Data Analytics Problems

2024-08-13

```
# Load the dataset
data <- read.csv("C:/Users/casha/Downloads/QBS103_GSE157103_series_matrix.csv", stringsAsFactors = FALSE

# head(data)
# had to look at results because was throwing an error when trying to decide columns of interest.

# Custom function to calculate sum
function_sum <- function(x) {
  total <- 0
  for (value in x) {
    if (!is.na(value)) { # ACCOUNT FOR NA's
      total <- total + value
    }
  }
  return(total)
}

# Extract columns of interest
# CASE-SENSITIVE AND WILL COME UP WITH 0 if Input Incorrectly
age <- as.numeric(data$age) # make numeric
```

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

```
ferritin <- as.numeric(data$ferritin.ng.ml.) #have to use form from dataframe
```

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

```
procalcitonin <- as.numeric(data$procalcitonin.ng.ml..)
```

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

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

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

```
# Use the function for each column of interest
function_sum_value <- apply(data.frame(age, ferritin, procalcitonin, lactate), 2, function_sum) # 1 ref

# Apply the built-in sum function to each column for comparison
# https://www.digitalocean.com/community/tutorials/sum-in-r
# to not get "NA" have to remove NA values
```

```r
sum_value <- apply(data.frame(age, ferritin, procalcitonin, lactate), 2, function(x) sum(x, na.rm = TRU


# Results should return the SAME values for both types of runs
# Use a list since simple and clean (tried print but would not run)
# alternative ways to display results - https://www.datacamp.com/tutorial/creating-lists-r
results_df <- data.frame(
  Column = c("age", "ferritin", "procalcitonin", "lactate"),

# for my function
  MY_function_sum_answer = c(function_sum_value["age"], function_sum_value["ferritin"], function_sum_va

# for built-in function
  sum_answer = c(sum_value["age"], sum_value["ferritin"], sum_value["procalcitonin"], sum_value["lactat
)

# Print the results data frame
print(results_df)
```

```
##                       Column MY_function_sum_answer sum_answer
## age                      age               7532.00    7532.00
## ferritin           ferritin               91687.00   91687.00
## procalcitonin procalcitonin                 313.93     313.93
## lactate             lactate                 124.31     124.31
```

