STATS 506 HW 1

Calder Moore

STATS 506 Homework 1

Problem 1 - Abalone Data

a)

```
#a)
#Set directory and load data
setwd("C:/Users/moore/Desktop/School/Master/STATS 506/STATS-506-HW-1/abalone")

abalone <- read.csv("abalone.data")

#Rename columns
colnames(abalone)[1:9] <- c("Sex",
                            "Length",
                            "Diameter",
                            "Height",
                            "Whole weight",
                            "Shucked weight",
                            "Viscera weight",
                            "Shell weight",
                            "Rings")
```

b)

```
#b)
table(abalone$Sex)
```

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

There are 1,307 females, 1,526 males, and 1,342 infants.

c) #1

```r
#c)

#1)
#Using covariance function between Rings and each weight
cov(abalone$Rings, abalone$`Whole weight`)
```

```
[1] 0.8549952
```

```r
cov(abalone$Rings, abalone$`Shucked weight`)
```

```
[1] 0.3014396
```

```r
cov(abalone$Rings, abalone$`Viscera weight`)
```

```
[1] 0.1781965
```

```r
cov(abalone$Rings, abalone$`Shell weight`)
```

```
[1] 0.2818386
```

Whole weight has the greatest correlation with Rings, with a correlation of 0.855.

c) #2

```r
#2)
cov(abalone$Sex == "F", abalone$`Whole weight`)
```

```
[1] 0.0681564
```

```r
cov(abalone$Sex == "M", abalone$`Whole weight`)
```

```
[1] 0.05960039
```

```r
cov(abalone$Sex == "I", abalone$`Whole weight`)
```

```
[1] -0.1277568
```

Being infant has the strongest correlation with Whole weight, with a correlation of -0.128. It is the greatest correlation in terms of magnitude, the correlation is more strongly negative than the others are positive.

c) #3

```r
#3)
#Find the observation with the most Rings, and pulling the values for each weight
#from that observation
max_weights <- c(abalone[which.max(abalone$Rings),]["Whole weight"],
                 abalone[which.max(abalone$Rings),]["Shucked weight"],
                 abalone[which.max(abalone$Rings),]["Viscera weight"],
                 abalone[which.max(abalone$Rings),]["Shell weight"])

max_weights
```

```
$`Whole weight`
[1] 1.8075

$`Shucked weight`
[1] 0.7055

$`Viscera weight`
[1] 0.3215

$`Shell weight`
[1] 0.475
```

Whole weight is 1.8075, Shucked weight is 0.7055, Viscera weight is 0.3215, Shell weight is 0.475

c) #4

```
#4)
#Count the observations with a viscera weight greater than shell weight, and divide
#by the total number of observations
weight_prop <- sum(abalone$`Viscera weight` > abalone$`Shell weight`)/nrow(abalone)
weight_prop
```

```
[1] 0.0651341
```

Only ~6.5% of the observations have a Viscera weight larger than their Shell weight.

d)

```
#d)
#Pull out the different weight values by sex
cor_F <- c(cov(abalone$Sex == "F", abalone$`Whole weight`),
           cov(abalone$Sex == "F", abalone$`Shucked weight`),
           cov(abalone$Sex == "F", abalone$`Viscera weight`),
           cov(abalone$Sex == "F", abalone$`Shell weight`))

cor_M <- c(cov(abalone$Sex == "M", abalone$`Whole weight`),
           cov(abalone$Sex == "M", abalone$`Shucked weight`),
           cov(abalone$Sex == "M", abalone$`Viscera weight`),
           cov(abalone$Sex == "M", abalone$`Shell weight`))

cor_I <- c(cov(abalone$Sex == "I", abalone$`Whole weight`),
           cov(abalone$Sex == "I", abalone$`Shucked weight`),
           cov(abalone$Sex == "I", abalone$`Viscera weight`),
           cov(abalone$Sex == "I", abalone$`Shell weight`))

#Create the table with the weights split out by sex, and rename the rows and columns
#for easier reading
sex_weight_table <- rbind(cor_F, cor_M, cor_I)
colnames(sex_weight_table) <- c("Whole weight", "Shucked weight",
                                "Viscera weight", "Shell weight")
rownames(sex_weight_table) <- c("Female", "Male", "Infant")
sex_weight_table
```

```
       Whole weight Shucked weight Viscera weight Shell weight
Female   0.06815640     0.02716934     0.01567647   0.01977180
Male     0.05960039     0.02694935     0.01280370   0.01580163
Infant  -0.12775680    -0.05411869    -0.02848017  -0.03557343
```

