Assignment 3

Vui Doan

2025-08-02

```
# read.csv2 reads the metadata file which is comma separated.

meta_data <- read.csv2(file = paste0("c:/Users/Jade/Desktop/",
"Dartmouth Classes/R QBS 103/GitRepo/Repository",
"_One/QBS103_GSE157103_series_matrix (1).csv")
,sep = ",")


# The variables are selected and the data.frame is subseted.

selected_variables <- c("age","ferritin.ng.ml.",
                        "procalcitonin.ng.ml..","lactate.mmol.l.")

meta_data_filtered <- meta_data[,selected_variables]


# Apply as.numeric to each column. This will produce NAs for values that
# cannot be directly tranformed.

meta_data_filtered <- data.frame(apply(meta_data_filtered,
                                       2,as.numeric))
```

```
## Warning in apply(meta_data_filtered, 2, as.numeric): NAs introduced by coercion
## Warning in apply(meta_data_filtered, 2, as.numeric): NAs introduced by coercion
## Warning in apply(meta_data_filtered, 2, as.numeric): NAs introduced by coercion
## Warning in apply(meta_data_filtered, 2, as.numeric): NAs introduced by coercion
```

Part One

```
add_vector <- function(vector) {

  # The sum_vector variable is initialized to 0.

  sum_vector <- 0

  # A for loop is use to examine each element in the vector.

  for (element in vector){
```

```
    # If the element is not an NA then it is added to sum_Vector.
    if(!is.na(element)){

      sum_vector <- sum_vector + element

    }

  }

  # The addtion of all the values is return.

  return(sum_vector)
}


# apply is used to add each column of the filtered metadata data.frame.

apply(meta_data_filtered, 2, FUN = add_vector)
```

```
##                   age       ferritin.ng.ml. procalcitonin.ng.ml..
##               7532.00              91687.00                313.93
##       lactate.mmol.l.
##                124.31
```

```
compute_mean <- function(vector){

  # Using the previous function to add
  # all valid vector values.

  summed_vector <- add_vector(vector)

  # Compute n as the number of
  # valid values.

  n <- !is.na(vector)

  n_total <- 0

  # a for loop is used to add one to
  # n_total for each TRUE

  for (value in n){

    if(value){

      n_total <- n_total + 1

    }

  }

  # The sum vector is divided by the number
  # of valid data points and the mean is returned.
```

```
  return(summed_vector / n_total)

}

# This applies our function to each coulumn of
#the data.frame. 2 indicates that the function
# should be applied to
# the data.frame by column.

print(apply(meta_data_filtered,2,compute_mean))
```

```
##                   age       ferritin.ng.ml. procalcitonin.ng.ml..
##             61.235772            833.518182              3.077745
##       lactate.mmol.l.
##              1.462471
```

```
# apply can be used including the value for arguments of the functions.
# In this case na.rm is set to TRUE so that
# The function can compute the eman.

print(apply(meta_data_filtered,2,mean,na.rm = TRUE))
```

```
##                   age       ferritin.ng.ml. procalcitonin.ng.ml..
##             61.235772            833.518182              3.077745
##       lactate.mmol.l.
##              1.462471
```

```
# The same results are obtained.
```

