Problem_Saet_2

Yifan Li

Link to the GitHub

The link to my GitHub repository is https://github.com/FYlee39/Stats-506/tree/main/PS2.

Problem 1

a.

Version one:

```r
#' Using for loop to implement the game
#'
#' @param n number of dice to roll
#' @return win total winnings
play_dice_v1 <- function(n){
  # cost 2 to play a roll
  win <- -2 * n
  roll_results <- sample(1: 6, n, replace=TRUE)
  for (single in roll_results){
    # if the dice shows 3 or 5, win the double points
    if (single == 3 | single == 5){
      win <- win + 2 * single
    }
  }
  return(win)
}

play_dice_v1(10)
```

```
[1] -4
```

Version two:

```r
#' Using built-in R vectorized functions to implement the game
#'
#' @param n number of dice to roll
#' @return win total winnings
play_dice_v2 <- function(n){
  # cost 2 to play a roll
  win <- -2 * n
  roll_results <- sample(1: 6, n, replace=TRUE)
  # For 3 and 5, let the winning be the double of itself
  win_index <- (roll_results == 3) | (roll_results == 5)
  roll_results[win_index] <- 2 * roll_results[win_index]
  # For other numbers, set them to be 0
  roll_results[!win_index] <- 0
  win <- win + sum(roll_results)
  return(win)
}

play_dice_v2(10)
```

```
[1] 20
```

Version three:

```r
#' Using table to implement the game
#'
#' @param n number of dice to roll
#' @return win total winnings
play_dice_v3 <- function(n){
  # cost 2 to play a roll
  win <- -2 * n
  roll_results <- table(sample(1: 6, n, replace=TRUE))
  # number of 3 been rolled
  num_three <- ifelse(!is.na(roll_results['3']), roll_results['3'], 0)
  # number of 5 been rolled
  num_five <- ifelse(!is.na(roll_results['5']), roll_results['5'], 0)
  # total wining points
  winning <- 3 * 2 * num_three + 5 * 2 * num_five
```

```
  win <- win + winning[[1]]
  return(win)
}

play_dice_v3(10)
```

```
[1] -10
```

Version four:

```
#' Using lapply to implement the game
#'
#' @param n number of dice to roll
#' @return win total winnings
play_dice_v4 <- function(n){
  # cost 2 to play a roll
  win <- -2 * n
  roll_results <- sample(1: 6, n, replace=TRUE)

  #' Get winning point of given roll
  #'
  #' @param x one roll result
  #' @return point the point gain from this rolling
  get_points <- function(x){
    point = 0
    # if the dice shows 3 or 5
    if (x == 3 | x == 5){
      # double the number to be the winning points
      point = x * 2
    }
    return(point)
  }

  winning <- sum(sapply(roll_results, get_points))
  win <- win + winning
  return(win)
}

play_dice_v4(10)
```

```
[1] -2
```

b.

Test for version one:

```r
play_dice_v1(3)
```

```
[1] 0
```

```r
play_dice_v1(3000)
```

```
[1] 1604
```

Test for version two:

```r
play_dice_v2(3)
```

```
[1] 0
```

```r
play_dice_v2(3000)
```

```
[1] 1954
```

Test for version three:

```r
play_dice_v3(3)
```

```
[1] 0
```

```r
play_dice_v3(3000)
```

```
[1] 2062
```

Test for version four:

```r
play_dice_v4(3)
```

```
[1] 10
```

```r
play_dice_v4(3000)
```

```
[1] 1532
```

c.

To demonstrate the same result, one needs set the same seed before each sampling. For 3 times of experiments:

```r
set.seed(09152024)
play_dice_v1(3)
```

```
[1] 4
```

```r
set.seed(09152024)
play_dice_v2(3)
```

```
[1] 4
```

```r
set.seed(09152024)
play_dice_v3(3)
```

```
[1] 4
```

```r
set.seed(09152024)
play_dice_v4(3)
```

```
[1] 4
```

For 3000 times of experiments:

```r
set.seed(09152024)
play_dice_v1(3000)
```

```
[1] 2344
```

```r
set.seed(09152024)
play_dice_v2(3000)
```

```
[1] 2344
```

```r
set.seed(09152024)
play_dice_v3(3000)
```

```
[1] 2344
```

```r
set.seed(09152024)
play_dice_v4(3000)
```

```
[1] 2344
```

d.