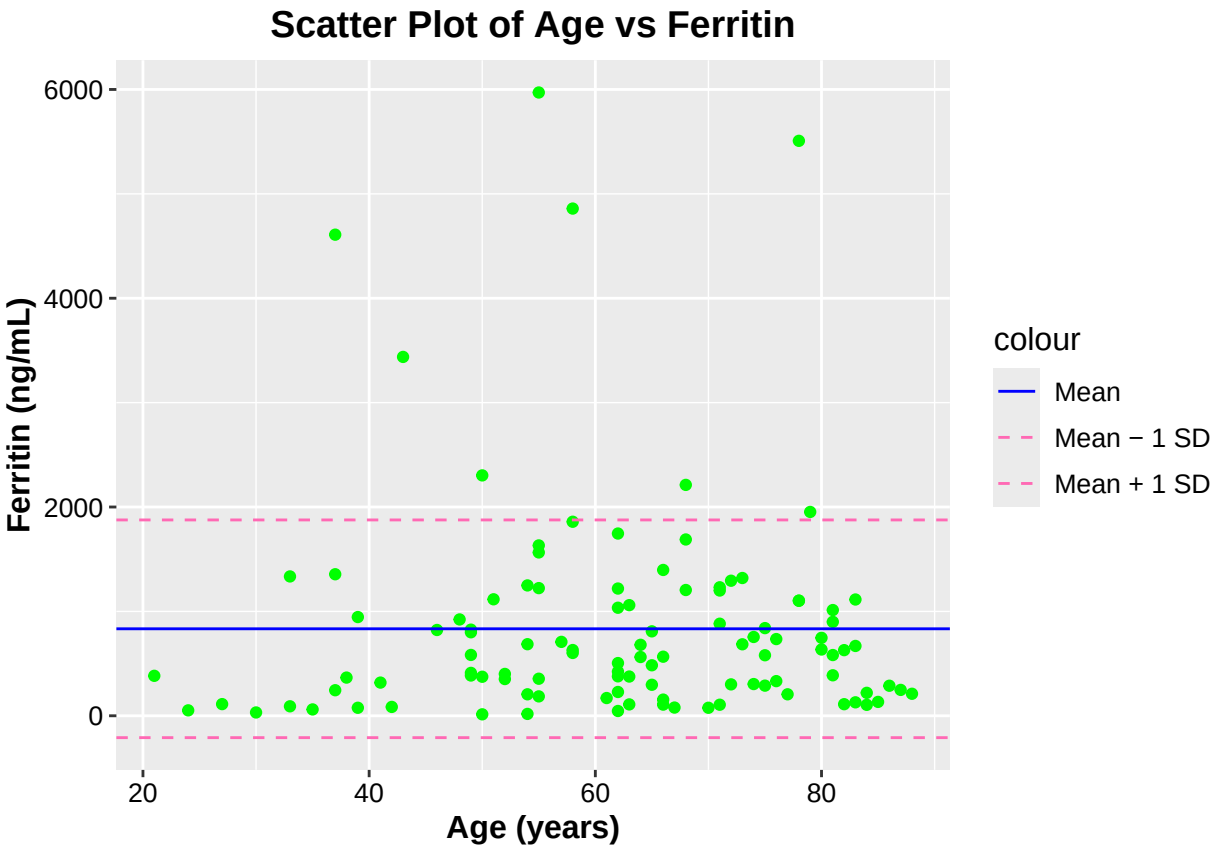
```r
# Custom mean function using manual_sum
function_mean <- function(x) {
  n <- sum(!is.na(x))  # Count of non-NA values
  if (n == 0) return(NA)
  total <- function_sum(x)
  return(total / n)  # Return mean (divide)
}

# Use the function for each column of interest
function_mean_value <- apply(data.frame(age, ferritin, procalcitonin, lactate), 2, function_mean)

# Apply the built-in mean function to each column for comparison
mean_value <- apply(data.frame(age, ferritin, procalcitonin, lactate), 2, function(x) mean(x, na.rm = T

# Results should return the SAME values for both types of runs
# for my function
results_df <- data.frame(
  Column = c("age", "ferritin", "procalcitonin", "lactate"),
  MY_function_mean_answer = c(function_mean_value["age"], function_mean_value["ferritin"], function_mea

# for built-in function
  mean_answer = c(mean_value["age"], mean_value["ferritin"], mean_value["procalcitonin"], mean_value["l
)

# Print the results data frame
print(results_df)
```

```
##                       Column MY_function_mean_answer mean_answer
```