Part 3

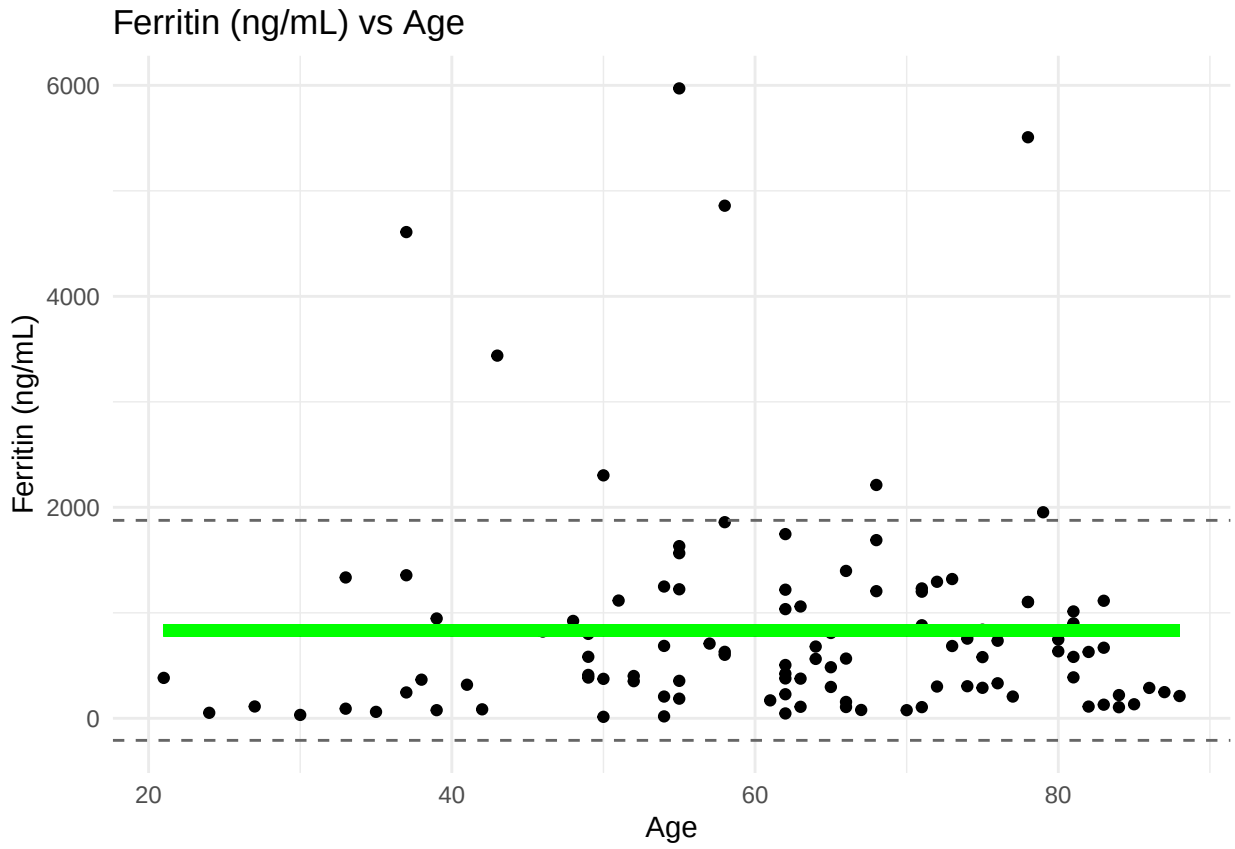
```
library(ggplot2)

plot_scater <- function(data_frame,selected_variable){

   # Regular expression and the sub and gsub functions are
   #used to format the string to produce the y-axis label.

   formatted_Variable_name <- sub("\\.","/",
   sub("\\.", " ", gsub("\\.*$", "", selected_variable)))

   # The first letter and the last letter are converted
   # to uppercase using a combination of toupper (makes capital)
   # and substr (which takes subsets of strings).

   formatted_Variable_name <- paste0(toupper(substr(formatted_Variable_name, 1, 1)),

   substr(formatted_Variable_name, 2, nchar(formatted_Variable_name)-1),
```

```
  toupper(substr(formatted_Variable_name,
                 nchar(formatted_Variable_name), nchar(formatted_Variable_name))))

  # The parethesis are added to the string using gsub and paste0.

  formatted_Variable_name <- paste0(gsub(" ",
  " (",formatted_Variable_name),")")

  # The mean of the selected variable is computed.

  mean_y_axis <- mean(data_frame[[selected_variable]], na.rm = TRUE)

  # The standard deviation is computed.

  sd_y_axis <- sd(data_frame[[selected_variable]], na.rm = TRUE)

  # geom_point() is used to generate the scatter plot.

  plot_out <- ggplot(meta_data_filtered, aes(x = age,
  y = !!sym(selected_variable))) + geom_point() +

  # geom_segment() is used to generate an horizontal
  # bar at the level of the variable mean.

  geom_segment(aes(x = min(age,na.rm = TRUE),
  xend = max(age,na.rm = TRUE), y = mean_y_axis, yend = mean_y_axis),
  color = "green", linewidth = 2) +

  # Dashed lines are used to show standard one stadndard
  # deviation above and below the mean using geom_hline.

  geom_hline(yintercept = mean_y_axis + sd_y_axis,
  linetype = "dashed", color = "gray40") +

  geom_hline(yintercept = mean_y_axis - sd_y_axis,
  linetype = "dashed", color = "gray40") +

  # The formated y and x axis labels are added.

  labs(y = formatted_Variable_name, x = "Age") +

  labs(title = paste0(formatted_Variable_name, " vs Age")) +

  # theme_minimal() is used to remove the background color.

  theme_minimal()

  # The plot is printed.

  print(plot_out)

}
```

```r
# The variables are defined.

variables_to_plot <- c("ferritin.ng.ml.",
                       "procalcitonin.ng.ml..","lactate.mmol.l.")

# A for loop is used to call the function for each
# variable and generate all the plots.

for (var_to_p in variables_to_plot){

  plot_scater(meta_data_filtered,var_to_p)

}
```