For low input (1,000):

```r
library(microbenchmark)
microbenchmark(v1=play_dice_v1(1000), v2=play_dice_v2(1000),
               v3=play_dice_v3(1000), v4=play_dice_v4(1000))
```

```
Unit: microseconds
 expr    min      lq     mean  median      uq    max neval  cld
   v1  274.3  281.85  315.472  286.40  297.65 2636.7   100 a
   v2  125.1  134.10  145.512  145.05  154.80  193.0   100  b
   v3  351.5  381.45  400.431  400.55  416.05  484.2   100   c
   v4 1259.2 1278.05 1343.738 1293.00 1344.20 3672.1   100    d
```

For large input (100,000):

```r
library(microbenchmark)
microbenchmark(v1=play_dice_v1(100000), v2=play_dice_v2(100000),
               v3=play_dice_v3(100000), v4=play_dice_v4(100000))
```

```
Unit: milliseconds
 expr      min        lq      mean    median       uq      max neval cld
   v1  26.6637  30.90800  37.84597  33.99315  41.5251 166.5723   100 a
   v2  10.8280  11.16655  11.94471  11.74655  12.1059  25.9205   100  b
   v3  13.0132  13.47210  16.40812  13.93550  14.9918 154.6938   100  b
   v4 132.2448 158.80760 177.16338 174.68360 187.4713 316.2325   100   c
```

From two experiments, one can find that among these four function, the implementation using built-in R vectorized functions is the fastest. Mean while, the function using `sapply` is the slowest.

e.

This game is unfair, to defend the decision using a Monte Carlo simulation, the version two will be used. There will be $100,000$ times of expeirments. Then the sample mean will be calculated, if the sample mean is no way near 0, then one can argue that this game is unfair.

```r
sum <- 0
# Do 100,000 times of expeiments, find the sample mean
n <- 100000
for (i in 1: n){
  sum <- sum + play_dice_v2(1)
}
sample_mean <- sum / n
sample_mean
```

```
[1] 0.6635
```

Since the sample mean is much greater than zero, one can argue that this is not a fair game.

Problem 2

a

```r
raw_data <- read.csv('cars.csv', header=TRUE, sep=',',
                     col.names=c('Height',
                                 'Length',
                                 'Width',
```

```r
                                              'Driveline',
                                              'Type',
                                              'Hybird',
                                              'Gears',
                                              'Transmission',
                                              'City_mpg',
                                              'Fuel_type',
                                              'Highway_mpg',
                                              'Classification',
                                              'ID',
                                              'Make',
                                              'Model_year',
                                              'Year',
                                              'horsepower',
                                              'Torque'))
head(raw_data)
```

```
  Height Length Width         Driveline
1    140    143   202   All-wheel drive
2    140    143   202 Front-wheel drive
3    140    143   202 Front-wheel drive
4    140    143   202   All-wheel drive
5    140    143   202   All-wheel drive
6     91     17    62   All-wheel drive
                                          Type Hybird Gears
1         Audi 3.2L 6 cylinder 250hp 236ft-lbs   True     6
2 Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo   True     6
3 Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo   True     6
4 Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo   True     6
5 Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo   True     6
6        Audi 3.2L 6 cylinder 265hp 243 ft-lbs   True     6
                    Transmission City_mpg Fuel_type Highway_mpg
1 6 Speed Automatic Select Shift       18  Gasoline          25
2 6 Speed Automatic Select Shift       22  Gasoline          28
3                 6 Speed Manual       21  Gasoline          30
4 6 Speed Automatic Select Shift       21  Gasoline          28
5 6 Speed Automatic Select Shift       21  Gasoline          28
6                 6 Speed Manual       16  Gasoline          27
          Classification                         ID Make   Model_year Year
1 Automatic transmission         2009 Audi A3 3.2 Audi 2009 Audi A3 2009
2 Automatic transmission      2009 Audi A3 2.0 T AT Audi 2009 Audi A3 2009
3    Manual transmission         2009 Audi A3 2.0 T Audi 2009 Audi A3 2009
```

```
4 Automatic transmission 2009 Audi A3 2.0 T Quattro Audi 2009 Audi A3 2009
5 Automatic transmission 2009 Audi A3 2.0 T Quattro Audi 2009 Audi A3 2009
6    Manual transmission           2009 Audi A5 3.2 Audi 2009 Audi A5 2009
  horsepower Torque
1        250    236
2        200    207
3        200    207
4        200    207
5        200    207
6        265    243
```

b.

```
gasoline_data <- raw_data[raw_data['Fuel_type'] == 'Gasoline', ]
head(gasoline_data)
```