e)

```r
#Create subsets of the data that only include the certain sex pairs so that we
#can run the t-tests for each pair: F and M, F and I, and M and I.

abalone_FM <- subset(abalone, abalone$Sex %in% c("F", "M"))
abalone_FI <- subset(abalone, abalone$Sex %in% c("F", "I"))
abalone_MI <- subset(abalone, abalone$Sex %in% c("M", "I"))

ttest_FM <- t.test(abalone_FM$Rings ~ abalone_FM$Sex)
ttest_FI <- t.test(abalone_FI$Rings ~ abalone_FI$Sex)
ttest_MI <- t.test(abalone_MI$Rings ~ abalone_MI$Sex)

ttest_FM
```

```
    Welch Two Sample t-test

data:  abalone_FM$Rings by abalone_FM$Sex
t = 3.69, df = 2741.8, p-value = 0.0002286
alternative hypothesis: true difference in means between group F and group M is not equal to
95 percent confidence interval:
 0.1999174 0.6533201
sample estimates:
mean in group F mean in group M
       11.12930        10.70269
```

```r
ttest_FI
```

```
    Welch Two Sample t-test

data:  abalone_FI$Rings by abalone_FI$Sex
t = 29.477, df = 2508.9, p-value < 2.2e-16
alternative hypothesis: true difference in means between group F and group I is not equal to
```

```
95 percent confidence interval:
 3.023380 3.454304
sample estimates:
mean in group F mean in group I 
      11.129304        7.890462 
```

```
ttest_MI
```

```


    Welch Two Sample t-test

data:  abalone_MI$Rings by abalone_MI$Sex
t = -27.194, df = 2857.9, p-value < 2.2e-16
alternative hypothesis: true difference in means between group I and group M is not equal to
95 percent confidence interval:
 -3.014995 -2.609451
sample estimates:
mean in group I mean in group M 
       7.890462       10.702685 
```

The difference in rings between females and males was the smallest. Both females and males had significantly different numbers of rings from infants.

Problem 2 - Food Expenditure Data

a)

```
#Load data
setwd("C:/Users/moore/Desktop/School/Master/STATS 506/STATS-506-HW-1")

food <- read.csv("food_expenditure.csv")
```

b)

```
#Rename variables
colnames(food) <- c("ID",
                    "Age",
                    "Household Members",
                    "State",
                    "Currency",
                    "Total Expenditures",
                    "Grocery Store Expenditures",
                    "Dining Out Expenditures",
                    "Misc Expenditures",
                    "Dine Out Count",
                    "Alcohol",
                    "Assistance Program")
```

c)

```
#Count observations before currency restriction
nrow(food)
```

```
[1] 262
```

```
#Restrict to only USD
food_USD <- subset(food, food$Currency %in% c("USD"))

#Recheck observation count
nrow(food_USD)
```

```
[1] 230
```

Number of observations decreased by 32.

d-h)

```
#d) Remove under 18 since they probably don't have a great sense of home expenditures
#usually, and also remove the couple of people who are apparently 150 since that seems
#sort of unlikely

food_clean <- subset(food_USD, food_USD$Age %in% c(18:100))

#e) Remove the observations with a state of XX
food_clean <- subset(food_clean, !food_clean$State %in% c("XX"))
```

```
#f) In total expenditures, there is an observation with "~350" rather than just 350.
#There are some negative values in the expenditure variables as well, and I'll remove
#them here but I'm not sure if it's great practice unless we're certain that they are
#the result of a data collection error.
food_clean$`Total Expenditures`[which(food_clean$`Total Expenditures` == "~350")] <- 350
food_clean <- subset(food_clean, food_clean$`Total Expenditures` > 0)
food_clean <- subset(food_clean, food_clean$`Grocery Store Expenditures` > 0)
food_clean <- subset(food_clean, food_clean$`Dining Out Expenditures` > 0)
food_clean <- subset(food_clean, food_clean$`Misc Expenditures` > 0)

#g) There are some entries in dine out count that seem excessive.
#Eating out 30 times in a week is crazy for example, so I'll remove observations with
#a count of 20 and over, but there are conceivably some people who would do it.
#I again don't necessarily think that removing outliers just because they're outliers
#is the best practice.

food_clean <- subset(food_clean, food_clean$`Dine Out Count` < 20)

#h)
nrow(food_clean)
```

```
[1] 116
```

Only 116 observations left after cleaning compared to an original 262.

Problem 3 - Collatz Conjecture

a)

```
#' Function to find the next value in the Collatz Conjecture sequence
#'
#' @param integer
#'
#' @returns the next number in the Collatz Conjecture sequence. Gives errors requesting
#' numeric and integer inputs if the user attempts a non numeric or negative.
#' Rounds floats to nearest integer value smaller than it.
#' @examples an input of 5 returns 16, of 16 returns 8 according to the rule
nextCollatz <- function(integer){
  #Check for numbers
  if(is.numeric(integer)){
    #check for positive numbers
    if(integer > 0){
      #Check for evens
      if(round(integer) %% 2 == 0){
        next_int <- round(integer)/2
      }
      #Check for odds
      else if(round(integer) %% 2 != 0){
        next_int <- 3*round(integer) + 1
      }
      return(next_int)
    }
    else{
      return("Input must be positive")
    }
  }
  else{
    return("Input must be numeric")
  }
}

#Test
nextCollatz(5)
```

```
[1] 16
```

```
nextCollatz(16)
```

```
[1] 8
```

b)

```
collatzSequence <- function(integer){
  seq <- c(integer)
  next_int <- nextCollatz(integer)
  while(next_int > 1){
    seq <- c(seq, next_int)
    next_int <- nextCollatz(next_int)
```

```r
    #Making sure it includes the final "1" to end the sequence
    if(next_int == 1){
      seq <- c(seq, next_int)
      break
    }
  }
  return(seq)
}

#Test
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
collatzList <- c()

for(i in 100:500){
  collatzList <- c(collatzList, length(collatzSequence(i)))
}
names(collatzList) <- c(100:500)

collatzList[which(collatzList == max(collatzList))]
```

```
327
144
```

```r
collatzList[which(collatzList == min(collatzList))]
```

```
128
  8
```

327 gives the longest sequence with a length of 144, and 128 has the shortest sequence, with a length of 8.