STATS506 - Assignment 1

Prathibha Muthukumara Prasanna

Table of contents

Problem 1 — Abalone Data

Problem 1a

```r
#loading the raw file into a data frame.
abalone <- read.csv("abalone/abalone.data",header = FALSE)
#assigning proper column names
names(abalone) <- c("Sex", "Length", "Diameter", "Height","WholeWeight",
                    "ShuckedWeight", "VisceraWeight", "ShellWeight","Rings")
```

Problem 1b

```r
table(abalone$Sex) #counting how many abalones are in each sex group
```

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

There are 1,307 females, 1,342 infants, and 1,528 males in the dataset.

Problem 1c (1)

```r
#selecting only the weight columns
weight_variables <- c("WholeWeight", "ShuckedWeight", "VisceraWeight",
                      "ShellWeight")
#computing correlations of each weight with rings
cors <- sapply(abalone[weight_variables], function(x) cor(x, abalone$Rings))
cors
```

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

```r
#finding the strongest correlation
which.max(abs(cors))
```

```
ShellWeight
          4
```

The correlations show that all four weight measures are positively related to the number of rings with ShellWeight having the highest correlation (0.6275740).

Problem 1c (2)

```r
#makes a list of three data frames (one for each sex)
#computes the correlation for each group.
sapply(split(abalone, abalone$Sex),function(df) cor(df$ShellWeight, df$Rings))
```

```
        F         I         M
0.4059070 0.7254357 0.5109967
```

The correlation between shell weight and rings is strongest among infants (~0.73), compared to females (~0.41) and males (~0.51).

Problem 1c (3)

```r
#finding the maximum number of rings
max_rings <- max(abalone$Rings)
#Subsetting rows where rings is equal to that maximum
abalone[abalone$Rings == max_rings, c("WholeWeight", "ShuckedWeight",
                                      "VisceraWeight", "ShellWeight")]
```

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

The abalone with the highest number of rings weighs about ~1.81 (whole weight). Its shucked weight is ~0.71, viscera weight is ~0.32, and shell weight is ~0.48.

Problem 1c (4)

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

```
[1] 6.511851
```

About 6.5% of abalones have a viscera weight greater than their shell weight.

Problem 1d

```r
corr_table <- sapply(weight_variables, function(w)
  {sapply(split(abalone, abalone$Sex),function(df) cor(df[[w]], df$Rings))})
corr_table
```

```
  WholeWeight ShuckedWeight VisceraWeight ShellWeight
F   0.2667585    0.09484802     0.2116154   0.4059070
I   0.6963268    0.62024577     0.6732727   0.7254357
M   0.3721966    0.22239382     0.3209535   0.5109967
```

The correlations between weight measures and rings differ substantially by sex.

Problem 1e

```r
#Female vs Male
t.test(Rings ~ Sex, data = subset(abalone, Sex %in% c("F","M")))
```

```

    Welch Two Sample t-test

data:  Rings by Sex
t = 3.6657, df = 2742.4, p-value = 0.0002514
alternative hypothesis: true difference in means between group F and group M is not equal to
95 percent confidence interval:
 0.1971045 0.6505082
sample estimates:
mean in group F mean in group M 
        11.1293         10.7055 
```

```r
#Female vs Infant
t.test(Rings ~ Sex, data = subset(abalone, Sex %in% c("F","I")))
```

```

    Welch Two Sample t-test

data:  Rings by 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 
```

```r
#Male vs Infant
t.test(Rings ~ Sex, data = subset(abalone, Sex %in% c("M","I")))
```

```

    Welch Two Sample t-test

data:  Rings by Sex
t = -27.221, df = 2859, 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:
 -3.017808 -2.612263
sample estimates:
mean in group I mean in group M
       7.890462       10.705497
```

Pairwise t-tests on the number of rings across the three sexes of abalone tests whether the average number of rings differs significantly between each pair of groups. Results suggest that the mean number of rings differs significantly across the three sexes of abalone.

Problem 2 — Food Expenditure Data

Problem 2a

```r
#loading the raw file into a data frame.
food <- read.csv("food_expenditure.csv", header = TRUE)
```

Problem 2b

```r
#assigning proper column names
names(food) <- c("ID", "Age", "HouseholdSize", "State", "Currency",
                 "FoodExpense", "Grocery", "DiningOut", "Misc",
                 "TimesDiningOut", "Alcohol", "FoodAssistance")
head(food)
```

```
  ID Age HouseholdSize State Currency FoodExpense Grocery DiningOut   Misc
1  1  68             7    LA      USD      436.35  168.59    140.71 109.77
2  2  88             5    WA      USD              452.10    192.94     NA
3  3  82             3    MS      USD       279.1  301.66    239.84 103.94
4  4  73             8    AK      USD      -20.98  139.66     69.19  44.84
5  5  89             0    IN      USD      494.87      NA    191.72 172.31
6  6  18             6    WI      EUR      276.32  394.44    283.20 114.06
  TimesDiningOut Alcohol FoodAssistance
1              4     Yes           None
2              1 Unknown           SNAP
3              9     Yes           None
4              2 Unknown           None
5              3     Yes           None
6              6 Unknown    Food Pantry
```

Problem 2c

```r
obs_before <- nrow(food)
food <- subset(food, Currency == "USD")
obs_after <- nrow(food)
c(Before_Filtering = obs_before, After_Filtering = obs_after)
```

```
Before_Filtering  After_Filtering 
             262              230 
```

After restricting the data to respondents who reported their food expenditures in US dollars, the number of observations decreased from 262 to 230.