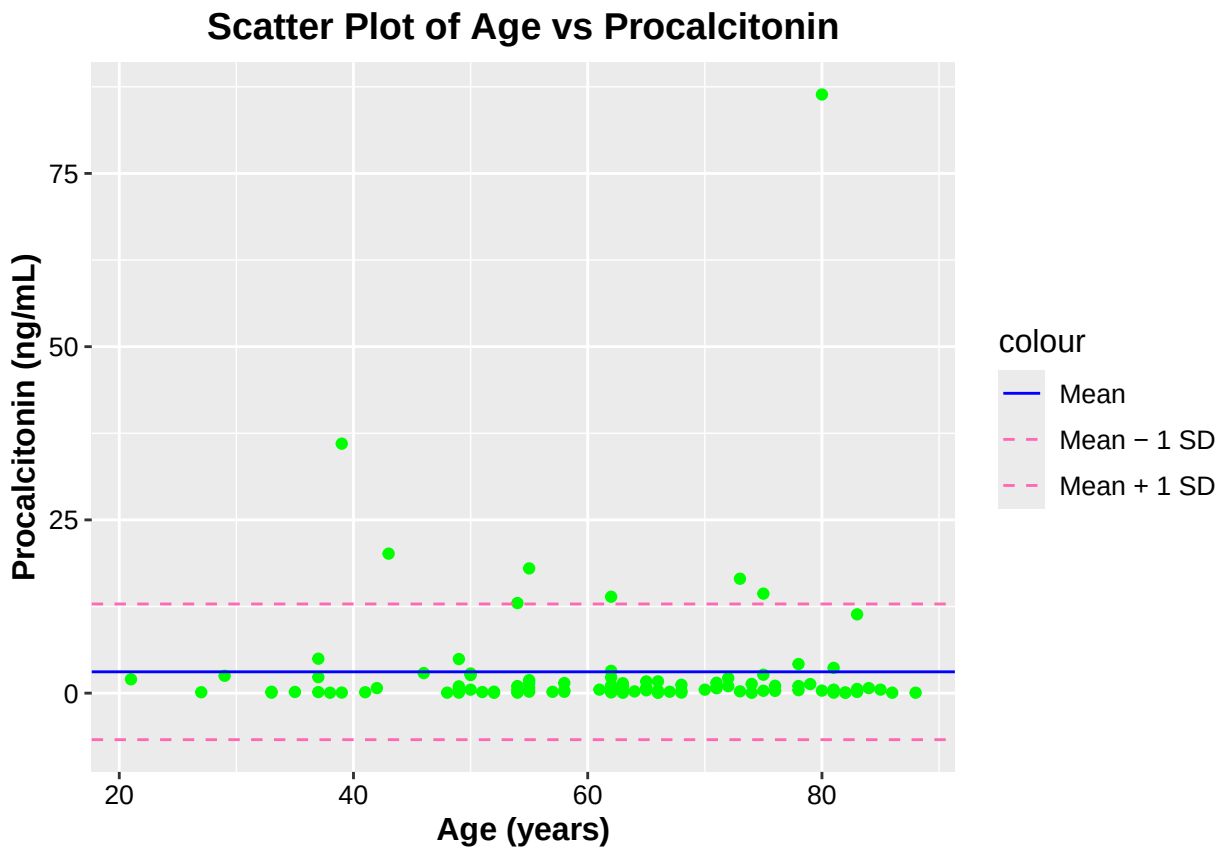
```
## age                       age              61.235772   61.235772
## ferritin             ferritin             833.518182  833.518182
## procalcitonin procalcitonin               3.077745    3.077745
## lactate               lactate               1.462471    1.462471
```

```r
# import package
library(ggplot2)

# an x label variable is unneccesary since it will always be "Age"
create_scatter <- function(x, y, y_label, title) {
  mean_value <- mean(y, na.rm = TRUE)
  sd_value <- sd(y, na.rm = TRUE)

  plot <- ggplot(data.frame(x, y), aes(x = x, y = y)) +
    geom_point(color = "green") +
    scale_color_manual(values = c("Mean" = "blue", "Mean + 1 SD" = "hotpink", "Mean - 1 SD" = "hotpink")
    geom_hline(aes(yintercept = mean_value, color = "Mean")) +

    # standard deviation by definition is plus from the mean
    geom_hline(aes(yintercept = mean_value + sd_value, color = "Mean + 1 SD"), linetype = "dashed") +

    # and subtraction from the mean
    geom_hline(aes(yintercept = mean_value - sd_value, color = "Mean - 1 SD"), linetype = "dashed") +

# y label and title will change, x label will not
    labs(x = "Age (years)", y = y_label, title = title) +

    theme(
      text = element_text(size = 12),
      plot.title = element_text(hjust = 0.5, face = "bold", size = 14),
      axis.title = element_text(face = "bold"),
      axis.text = element_text(color = "black"),
    )

  print(plot)
}

# identify variables for each plot and give desciptive y-axis label
y_definitions <- list(
  ferritin = list(data = ferritin, label = "Ferritin (ng/mL)", title = "Scatter Plot of Age vs Ferritin
  procalcitonin = list(data = procalcitonin, label = "Procalcitonin (ng/mL)", title = "Scatter Plot of
  lactate = list(data = lactate, label = "Lactate (mmol/L)", title = "Scatter Plot of Age vs Lactate")
)

# a for loop to go through all variables of interest
for (var in y_definitions) {
  create_scatter(age, var$data, var$label, var$title)
}
```

