# Homework02

Yuchen Shao

Link to GitHub repository

https://github.com/LAREINA-SHAO/STATS-506.git

## Problem 1 - Dice Game

### a. Version 1

```
#' Title Dice Game Function - Using a loop
#'
#' @param n The number of dice to roll
#'
#' @return  Total winnings
play_dice1 <- function(n){
  if (n < 0) {
    stop("Please input a non-negative number.")
  }

  # if no dice are rolled, return 0
  if (n == 0){
    return(0)
  }

  total_winnings <- 0
  # roll the dice n times
  die <- sample(1:6, n, replace = TRUE)

  for (i in 1:n) {
    # it costs $2 per game
    total_winnings <- total_winnings - 2
```

```r
    # on a roll of 3 or 5, win twice the roll
    if (die[i] == 3 | die[i] == 5) {
      total_winnings <- total_winnings + 2 * die[i]
    }
  }
  return(total_winnings)
}
```

## a. Version 2

```r
#' Title Dice Game Function - using built-in R vectorized functions
#'
#' @param n The number of dice to roll
#'
#' @return Total winnings
play_dice2 <- function(n){
  if (n < 0) {
    stop("Please input a non-negative number.")
  }

  # if no dice are rolled, return 0
  if (n == 0){
    return(0)
  }

  # roll the dice n times
  die <- sample(1:6, n, replace = TRUE)

  #winnings for rolls of 3 or 5
  winnings <- ifelse(die == 3 | die == 5, 2 * die, 0)
  total_winnings <- + sum(winnings) - 2 * n
  return(total_winnings)
}
```

## a. Version 3

```r
#' Title Dice Game Function - using table()
#'
```

```r
#' @param n The number of dice to roll
#'
#' @return Total winnings
play_dice3 <- function(n){
  if (n < 0) {
    stop("Please input a non-negative number.")
  }

  # if no dice are rolled, return 0
  if (n == 0){
    return(0)
  }

  # roll the dice n times
  die <- sample(1:6, n, replace = TRUE)

  # create a frequency table (include 0 counts)
  die_table <- table(factor(die, levels = 1:6))

  # add together winnings and subtract out the total cost
  winnings_3 <- die_table["3"] * 2 * 3
  winnings_5 <- die_table["5"] * 2 * 5
  total_winnings <- winnings_3 + winnings_5 - 2 * n
  names(total_winnings) <- NULL
  return(total_winnings)
}
```

### a. Version 4

```r
#' Title Dice Game Function - using sapply
#'
#' @param n The number of dice to roll
#'
#' @return Total winnings
play_dice4 <- function(n){
  if (n < 0) {
    stop("Please input a non-negative number.")
  }

  # if no dice are rolled, return 0
```

```r
  if (n == 0){
    return(0)
  }

  # roll the dice n times
  die <- sample(1:6, n, replace = TRUE)

  # apply the game rules using sapply
  winnings <- sapply(die, function(x) {
    if (x == 3 || x == 5) {
      return(2 * x)
    } else {
      return(0)
    }
  })
  total_winnings <- sum(winnings) - 2 * n
  return(total_winnings)
}
```

**b.**

```r
# total winnings for Version 1 with inputs of 3 and 3,000 dice rolls
c(play_dice1(3), play_dice1(3000))
```

```
[1]    0 2142
```

```r
# total winnings for Version 2 with inputs of 3 and 3,000 dice rolls
c(play_dice2(3), play_dice2(3000))
```

```
[1]   -6 2208
```

```r
# total winnings for Version 3 with inputs of 3 and 3,000 dice rolls
c(play_dice3(3), play_dice3(3000))
```

```
[1]   10 1992
```

```r
# total winnings for Version 4 with inputs of 3 and 3,000 dice rolls
c(play_dice4(3), play_dice4(3000))
```

```
[1]   20 2372
```

