Stats-506 Problem Set #1

GitHub Repository: https://github.com/zkl2002/Stats-506/

Zhekai Liu

Problem 1 - Abalone Data

(a).

```
abalone <- read.csv("abalone/abalone.data", header = FALSE)

colnames(abalone) <- c("Sex", "Length", "Diameter", "Height",
                       "WholeWeight", "ShuckedWeight", "VisceraWeight",
                       "ShellWeight", "Rings")
```

```
head(abalone)
```

```
  Sex Length Diameter Height WholeWeight ShuckedWeight VisceraWeight
1   M  0.455    0.365  0.095      0.5140        0.2245        0.1010
2   M  0.350    0.265  0.090      0.2255        0.0995        0.0485
3   F  0.530    0.420  0.135      0.6770        0.2565        0.1415
4   M  0.440    0.365  0.125      0.5160        0.2155        0.1140
5   I  0.330    0.255  0.080      0.2050        0.0895        0.0395
6   I  0.425    0.300  0.095      0.3515        0.1410        0.0775
  ShellWeight Rings
1       0.150    15
2       0.070     7
3       0.210     9
4       0.155    10
5       0.055     7
6       0.120     8
```

(b).

```
table(abalone$Sex)
```

```
   F    I    M
1307 1342 1528
```

Female: 1,307, Infant: 1,342, Male: 1,528.

(c).

(1).

```
weight_cors <- c(
  WholeWeight = cor(abalone$WholeWeight, abalone$Rings),
  ShuckedWeight = cor(abalone$ShuckedWeight, abalone$Rings),
  VisceraWeight = cor(abalone$VisceraWeight, abalone$Rings),
  ShellWeight = cor(abalone$ShellWeight, abalone$Rings)
)
weight_cors
```

```
  WholeWeight ShuckedWeight VisceraWeight   ShellWeight
    0.5403897     0.4208837     0.5038192     0.6275740
```

Shell weight has the highest correlation with rings.

(2).

```
sex_cors <- c(
  Male = cor(abalone$ShellWeight[abalone$Sex == "M"],
             abalone$Rings[abalone$Sex == "M"]),
  Female = cor(abalone$ShellWeight[abalone$Sex == "F"],
               abalone$Rings[abalone$Sex == "F"]),
  Infant = cor(abalone$ShellWeight[abalone$Sex == "I"],
               abalone$Rings[abalone$Sex == "I"])
  )
sex_cors
```

```
     Male    Female    Infant
0.5109967 0.4059070 0.7254357
```

For Shell weight, infant has the highest correlation with rings.

(3).

```
max_rings <- max(abalone$Rings)

abalone_max_rings <- abalone[abalone$Rings == max_rings,
                             c("WholeWeight", "ShuckedWeight",
                               "VisceraWeight","ShellWeight")]

abalone_max_rings
```

```
    WholeWeight ShuckedWeight VisceraWeight ShellWeight
481      1.8075        0.7055        0.3215       0.475
```

For abalone with most rings, its whole weight is 1.8075; shucked weight is 0.7055; viscera weight is 0.3215; shell weight is 0.475.

(4).

```
pct <- mean(abalone$VisceraWeight > abalone$ShellWeight) * 100

pct
```

```
[1] 6.511851
```

About 6.512% abalones have a viscera weight larger than their shell weight.

(d).

```r
weight_cols <- c("WholeWeight", "ShuckedWeight", "VisceraWeight", "ShellWeight")

# function to compute correlation with Rings
# input:  w - a column name from weight_cols
# output: the correlation between that column and Rings
M <- sapply(weight_cols, function(w) cor(abalone[abalone$Sex == "M", w],
                                         abalone$Rings[abalone$Sex == "M"]))

F <- sapply(weight_cols, function(w) cor(abalone[abalone$Sex == "F", w],
                                         abalone$Rings[abalone$Sex == "F"]))

I <- sapply(weight_cols, function(w) cor(abalone[abalone$Sex == "I", w],
                                         abalone$Rings[abalone$Sex == "I"]))

cors <- rbind(Male = M,
              Female = F,
              Infant = I)

cors
```

```
       WholeWeight ShuckedWeight VisceraWeight ShellWeight
Male     0.3721966    0.22239382     0.3209535   0.5109967
Female   0.2667585    0.09484802     0.2116154   0.4059070
Infant   0.6963268    0.62024577     0.6732727   0.7254357
```

(e).

```r
rings_M <- abalone$Rings[abalone$Sex == "M"]
rings_F <- abalone$Rings[abalone$Sex == "F"]
rings_I <- abalone$Rings[abalone$Sex == "I"]
```

```r
# F vs M
t.test(rings_F, rings_M)
```

```
    Welch Two Sample t-test

data:  rings_F and rings_M
t = 3.6657, df = 2742.4, p-value = 0.0002514
```

```
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.1971045 0.6505082
sample estimates:
mean of x mean of y
  11.1293   10.7055
```

```r
# I vs M
t.test(rings_I, rings_M)
```

```
    Welch Two Sample t-test

data:  rings_I and rings_M
t = -27.221, df = 2859, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.017808 -2.612263
sample estimates:
mean of x mean of y
 7.890462 10.705497
```

```r
# F vs I
t.test(rings_F, rings_I)
```

```
    Welch Two Sample t-test

data:  rings_F and rings_I
t = 29.477, df = 2508.9, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 3.023380 3.454304
sample estimates:
mean of x mean of y
11.129304  7.890462
```

From the three t-tests, we could find that the mean number of rings differs significantly among the three sexes of abalone.