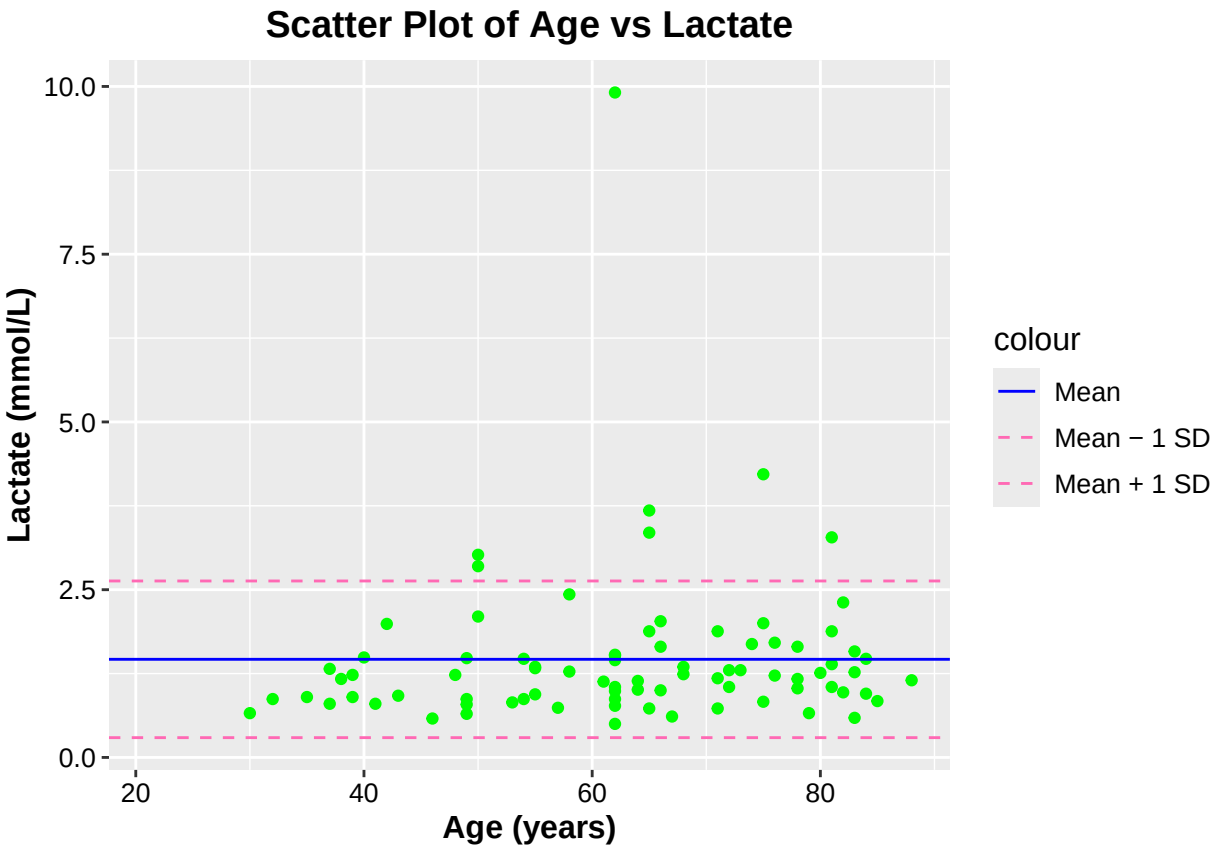
```
## Warning: Removed 19 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




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




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


Scatter Plot of Age vs Lactate
Lactate (mmol/L)
10.0
7.5
5.0
2.5
0.0
20
40
60
80
Age (years)
colour
Mean
Mean − 1 SD
Mean + 1 SD