Problem 2d

A reasonable rule is to exclude minors (< 18 years old) and exclude implausibly high ages (eg: > 100).

```r
obs_before_age <- nrow(food)
food <- subset(food, Age >= 18 & Age <= 100)
obs_after_age <- nrow(food)
c(Before_AgeFiltering = obs_before_age, After_AgeFiltering = obs_after_age)
```

```
Before_AgeFiltering  After_AgeFiltering 
                230                 196 
```

After applying the age cleaning rule (keeping respondents between 18 and 100 years old), the number of observations decreased from 230 to 196.

Problem 2e

A reasonable rule is to keep only valid US state entries and exclude blank/missing entries.

```r
obs_before_state <- nrow(food)
#keeping rows with exactly 2 uppercase letters in the State column
food <- subset(food, grepl("^[A-Z]{2}$", State))
obs_after_state <- nrow(food)
c(Before_StateFiltering = obs_before_state, After_StateFiltering = obs_after_state)
```

```
Before_StateFiltering  After_StateFiltering 
                  196                   191 
```

After applying the state cleaning rule (keeping valid entries and excluding blank entries), the number of observations decreased from 196 to 191.

Problem 2f

The variables are: FoodExpense, Grocery, DiningOut, Misc A reasonable rule is to exclude rows with negative values and exclude rows with implausibly high values (eg: > $4000 in a single week).

```r
obs_before_food <- nrow(food)
food <- subset(food, FoodExpense >= 0 & FoodExpense <= 4000 & Grocery >= 0
               & Grocery <= 4000 & DiningOut >= 0 & DiningOut <= 4000
               & Misc >= 0 & Misc <= 4000)
obs_after_food <- nrow(food)
c(Before_FoodFiltering = obs_before_food, After_FoodFiltering = obs_after_food)
```

```
Before_FoodFiltering  After_FoodFiltering 
                 191                   58 
```

After applying the expenditure cleaning rule, the number of observations decreased from 191 to 58.

Problem 2g

A reasonble rule is to exclude implausibly large values (eg: > 50 times in one week).

```r
obs_before_dining <- nrow(food)
food <- subset(food, TimesDiningOut >= 0 & TimesDiningOut <= 50)
obs_after_dining <- nrow(food)
c(Before_DiningFiltering = obs_before_dining, After_DiningFiltering = obs_after_dining)
```

```
Before_DiningFiltering  After_DiningFiltering 
                    58                     58 
```

Problem 2h

```r
print(paste0("Final number of observations in the table: ", nrow(food)))
```

```
[1] "Final number of observations in the table: 58"
```

The cleaned dataset consists of 58 respondents who meet all criteria.

Problem 3 — Collatz conjecture

Problem 3a

```r
#' Compute the next number in the Collatz sequence
#'
#' @param n a positive integer
#' @return the next number in the Collatz sequence

nextCollatz <- function(n) {
  #input must be a positive integer
  if (!is.numeric(n) || n <= 0 || n != as.integer(n)) {
    stop("Input must be a positive integer")
  }

  # if even - divide by 2
  if (n %% 2 == 0) {
    return(n / 2)
  }
  # if odd - 3n + 1
  else {
    return(3 * n + 1)
  }
}
```

```r
nextCollatz(5) #example test
```

```
[1] 16
```

```r
nextCollatz(16) #example test
```

```
[1] 8
```

Problem 3b

```r
#' Collatz sequence generator
#'
#' @param n a positive integer
#' @return a list with:
#'   - seq: the Collatz sequence (from n to 1)
#'   - length: length of the sequence

collatzSequence <- function(n) {
  #input must be a positive integer
  if (!is.numeric(n) || length(n) != 1 || n <= 0 || n != as.integer(n)) {
    stop("Input must be a single positive integer")
  }

  seq <- c(n)   #start sequence with n
  while (seq[length(seq)] != 1) {       #check last element using length()
    seq <- c(seq, nextCollatz(seq[length(seq)]))  #append next number
  }

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

```r
collatzSequence(5) #example test
```

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

$length
[1] 6
```

```r
collatzSequence(19) #example test
```

```
$seq
 [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
```

Problem 3c

```r
start_values <- 100:500
#sequence lengths for each start
seq_length <- sapply(start_values, function(n) collatzSequence(n)$length)
#which.min returns the first minimum - lowest start in case of ties
short_start <- start_values[which.min(seq_length)]
short_length <- min(seq_length)
#which.max returns the first maximum - lowest start in case of ties
long_start   <- start_values[which.max(seq_length)]
long_length  <- max(seq_length)
list(
  shortest = list(start = short_start, length = short_length),
  longest  = list(start = long_start,  length = long_length)
)
```

```
$shortest
$shortest$start
[1] 128

$shortest$length
[1] 8


$longest
$longest$start
[1] 327

$longest$length
[1] 144
```

The shortest sequence starts at 128 and has length 8 while the longest sequence starts at 327 and has length 144.

Attribution of Sources

For Problem 1, I used the following references: https://www.geeksforgeeks.org/divide-the-data-into-groups-in-r-programming-split-function/ for exploring simpler alternatives for dividing data frames.
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/subset for understanding subsetting of data frames.

https://www.datacamp.com/tutorial/t-tests-r-tutorial as a reference for performing and interpreting t-tests.

For Problem 2, I used the following reference: https://www.educative.io/answers/what-is-the-grepl-function-in-r while exploring approaches for subsetting valid state codes. This resource helped me understand how to apply grepl() for filtering data based on string patterns.

For Problem 3, I used the following references: https://www.r-bloggers.com/2019/03/learning-r-the-collatz-conjecture/ https://www.r-bloggers.com/2019/07/testing-the-collatz-conjecture-with-r/ helped me understand how to structure functions (nextCollatz, collatzSequence)

Github Repository

https://github.com/prathii7/Computational-Methods-and-Tools-in-Statistics