**c.**

```r
# inputs 3
# set the seed for reproducibility
set.seed(111)
result1 <- play_dice1(3)

# reset the seed to ensure same random sequence
set.seed(111)
result2 <- play_dice2(3)

set.seed(111)
result3 <- play_dice3(3)

set.seed(111)
result4 <- play_dice4(3)

c(result1, result2, result3, result4)
```

```
[1] 0 0 0 0
```

```r
# inputs 3000
set.seed(111)
result1 <- play_dice1(3000)

set.seed(111)
result2 <- play_dice2(3000)

set.seed(111)
result3 <- play_dice3(3000)

set.seed(111)
result4 <- play_dice4(3000)

c(result1, result2, result3, result4)
```

```
[1] 1918 1918 1918 1918
```

**d.**

```r
#install.packages("microbenchmark")
library(microbenchmark)
```

```r
#' Title Benchmarks for Dice Game Implementations
#'
#' @param n The number of dice rolls
#'
#' @return A `microbenchmark` object showing the performance of each version
BenchMarks <- function(n) {
  microbenchmark(
    play_dice1 = play_dice1(n),
    play_dice2 = play_dice2(n),
    play_dice3 = play_dice3(n),
    play_dice4 = play_dice4(n),
    times = 10
  )
}

set.seed(111)

print(BenchMarks(1000))
```

```
Unit: microseconds
       expr     min      lq     mean   median      uq     max neval
 play_dice1 267.383 270.512 273.3922 272.8575 275.514 283.957    10
 play_dice2 106.325 118.091 148.4071 137.3160 163.523 256.714    10
 play_dice3 199.137 210.658 238.6254 238.2575 266.285 276.775    10
 play_dice4 720.776 729.446 797.7464 760.5480 820.630 977.404    10
```

```r
print(BenchMarks(100000))
```

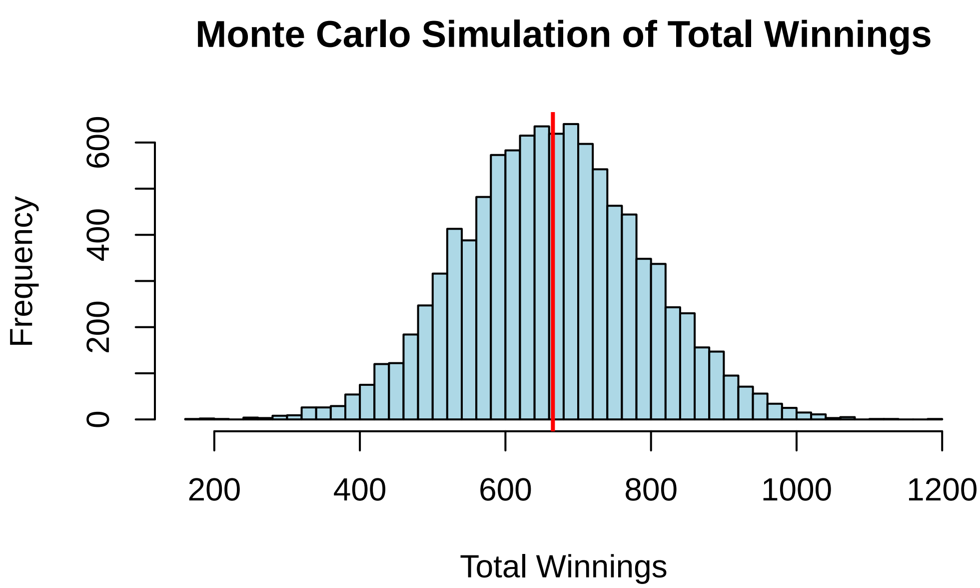
```
Unit: milliseconds
       expr       min        lq      mean    median       uq      max neval
 play_dice1 24.019454 26.136917 29.299269 28.420536 31.51776 39.91268    10
 play_dice2  7.317755  7.956419  9.446681  9.725216 10.47302 11.81915    10
```

```
play_dice3  9.280632 10.741924  11.230799 11.306169  12.27452  12.44183    10
play_dice4 75.367440 89.587469 105.412463 97.680122 117.30530 171.67006    10
```

The function implemented using R's built-in vectorized functions is the fastest, while the function using sapply() performs the slowest. The function using a loop performs better than the one using table() when handling large inputs, but this advantage is negligible with smaller input sizes.

