
         main = "Gene Expression Heatmap",
         show_colnames = FALSE
)
```



4.Going through the documentation for ggplot2, generate a plot type that we did not previously discuss in class that describes your data in a new and unique way (5 pts).

```
library(dbplyr)
```

```
##
## Attaching package: 'dbplyr'

## The following objects are masked from 'package:dplyr':
##
##     ident, sql
```

```
genes_expression_1<-read.csv(file="QBS103_GSE157103_genes.csv",header=T,stringsAsFactors=F)
metadata_1<-read.csv("QBS103_GSE157103_series_matrix.csv", header = T, stringsAsFactors = F)

#Using 'which' to select the chosen gene and identify it
gene_AAK1<-genes_expression[which(genes_expression[,1]=='AAK1'),]

#use pipe and merge to convert the gene expression to required format and link two dataframes
gene_AAK1<-gene_AAK1 %>%
  gather(key=participant_id,value=expression)
linked_data2<-merge(metadata_1,gene_AAK1)
linked_data2$expression<-as.numeric(linked_data2$expression)

linked_data2<-linked_data2%>%
```

```r
  filter(sex!='unknown')

#generate the density plot
ggplot(linked_data2, aes(x = expression, fill = sex)) +
  geom_density(alpha = 0.7) +
  scale_fill_manual(values = c("pink", "lightblue",'darkgreen')) +
  labs(title = "AAK1 Density Plot of Gene Expression Levels by Sex",
       x = "AAK1 Gene Expression Level",
       y = "Density") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, size = 18, face = "bold"),
        axis.text = element_text(size = 12),
        legend.title = element_text(size = 12),
        legend.text = element_text(size = 10))
```

```
## Warning: Groups with fewer than two data points have been dropped.
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```