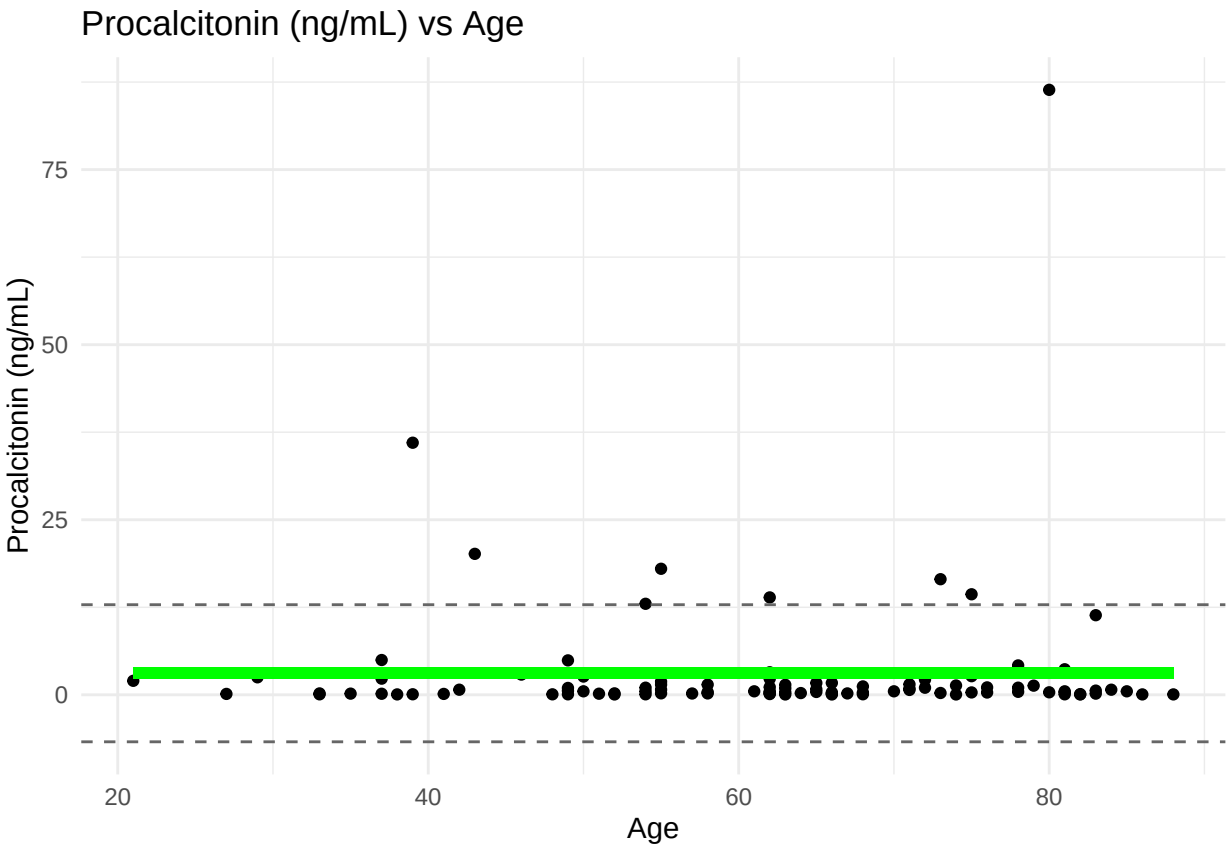
```
## Warning in geom_segment(aes(x = min(age, na.rm = TRUE), xend = max(age, : All aesthetics have length
## i Please consider using ‘annotate()‘ or provide this layer with data containing
##   a single row.
```

```
## Warning: Removed 19 rows containing missing values or values outside the scale range
## (‘geom_point()‘).
```