### e.

```r
#simulate 10000 times and pre-allocate a vector to store results
results <- numeric(10000)

for (i in 1:10000) {
  # play 1000 dice rolls in each simulation
  results[i] <- play_dice2(1000)
}

# Plot a histogram of the total winnings
hist(results, breaks = 50, main = "Monte Carlo Simulation of Total Winnings",
     xlab = "Total Winnings", col = "lightblue", border = "black")
abline(v = mean(results), col = "red", lwd = 2)
```



```
cat("Mean of total winnings:", mean(results), "\n")
```

```
Mean of total winnings: 665.2222
```

This is not a fair game. The mean of total winnings across 10,000 simulations is significantly above zero, indicating that the player, on average, makes a profit.

## Problem 2 - Linear Regression

### a.

```
cars <- read.csv("cars.csv")
names(cars)
```

```
 [1] "Dimensions.Height"
 [2] "Dimensions.Length"
 [3] "Dimensions.Width"
 [4] "Engine.Information.Driveline"
 [5] "Engine.Information.Engine.Type"
 [6] "Engine.Information.Hybrid"
 [7] "Engine.Information.Number.of.Forward.Gears"
 [8] "Engine.Information.Transmission"
 [9] "Fuel.Information.City.mpg"
[10] "Fuel.Information.Fuel.Type"
[11] "Fuel.Information.Highway.mpg"
[12] "Identification.Classification"
[13] "Identification.ID"
[14] "Identification.Make"
[15] "Identification.Model.Year"
[16] "Identification.Year"
[17] "Engine.Information.Engine.Statistics.Horsepower"
[18] "Engine.Information.Engine.Statistics.Torque"
```

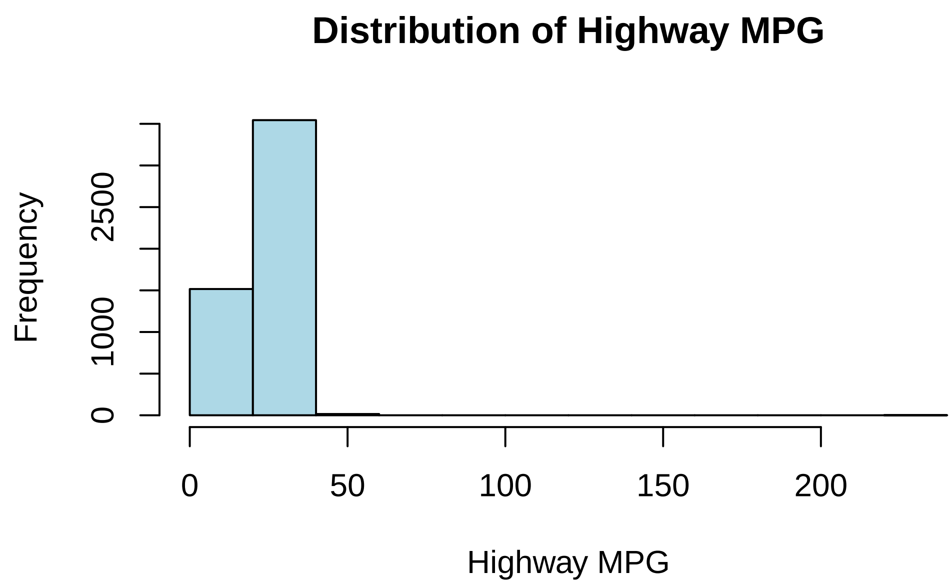
```
# rename the columns
names(cars) <- c("height", "length", "width", "driveline", "engine_type",
                 "hybrid","forward_gears", "transmission", "city_mpg",
                 "fuel_type","highway_mpg", "classification", "ID", "make",
                 "model_year", "year", "horsepower", "torque")
```

**b.**

```r
gasoline_cars <- cars[cars$fuel == "Gasoline", ]
```

**c.**

```r
# the distribution of highway_mpg
hist(cars$highway_mpg, main = "Distribution of Highway MPG",
     xlab = "Highway MPG", col = "lightblue", border = "black")
```

