STATS506HW1

Problem 1.

(a)

```
df <- read.csv("abalone/abalone.data")
colnames(df) <- c("Sex", "Length", "Diameter", "Height", "Whole weight", "Shucked weight", "V
head(df)
```

```
  Sex Length Diameter Height Whole weight Shucked weight Viscera weight
1   M  0.350    0.265  0.090       0.2255         0.0995         0.0485
2   F  0.530    0.420  0.135       0.6770         0.2565         0.1415
3   M  0.440    0.365  0.125       0.5160         0.2155         0.1140
4   I  0.330    0.255  0.080       0.2050         0.0895         0.0395
5   I  0.425    0.300  0.095       0.3515         0.1410         0.0775
6   F  0.530    0.415  0.150       0.7775         0.2370         0.1415
  Shell weight Rings
1        0.070     7
2        0.210     9
3        0.155    10
4        0.055     7
5        0.120     8
6        0.330    20
```

(b)

```
table(df$Sex)
```

```
   F    I    M
1307 1342 1527
```

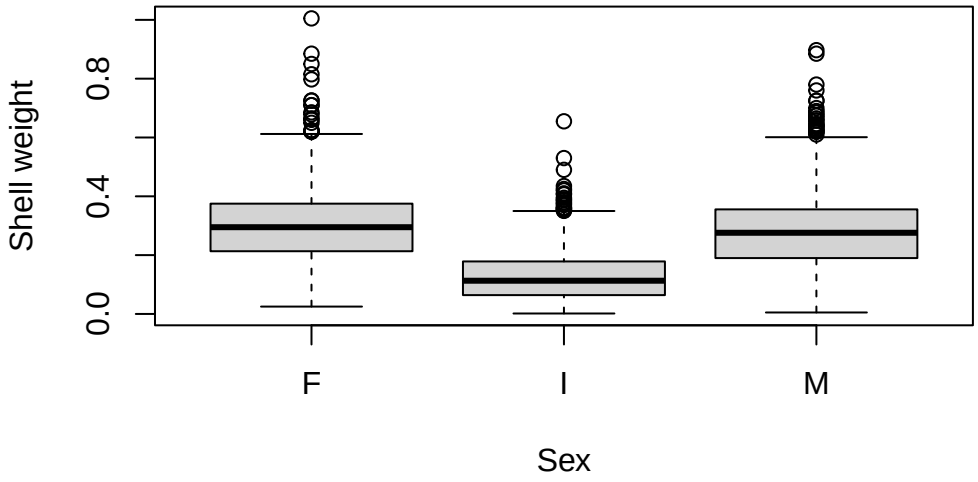
In the Abalone dataset, 1,307 are female, 1,342 are infants, and 1,528 are male.

(c)

```
cor(df$Rings, df[, c("Whole weight", "Shucked weight", "Viscera weight", "Shell weight")])
```

```
     Whole weight Shucked weight Viscera weight Shell weight
[1,]    0.5408179      0.4212556      0.5042735    0.6280306
```

```
boxplot(`Shell weight` ~ Sex, data = df,
        ylab = "Shell weight", xlab = "Sex")
```




```
df$`Whole weight`[max(df$Rings)]
```

```
[1] 0.8635
```

```
df$`Shucked weight` [max(df$Rings)]
```

```
[1] 0.393
```

```r
df$`Viscera weight` [max(df$Rings)]
```

```
[1] 0.227
```

```r
df$`Shell weight` [max(df$Rings)]
```

```
[1] 0.2
```

```r
sum(df$`Viscera weight` > df$`Shell weight`)/length(df$Sex)
```

```
[1] 0.0651341
```

1. Shell weight has the highest correlation with rings. (0.63)
2. Based on the boxplot above, Females and Males have stronger correlation with shell weight as their medians are higher and the distributions are wider.
3. The correlation are, whole weight = 0.8635, shucked weight = 0.393, viscera weight = 0.227 and shell weight = 0.2
4. Approximately 6.5% of abalones have a viscera weight larger than their shell weight

(d)

```r
abalone_weight <- c("Whole weight", "Shucked weight", "Viscera weight", "Shell weight")
cor_female <- sapply(abalone_weight, function(weight) cor(df[df$Sex == "F", weight], df[df$Se
cor_male <- sapply(abalone_weight, function(weight) cor(df[df$Sex == "M", weight], df[df$Sex
cor_infant <- sapply(abalone_weight, function(weight) cor(df[df$Sex == "I", weight], df[df$Se

correlation_table <- rbind(Female = cor_female,
                   Infant = cor_infant,
                   Male   = cor_male)
correlation_table
```

```
       Whole weight Shucked weight Viscera weight Shell weight
Female    0.2667585     0.09484802      0.2116154    0.4059070
Infant    0.6963268     0.62024577      0.6732727    0.7254357
Male      0.3735125     0.22347447      0.3223073    0.5124437
```

(e)

```r
summary(lm(Rings ~ Sex, data = df))
```

```
Call:
lm(formula = Rings ~ Sex, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-7.7027 -1.8905 -0.7027  1.1095 17.8707

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 11.12930    0.08012 138.908  < 2e-16 ***
SexI        -3.23884    0.11257 -28.773  < 2e-16 ***
SexM        -0.42662    0.10915  -3.909 9.43e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.897 on 4173 degrees of freedom
Multiple R-squared:  0.193, Adjusted R-squared:  0.1926
F-statistic:   499 on 2 and 4173 DF,  p-value: < 2.2e-16
```

A linear regression model was applied with Rings as the dependent variable and Sex as a factor. The results indicate that Sex has a significant effect on the mean number of Rings. Infants had substantially fewer Rings compared to Females, while Males also had a lower mean number of Rings than Females. However, the difference is small.

