

STATS-506 HW1

Zekai Xu

2025-08-31

Question 1

Part (a)

```
# Read `abalone.data`
abalone <- read.csv("abalone.data", header = FALSE)
# Manually set data.frame column name
names(abalone) <- c("Sex", "Length", "Diameter", "Height", "Whole weight",
                    "Shucked weight", "Viscera weight", "Shell weight", "Rings")
# Print head of data
head(abalone)
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
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

	Shell weight	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

Part (b)

```
# Count the number of observations belonging to each sex
table(abalone$Sex)
```

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

Part (c)

1.

```
# Compute correlation of all weight columns and rings
corr_mat <- cor(abalone[, - (1:4)])
# Extract correlation between all weight columns and rings
corr_mat <- corr_mat[, "Rings"]
# Display correlation
corr_mat
```

Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0.5403897	0.4208837	0.5038192	0.6275740	1.0000000

Shell weight has the highest correlation with Rings.

2.

```
library(dplyr)
```

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

filter, lag

The following objects are masked from 'package:base':

intersect, setdiff, setequal, union

```
abalone %>%
  group_by(Sex) %>%
  summarise(corr = cor(`Shell weight`, Rings))
```

```
# A tibble: 3 x 2
  Sex    corr
  <chr> <dbl>
1 F     0.406
2 I     0.725
3 M     0.511
```

For Shell weight, sex I (infant) has the highest correlation.

3.

```
abalone %>%
  filter(Rings == max(Rings)) %>%
  select(`Whole weight`, `Shucked weight`, `Viscera weight`, `Shell weight`)
```

```
Whole weight Shucked weight Viscera weight Shell weight
1          1.8075          0.7055          0.3215          0.475
```

4.

```
mean(abalone$`Viscera weight` > abalone$`Shell weight`) * 100
```

```
[1] 6.511851
```

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

Part (d)

```
corr_table <- abalone %>%
  group_by(Sex) %>%
  summarize(across(c(`Whole weight`, `Shucked weight`, `Viscera weight`, `Shell weight`),
    ~ cor(.x, Rings)))
corr_table
```

```
# A tibble: 3 x 5
  Sex    `Whole weight` `Shucked weight` `Viscera weight` `Shell weight`
  <chr>      <dbl>          <dbl>          <dbl>          <dbl>
1 F          0.267          0.0948          0.212          0.406
2 I          0.696          0.620          0.673          0.725
3 M          0.372          0.222          0.321          0.511
```

Part (e)

```
pairwise.t.test(abalone$Rings, abalone$Sex, p.adjust.method = "bonferroni")
```

Pairwise comparisons using t tests with pooled SD

data: abalone\$Rings and abalone\$Sex

```
      F      I
I < 2e-16 -
M 0.00031 < 2e-16
```

P value adjustment method: bonferroni

Since all pairwise t-test p-values are well below 0.05, we conclude that sex is a significant factor influencing the number of rings.

Question 2

Part (a)

```
food_exp <- read.csv("food_expenditure.csv", header = TRUE)
head(food_exp)
```

```
  ID What.is.your.age.
1  1                68
2  2                88
3  3                82
4  4                73
```

5 5 89

6 6 18

How.many.individuals.live.in.your.household.for.which.you.are.responsible.for.food.expendi

1

2

3

4

5

6

What.state.do.you.live.in.

1 LA

2 WA

3 MS

4 AK

5 IN

6 WI

What.currency.are.you.reporting.your.food.expenditures.in.

1 USD

2 USD

3 USD

4 USD

5 USD

6 EUR

What.was.your.total.food.expenditure.in.the.last.week.

1 436.35

2

3 279.1

4 -20.98

5 494.87

6 276.32

What.was.your.total.food.expenditures.at.grocery.stores.in.the.last.week.

1 168.59

2 452.10

3 301.66

4 139.66

5 NA

6 394.44

What.was.your.food.expenditure.while.dining.out.in.the.last.week.

1 140.71

2 192.94

3 239.84

4 69.19

5 191.72

6		283.20
	What.was.your.food.expenditure..miscellaneous..in.the.last.week.	
1		109.77
2		NA
3		103.94
4		44.84
5		172.31
6		114.06
	How.many.times.did.you.dine.out.last.week.	
1	4	
2	1	
3	9	
4	2	
5	3	
6	6	
	Are.you.including.alcohol.in.your.food.expenditures.	
1	Yes	
2	Unknown	
3	Yes	
4	Unknown	
5	Yes	
6	Unknown	
	What.food.assistance.programs..if.any..did.you.use.for.your.food.expenditures.last.week.	
1		None
2		SNAP
3		None
4		None
5		None
6		Food Pantry

Part (b)

```
names(food_exp) <- c(
  "ID",
  "Age",
  "FamilySize",
  "State",
  "Currency",
  "FoodExp",
  "GroceryExp",
  "DiningExp",
```

```

    "MiscExp",
    "DiningOutCount",
    "AlcoholIncluded",
    "FoodAssistProg"
  )
head(food_exp)

```

	ID	Age	FamilySize	State	Currency	FoodExp	GroceryExp	DiningExp	MiscExp
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

	DiningOutCount	AlcoholIncluded	FoodAssistProg
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

Part (c)

```

n_before <- nrow(food_exp)
n_after <- nrow(subset(food_exp, Currency == "USD"))

cat("Before restricting: ", n_before, "\n")

```

Before restricting: 262

```

cat("After restricting: ", n_after, "\n")

```

After restricting: 230

Part (d)

Records were excluded if the reported age was below 18 or above 100.