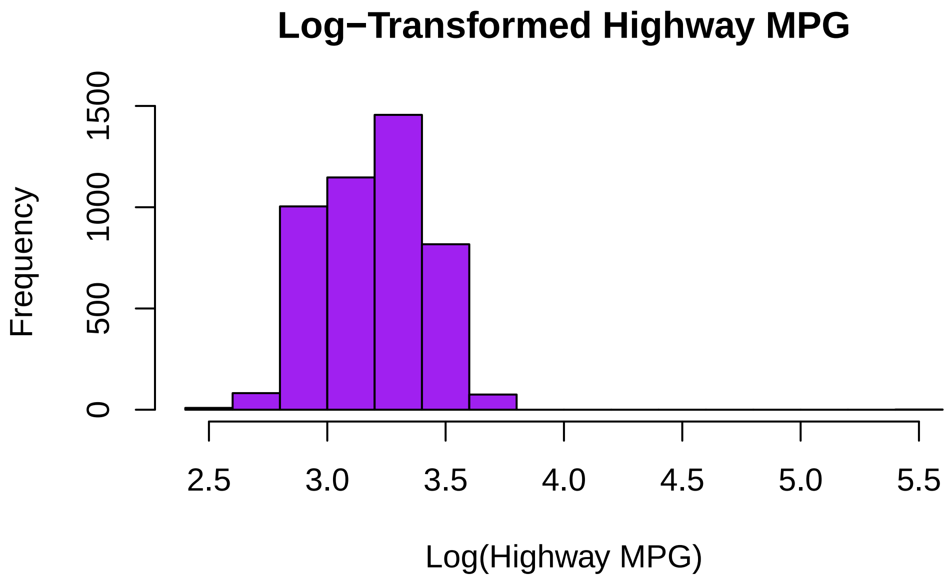



```r
summary(gasoline_cars$highway_mpg)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  13.00   21.00   25.00   24.97   28.00  223.00 
```

```r
# log transformation
gasoline_cars$log_highway_mpg <- log(gasoline_cars$highway_mpg)
```

```
# the distribution of log_highway_mpg
hist(gasoline_cars$log_highway_mpg, main = "Log-Transformed Highway MPG",
     xlab = "Log(Highway MPG)", col = "purple", border = "black")
```




```
summary(gasoline_cars$log_highway_mpg)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  2.565   3.045   3.219   3.194   3.332   5.407 
```

The distribution of highway_mpg is highly right-skewed, suggesting the presence of outliers and a long tail of higher values. To normalize the distribution, I applied a logarithmic transformation, which helped reduce skewness and made the data more suitable for analysis.

##d.

```
# convert 'year' to a factor (categorical variable)
gasoline_cars$year <- as.factor(gasoline_cars$year)
# fit a linear regression model
model <- lm(log_highway_mpg ~ torque + horsepower + height + length + width +
              year, data = gasoline_cars)
summary(model)
```