Problem 2.

(a)

```r
df_food <- read.csv("food_expenditure.csv")
```

(b)

```r
colnames(df_food) <- c("ID", "age", "household", "state", "currency", "total_food_expenditure
```

(c)

```r
dim(df_food)
```

```
[1] 262  12
```

```
df_food_usd <- df_food[df_food$currency == "USD", ]
dim(df_food_usd)
```

```
[1] 230  12
```

The number of rows decreased by 32 after applying the USD currency restriction.

(d)

```
df_food_usd <- df_food_usd[!is.na(df_food_usd$age) & df_food_usd$age > 8 & df_food_usd$age <
```

I eliminated ages below 8 years and above 100 years because respondents in these groups may not fully understand the questions or provide reliable answers. In addition, I eliminated rows with NA values since the information was missing.

(e)

```
df_food_usd <- df_food_usd[!is.na(df_food_usd$state) & df_food_usd$state != "", ]
```

I eliminated rows with NA values and empty strings, as the information was missing.

(f)

```
cols <- c("total_food_expenditure", "food_expenditure_grocery_store", "food_expenditure_dini

df_food_usd[cols] <- lapply(df_food_usd[cols], function(x) {x[x == ""] <- NA
x})


df_food_usd <- df_food_usd[!is.na(df_food_usd$total_food_expenditure) &
                           !is.na(df_food_usd$food_expenditure_grocery_store) &
                           !is.na(df_food_usd$food_expenditure_dining) &
                           !is.na(df_food_usd$food_expenditure_miscellaneous), ]

df_food_usd[cols] <- lapply(df_food_usd[cols], function(x) {
  x <- gsub("[^0-9\\.-]", "", x)
  as.numeric(x)
})
```

I eliminated rows with NA values, empty strings, and non-numeric values, as these represent missing or invalid information that cannot be used in calculations.

(g)

```
df_food_usd <- df_food_usd[!is.na(df_food_usd$dine_out_count) & df_food_usd$dine_out_count !=
```

I eliminated rows with NA values and empty strings, as the information was missing.

(h)

```
dim(df_food_usd)
```

```
[1] 163  12
```

The total number of rows after data cleaning is 196.

Problem 3.

(a)

```
#' This function generates the next Collatz value. If the argument is not positive
#' integer, the function will stop and print error message. If the number is even,
#' the next number is obtained by dividing by 2. If it is odd, the next number is
#' obtained by multiplying by 3 and adding 1.
#'
#' @param num A positive integer, the current value of Collatz sequence.
#'
#' @return A positive integer of the next value in the Collatz sequence.
nextCollatz <- function(num){
  if(num != as.integer(num)){
    stop("Argument should be an integer")
  }else if(num <= 0){
    stop("Argument should be positive")
  }

  if(num %% 2 == 0){
    num <- num/2
  }else{
    num <- 3*num + 1
  }
  return(num)
}
```

```r
nextCollatz(5)
```

```
[1] 16
```

```r
nextCollatz(16)
```

```
[1] 8
```

(b)

```r
#' This function generates the Collatz sequence starting from a positive input
#' integer and ending at 1. If the number is even, the next number is obtained by
#' dividing by 2. If it is odd, the next number is obtained by multiplying by 3 and
#' adding 1. The while loop stops until the value reaches 1.
#'
#' @param num A positive integer, the starting value of the Collatz sequence.
#'
#' @return An integer vector containing the Collatz sequence starting
#'   from num and ending at 1.
collatzSequence <- function(num){
  collatz_seq <- c(num)
  while (num != 1){
   num <- nextCollatz(num)
   collatz_seq <- c(collatz_seq, num)
  }
  return(collatz_seq)
}
```

```r
collatzSequence(5)
```

```
[1]  5 16  8  4  2  1
```

```r
collatzSequence(19)
```

```
 [1] 19 58 29 88 44 22 11 34 17 52 26 13 40 20 10  5 16  8  4  2  1
```

(c)

```r
short_len <- 100
short_start <- NA
long_len <- 0
long_start <- NA

for (i in 100:500) {
  seq_len <- length(collatzSequence(i))

  if (seq_len < short_len) {
    short_len <- seq_len
    short_start <- i
  }

  if (seq_len > long_len) {
    long_len <- seq_len
    long_start <- i
  }
}
print(paste("Shortest starts at", short_start, "with length", short_len))
```

```
[1] "Shortest starts at 128 with length 8"
```

```r
print(paste("Longest starts at", long_start, "with length", long_len))
```

```
[1] "Longest starts at 327 with length 144"
```