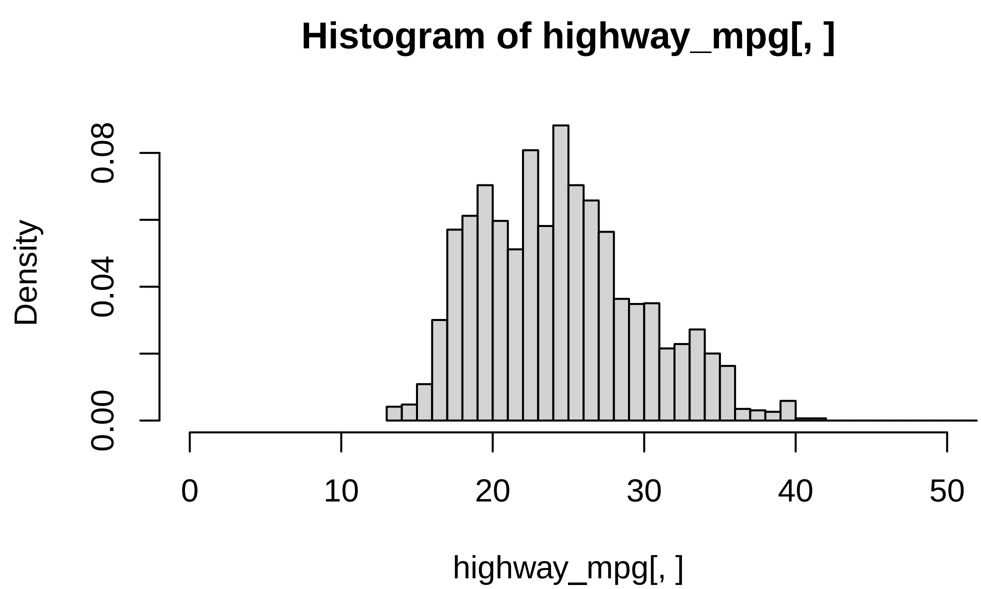
```
  Height Length Width         Driveline
1    140    143   202   All-wheel drive
2    140    143   202 Front-wheel drive
3    140    143   202 Front-wheel drive
4    140    143   202   All-wheel drive
5    140    143   202   All-wheel drive
6     91     17    62   All-wheel drive
                                          Type Hybird Gears
1         Audi 3.2L 6 cylinder 250hp 236ft-lbs   True     6
2 Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo   True     6
3 Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo   True     6
4 Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo   True     6
5 Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo   True     6
6        Audi 3.2L 6 cylinder 265hp 243 ft-lbs   True     6
                  Transmission City_mpg Fuel_type Highway_mpg
1 6 Speed Automatic Select Shift       18  Gasoline          25
2 6 Speed Automatic Select Shift       22  Gasoline          28
3                 6 Speed Manual       21  Gasoline          30
4 6 Speed Automatic Select Shift       21  Gasoline          28
5 6 Speed Automatic Select Shift       21  Gasoline          28
6                 6 Speed Manual       16  Gasoline          27
          Classification                            ID Make   Model_year Year
1 Automatic transmission           2009 Audi A3 3.2 Audi 2009 Audi A3 2009
2 Automatic transmission      2009 Audi A3 2.0 T AT Audi 2009 Audi A3 2009
```

```
3    Manual transmission         2009 Audi A3 2.0 T Audi 2009 Audi A3 2009
4 Automatic transmission 2009 Audi A3 2.0 T Quattro Audi 2009 Audi A3 2009
5 Automatic transmission 2009 Audi A3 2.0 T Quattro Audi 2009 Audi A3 2009
6    Manual transmission           2009 Audi A5 3.2 Audi 2009 Audi A5 2009
  horsepower Torque
1        250    236
2        200    207
3        200    207
4        200    207
5        200    207
6        265    243
```

c.