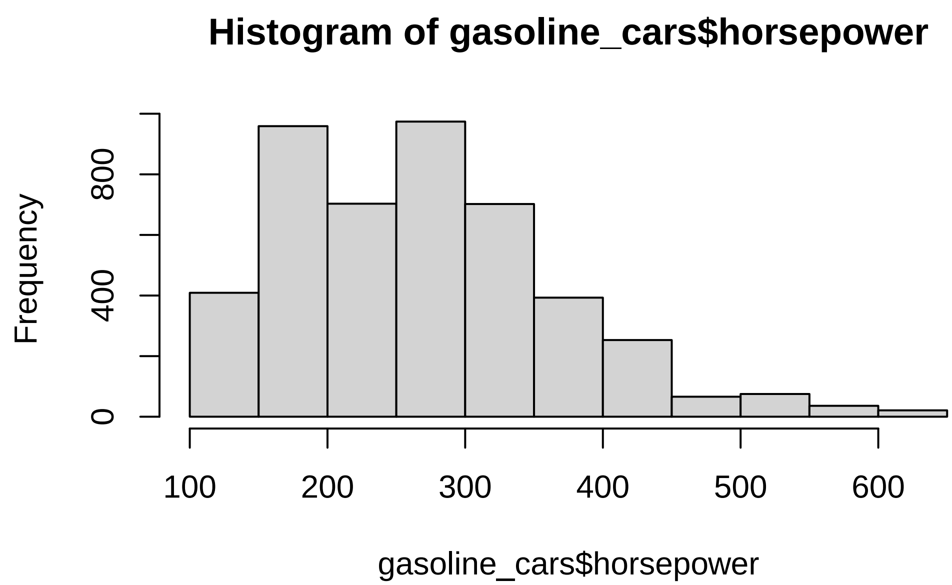
```
Call:
lm(formula = log_highway_mpg ~ torque + horsepower + height +
    length + width + year, data = gasoline_cars)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.54759 -0.09385 -0.00414  0.09894  2.41852 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  3.507e+00  2.216e-02 158.236  < 2e-16 ***
torque      -2.294e-03  6.757e-05 -33.956  < 2e-16 ***
horsepower   9.238e-04  6.984e-05  13.227  < 2e-16 ***
height       4.050e-04  3.456e-05  11.719  < 2e-16 ***
length       3.475e-05  2.710e-05   1.282  0.19980    
width       -8.722e-05  2.774e-05  -3.144  0.00168 ** 
year2010    -2.181e-02  2.076e-02  -1.051  0.29342    
year2011    -2.430e-03  2.072e-02  -0.117  0.90665    
year2012     4.012e-02  2.089e-02   1.921  0.05485 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1412 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
```

The estimated coefficient for torque is approximately -0.00229, meaning that for each additional unit of torque, the logarithm of highway MPG decreases by about 0.00229, holding all other variables constant. This indicates an inverse relationship between torque and highway fuel efficiency, where higher torque is associated with lower highway MPG. Additionally, this coefficient is highly significant, with a p-value less than 2e-16, confirming the robustness of the relationship.

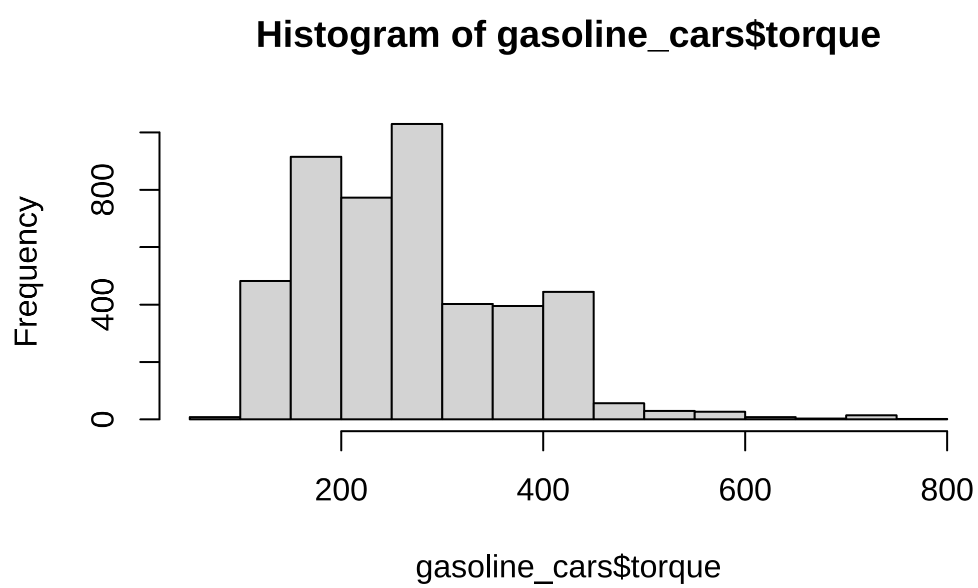
### e.

```r
#install.packages("interactions")
library(interactions)
```

```r
hist(gasoline_cars$horsepower)
```