```
food_clean <- food_exp %>%  
  filter(Age >= 18, Age <= 100)  
summary(food_clean$Age)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
18.00	37.50	54.00	55.33	75.50	90.00

Part (e)

Records were excluded if the reported state was invalid american state abbreviation.

```
# Create vector containing all valid american states abbreviation  
valid_states <- c(state.abb, "DC")  
  
food_clean <- food_clean %>%  
  filter(State %in% valid_states)  
unique(food_clean$State)
```

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

Part (f)

Records were excluded if any expenditure of the four was below 0 or above 10000.

```
# Convert `FoodExp` column in numeric value  
food_clean$FoodExp <- as.numeric((food_clean$FoodExp))
```

Warning: NAs introduced by coercion


```
food_clean <- food_clean %>%
  filter(FoodExp >= 0, FoodExp <= 10000,
         GroceryExp >= 0, GroceryExp <= 10000,
         DiningExp >= 0, DiningExp <= 10000,
         MiscExp >= 0, MiscExp <= 10000)
summary(food_clean[, c("FoodExp", "GroceryExp", "DiningExp", "MiscExp")])
```

FoodExp	GroceryExp	DiningExp	MiscExp
Min. : 0.0	Min. : 9.45	Min. : 4.33	Min. : 3.50
1st Qu.: 272.7	1st Qu.:143.92	1st Qu.: 60.98	1st Qu.: 41.33
Median : 394.3	Median :220.24	Median :101.38	Median : 81.82
Mean : 408.1	Mean :228.96	Mean :108.94	Mean : 94.83
3rd Qu.: 549.8	3rd Qu.:301.74	3rd Qu.:145.96	3rd Qu.:129.84
Max. :1049.2	Max. :655.63	Max. :335.12	Max. :316.47

Part (g)

Records were excluded if the reported dining out number was below 0 or above 30.

```
food_clean <- food_clean %>%
  filter(DiningOutCount >= 0, DiningOutCount <= 30)
summary(food_clean$DiningOutCount)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.000	3.000	6.000	7.212	9.000	30.000

Part (h)

```
n_after <- nrow(food_clean)
cat("Number of observations after cleaning: ", n_after, "\n")
```

Number of observations after cleaning: 151

Question 3

Part (a)

```
#' Compute the next Collatz number
#' @parameter n: a positive integer
#' @return Integer: the next number in the Collatz sequence
nextCollatz <- function(n)
{
  # Error handling
  if (!is.numeric(n) || length(n) != 1 || n <= 0 || n != as.integer(n))
  {
    stop("Input must be a single positive integer!")
  }

  # Compute next Collatz number
  if (n %% 2 == 0)
    return (n / 2)
  else
    return (3 * n + 1)
}
nextCollatz(5)
```

```
[1] 16
```

```
nextCollatz(16)
```

```
[1] 8
```

Part (b)

```
#' Compute entire Collatz sequence
#' @parameter n: a positive integer
#' @return a 2-element list: [Collatz sequence, sequence length]

collatzSequence <- function(n)
{
  # Error handling
```

```

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

# Generate the Collatz sequence
seq_vals <- n
while (n != 1)
{
  if (n %% 2 == 0)
    n <- n / 2
  else
    n <- 3 * n + 1
  seq_vals <- c(seq_vals, n)
}
return (list(sequence = seq_vals, length = length(seq_vals)))
}

collatzSequence(5)

```

```

$sequence
[1] 5 16 8 4 2 1

```

```

$length
[1] 6

```

```
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

```

Part (c)

```

shortest_val <- Inf
shortest_seq_len <- Inf
longest_val <- Inf

```

```

longest_seq_len <- 0
for (i in 100:500)
{
  # Compute current Collatz sequence length
  current_seq_len <- collatzSequence(i)$length

  if (current_seq_len < shortest_seq_len) # Update shortest sequence length
  {
    shortest_val <- i
    shortest_seq_len <- current_seq_len
  }
  else if (current_seq_len > longest_seq_len) # Update longest sequence length
  {
    longest_val <- i
    longest_seq_len <- current_seq_len
  }
}
# Print final result
cat("The shortest Collatz sequence starts with: ", shortest_val, "\n")

```

The shortest Collatz sequence starts with: 128

```

cat("The longest Collatz sequence starts with: ", longest_val, "\n")

```

The longest Collatz sequence starts with: 327