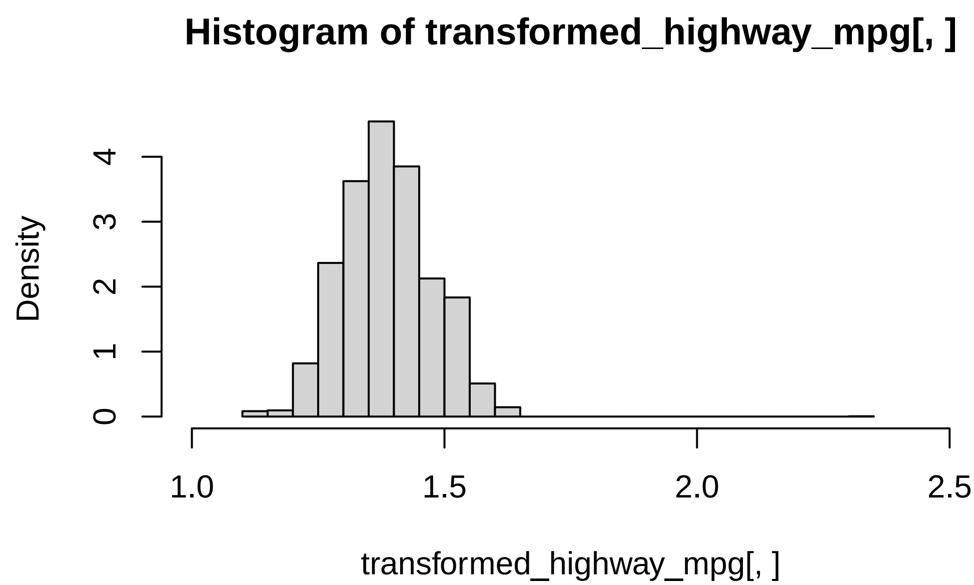
The original data distribution is:

```r
highway_mpg <- gasoline_data['Highway_mpg']
hist(highway_mpg[,], breaks = 200, probability = TRUE, xlim = c(0, 50))
```



Since the data are all positive and the distribution has a right skew with a long tail, a log transformation would likely be the best choice. Then update the data in the data frame.

```r
transformed_highway_mpg <- log10(highway_mpg)
hist(transformed_highway_mpg[,], breaks = 20, probability = TRUE, xlim = c(1, 2.5))
```




```r
gasoline_data['Highway_mpg'] <- transformed_highway_mpg
```

d.

```r
# Make Year to be categorical variable
gasoline_data$Year <- as.factor(gasoline_data$Year)
d_lm <- lm(Highway_mpg~Torque + horsepower
           + Height + Length + Width
           + Year, data=gasoline_data)
summary(d_lm)
```

```
Call:
lm(formula = Highway_mpg ~ Torque + horsepower + Height + Length +
    Width + Year, data = gasoline_data)

Residuals:
```

```
     Min       1Q   Median       3Q      Max
-0.23782 -0.04076 -0.00180  0.04297  1.05035

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.523e+00  9.625e-03 158.236  < 2e-16 ***
Torque      -9.964e-04  2.934e-05 -33.956  < 2e-16 ***
horsepower   4.012e-04  3.033e-05  13.227  < 2e-16 ***
Height       1.759e-04  1.501e-05  11.719  < 2e-16 ***
Length       1.509e-05  1.177e-05   1.282  0.19980
Width       -3.788e-05  1.205e-05  -3.144  0.00168 **
Year2010    -9.473e-03  9.015e-03  -1.051  0.29342
Year2011    -1.055e-03  9.000e-03  -0.117  0.90665
Year2012     1.742e-02  9.071e-03   1.921  0.05485 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0613 on 4582 degrees of freedom
Multiple R-squared:  0.5638,    Adjusted R-squared:  0.563
F-statistic: 740.3 on 8 and 4582 DF,  p-value: < 2.2e-16
```

From the coefficient of torque, which is -9.964155×10^{-4}, meaning for each additional unit of torque, highway MPG will decrease by -9.964155×10^{-4} while holding other variables constant. The coefficient is significant with a p-value less than 2×10^{-16}, indicating that the relationship is statistically significant.

e.