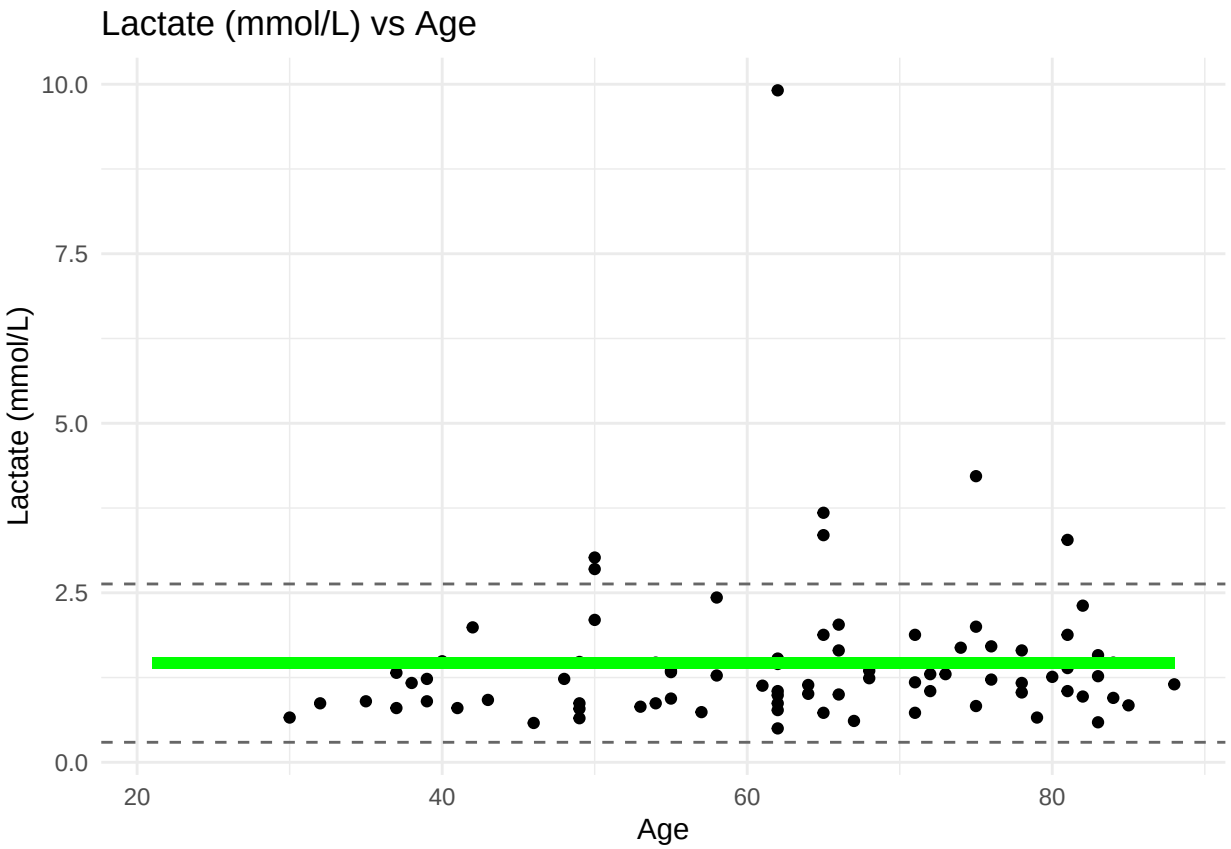



```
## Warning in geom_segment(aes(x = min(age, na.rm = TRUE), xend = max(age, : All aesthetics have length
## i Please consider using ‘annotate()‘ or provide this layer with data containing
##   a single row.
```

```
## Warning: Removed 27 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
## Warning in geom_segment(aes(x = min(age, na.rm = TRUE), xend = max(age, : All aesthetics have length
## i Please consider using `annotate()` or provide this layer with data containing
##   a single row.

## Warning: Removed 44 rows containing missing values or values outside the scale range
## (`geom_point()`).
```


Lactate (mmol/L) vs Age
Lactate (mmol/L)
10.0
7.5
5.0
2.5
0.0
20
40
60
80
Age