Problem 2 - Food Expenditure Data

(a).

```
food <- read.csv("food_expenditure.csv", stringsAsFactors = FALSE)
```

(b).

```
colnames(food) <- c(
  "id",
  "age",
  "household_size",
  "state",
  "currency",
  "food_exp_total",
  "food_exp_grocery",
  "food_exp_diningout",
  "food_exp_misc",
  "dineout_times",
  "include_alcohol",
  "food_program"
)
```

(c).

```
n_before <- nrow(food)
food_usd <- subset(food, currency == "USD")
n_after <- nrow(food_usd)
cat("Number of observations before filtering:", n_before, "\n")
```

```
Number of observations before filtering: 262
```

```
cat("Number of observations after filtering (USD only):", n_after, "\n")
```

```
Number of observations after filtering (USD only): 230
```

(d).

```
food_clean <- subset(food_usd, !is.na(age) & age >= 18 & age < 100)
```

For the age variable, we excluded missing values and retained only respondents between 18 and 100 years old.

(e).

```
unique(food_clean$state)
```

```
 [1] "LA" "WA" "MS" "AK" "IN" "DC" "ID" "HI" "ND" "UT" "NV" "WI" "VA" "RI" "MT"
[16] "IL" "OR" "CT" "GA" "NC" "TX" "NE" "NY" "CO" "WY" "FL" "AZ" "NM" "MI" "MN"
[31] ""   "OH" "OK" "ME" "AL" "WV" "CA" "KS" "VT" "XX" "PA" "PR" "MO" "NH" "MA"
[46] "MD" "TN" "DE" "SC" "IA" "NJ" "SD" "AR" "KY"
```

```
food_clean <- subset(food_clean, !(state %in% c("", "XX")))
```

By checking the unique state values, we identified empty entries and the code 'XX', which are not valid state abbreviations. These rows were removed from the dataset.

(f).

```
str(food_clean[, c("food_exp_total", "food_exp_grocery",
                   "food_exp_diningout", "food_exp_misc")])
```

```
'data.frame':   186 obs. of  4 variables:
 $ food_exp_total    : chr  "436.35" "" "279.1" "-20.98" ...
 $ food_exp_grocery  : num  169 452 302 140 NA ...
 $ food_exp_diningout: num  140.7 192.9 239.8 69.2 191.7 ...
 $ food_exp_misc     : num  109.8 NA 103.9 44.8 172.3 ...
```

```
food_clean$food_exp_total[food_clean$food_exp_total == ""] <- NA
food_clean$food_exp_total <- suppressWarnings(
                              as.numeric(food_clean$food_exp_total))

food_clean <- subset(food_clean,
                     !is.na(food_exp_total) & !is.na(food_exp_grocery) &
                     !is.na(food_exp_diningout) & !is.na(food_exp_misc) &
                     food_exp_total > 0 & food_exp_grocery >= 0 &
                     food_exp_diningout >= 0 & food_exp_misc >= 0)
```

By checking the variable types, we found that the total expenditure was stored as a character variable, so we converted it into numeric. We then removed all rows with missing values, keeping only those with a strictly positive total expenditure and non-negative values for the other expenditure variables.

(g).

```
typeof(food_clean$dineout_times)
```

```
[1] "integer"
```

```
food_clean <- subset(food_clean,
                     !is.na(dineout_times) &
                     dineout_times >= 0 & dineout_times <= 21)
```

Since dine-out times are already stored as integers, we removed rows with missing values and retained non-negative values and dineout less than 21 per week.

(h).

```
nrow(food_clean)
```

```
[1] 119
```

The final number of observations after this cleaning is 119.

Problem 3 - Palindromic Numbers

(a).

```
# Return the next number in the Collatz sequence
nextCollatz <- function(n) {
  # Input : n - a positive integer
  # Output: the next number in the Collatz sequence of input

  # check input validation
  if (!is.numeric(n) || n %% 1 != 0 || n <= 0)
    stop("n must be a positive integer.")

  # Collatz step
  if (n%%2==0){
    return (n/2)
  }
  else{
    return (3*n+1)
  }
}
```

```
nextCollatz(5)
```

```
[1] 16
```

```
nextCollatz(16)
```

```
[1] 8
```

(b).

```
# Return the full Collatz sequence for a given positive integer
collatzSequence <- function(n) {
  # Input : n - a positive integer
  # Output: the list of the Collatz sequence of input

  # Input validation
```

```r
  if (!is.numeric(n) || n %% 1 != 0 || n <= 0) {
    stop("n must be a positive integer.")
  }

  seq <- n

  while (n != 1) {
    if (n %% 2 == 0) {
      n <- n / 2
    } else {
      n <- 3 * n + 1
    }
    seq <- c(seq, n)
  }

  return(list(
    sequence = seq,
    length = length(seq)
  ))
}
```

```r
collatzSequence(5)
```

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

$length
[1] 6
```

```r
collatzSequence(19)
```

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

$length
[1] 21
```

(c).

```r
min_len <- Inf
min_start <- NA
max_len <- -Inf
max_start <- NA

for (i in 100:500) {
  seq_info <- collatzSequence(i)
  len <- seq_info$length

  if (len < min_len || (len == min_len && i < min_start)) {
    min_len <- len
    min_start <- i
  }

  if (len > max_len || (len == max_len && i < max_start)) {
    max_len <- len
    max_start <- i
  }
}

cat("Shortest Collatz sequence: start =", min_start, "length =", min_len, "\n")
```

```
Shortest Collatz sequence: start = 128 length = 8
```

```r
cat("Longest  Collatz sequence: start =", max_start, "length =", max_len, "\n")
```

```
Longest  Collatz sequence: start = 327 length = 144
```