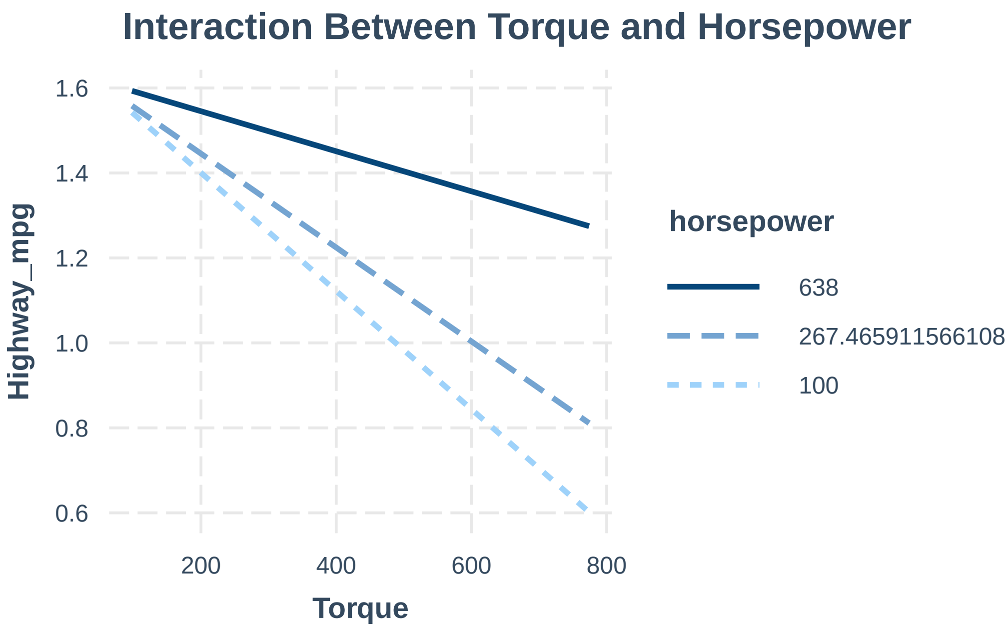
```r
e_lm <- lm(Highway_mpg ~ horsepower * Torque +
           + Height + Length + Width, data=gasoline_data)
summary(e_lm)
```

```
Call:
lm(formula = Highway_mpg ~ horsepower * Torque + +Height + Length +
    Width, data = gasoline_data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.23084 -0.03530 -0.00158  0.03424  1.06415
```

```
Coefficients:
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)        1.673e+00  6.746e-03 247.952  < 2e-16 ***
horsepower        -7.294e-05  3.359e-05  -2.172   0.0299 *  
Torque            -1.562e-03  3.352e-05 -46.618  < 2e-16 ***
Height             1.273e-04  1.414e-05   9.002  < 2e-16 ***
Length             1.042e-05  1.102e-05   0.946   0.3443    
Width             -4.992e-05  1.129e-05  -4.422 9.98e-06 ***
horsepower:Torque  1.710e-06  6.139e-08  27.864  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05747 on 4584 degrees of freedom
Multiple R-squared:  0.6165,	Adjusted R-squared:  0.616 
F-statistic:  1228 on 6 and 4584 DF,  p-value: < 2.2e-16
```

For the three different horsepower values, they will be the min, mean and the max value.

```r
library(interactions)

horsepower_range <- c(min(gasoline_data$horsepower),
                              mean(gasoline_data$horsepower),
                              max(gasoline_data$horsepower))

interact_plot(e_lm, pred=Torque, modx=horsepower, data=gasoline_data,
              modx.values=horsepower_range,
              main.title="Interaction Between Torque and Horsepower")
```



f.

```
# create the design matrix
design_matrix <-  model.matrix(Highway_mpg ~ Torque + horsepower
           + Height + Length + Width
           + Year, data=gasoline_data)

Y <- gasoline_data$Highway_mpg

design_matrix_t <- t(design_matrix)

XTX_inv <- solve(design_matrix_t %*% design_matrix)

XTY <- design_matrix_t %*% Y

beta_hat <- XTX_inv %*% XTY
# Compare two values
beta_hat
```

```
                     [,1]
(Intercept)  1.523037e+00
```

```
Torque      -9.964155e-04
horsepower   4.012067e-04
Height       1.758848e-04
Length       1.509263e-05
Width       -3.788045e-05
Year2010    -9.473036e-03
Year2011    -1.055492e-03
Year2012     1.742184e-02
```

```r
manual_coef <- setNames(as.vector(beta_hat), names(d_lm$coefficients))
all.equal(manual_coef, d_lm$coefficients)
```

```
[1] TRUE
```

The result is True, which shows that the manual result is the same with the `lm` result.