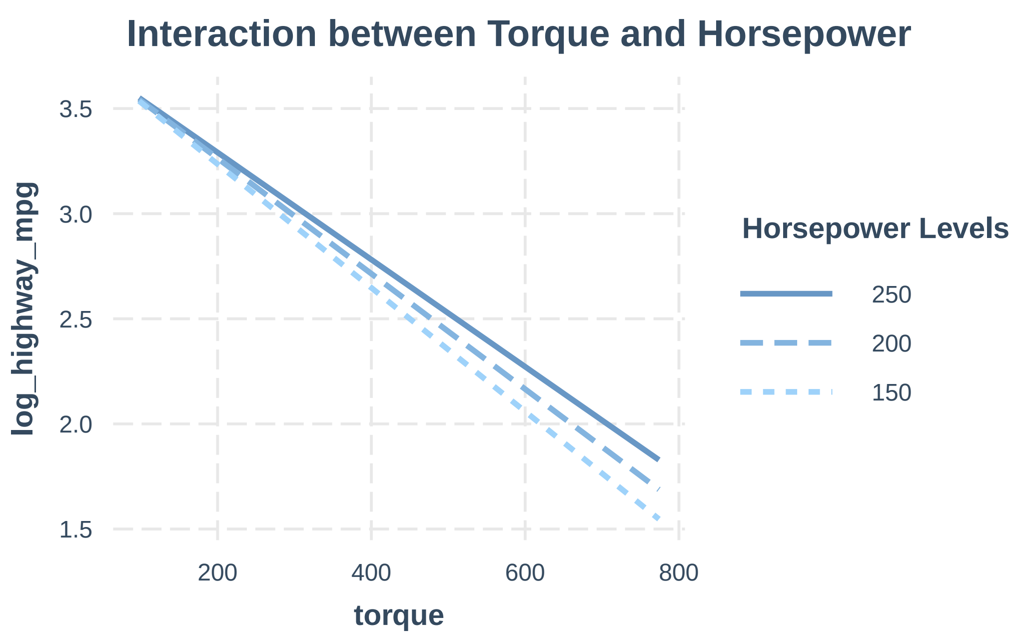



```r
hist(gasoline_cars$torque)
```



```r
model_interaction  <- lm(log_highway_mpg ~ horsepower*torque + height + length +
                           width + as.factor(year), data = gasoline_cars)

interact_plot(model_interaction, pred = "torque", modx = "horsepower",
                 # three different reasonable values of horsepower
                 modx.values = c(150, 200, 250),
                 at = list(year = 2010),
                 main.title = "Interaction between Torque and Horsepower",
                 xlab = "Torque",
                 ylab = "Log Highway MPG",
                 legend.main = "Horsepower Levels")
```

```
Using data gasoline_cars from global environment. This could cause
incorrect results if gasoline_cars has been altered since the model was
fit. You can manually provide the data to the "data =" argument.
```




The plot shows a negative relationship between torque and the logarithm of highway MPG, where increasing torque leads to a decrease in log_highway_mpg. The effect of torque on MPG is more pronounced for lower horsepower values, as shown by the steeper slopes for lower horsepower levels (150 and 200) compared to higher horsepower (250).

### f.

```r
# design matrix X for the linear regression
X <- model.matrix(log_highway_mpg ~ torque + horsepower + height + length +
                    width + as.factor(year), data = gasoline_cars)
y <- gasoline_cars$log_highway_mpg


# manually compute the OLS regression coefficients using matrix operations
# beta_hat = (X^T X)^(-1) X^T y
XtX_inv <- solve(t(X) %*% X)
XtY <- t(X) %*% y
beta_hat <- XtX_inv %*% XtY

cbind(model$coef, beta_hat)
```

```
                     [,1]          [,2]
(Intercept)  3.506922e+00  3.506922e+00
torque      -2.294331e-03 -2.294331e-03
horsepower   9.238126e-04  9.238126e-04
height       4.049897e-04  4.049897e-04
length       3.475207e-05  3.475207e-05
width       -8.722295e-05 -8.722295e-05
year2010    -2.181247e-02 -2.181247e-02
year2011    -2.430359e-03 -2.430359e-03
year2012     4.011528e-02  4.011528e-02
```

The beta_hat calculated manually is the same as lm did prior.