STAT 506 HW 2

ZHAO Shengchun

Github URL:

https://github.com/ZHAOShengchun67734538/STAT-506-HW-2

Question 1

(a)

```
# Version 1
#' Dice game by using loop
#' @param value (which is the number of dice to roll)
#' @return Total winnings
#' @examples we randomly roll a dice 4 times
#' the number is {3,4,6,5}, the winning should be
#' 2x3-2-2-2+2x5-2 = 8, so the function will return 8
game.loop = function(value)
{
  # First we need to check whether the input is valid.
  # we must assure that the input is a positive integer.
  if(!is.numeric(value))
  {
    warning("Input must be a positive integer.")
    stop("This input is not a numeric, please try again.")
  }
  if(abs(value) != value)
  {
    warning("Input must be a positive integer.")
    stop("This input is not a positive number, please try again.")
  }
  if(value != as.integer(value))
  {
```

```
    warning("Input must be a positive integer.")
    stop("This input is not an integer, please try again.")
  }
  # Main Game Part
  roll = sample(1:6, size = value, replace = TRUE)
  winning = 0
  for(i in 1:value)
  {
    if(roll[i] == 3)
    {
      #if the roll is 3,
      #It is winning 2*3-2=4
      winning = winning + 4
    }else if(roll[i] == 5)
    {
      #If the roll is 5,
      #It is winning 2*5-2=8
      winning = winning + 8
    }else{
      winning = winning - 2
    }
  }
  return(winning)
}
```

```
# Version 2
#' Dice game by using vectorized function
#' @param value (which is the number of dice to roll)
#' @return Total winnings
#' @examples we randomly roll a dice 4 times
#' the number is {3,4,6,5}, the winning should be
#' 2x3-2-2-2+2x5-2 = 8, so the function will return 8
game.vectorized = function(value)
{
  # First we need to check whether the input is valid.
  # we must assure that the input is a positive integer.
  if(!is.numeric(value))
  {
    warning("Input must be a positive integer.")
    stop("This input is not a numeric, please try again.")
  }
```

```r
  if(abs(value) != value)
  {
    warning("Input must be a positive integer.")
    stop("This input is not a positive number, please try again.")
  }
  if(value != as.integer(value))
  {
    warning("Input must be a positive integer.")
    stop("This input is not an integer, please try again.")
  }
  # Main Game Part
  roll = sample(1:6, size = value, replace = TRUE)
  winning = ifelse(roll == 3 | roll == 5, (2*roll-2), -2)
  return(sum(winning))
}
```

```r
# Version 3
#' Dice game by using table() in r
#' @param value (which is the number of dice to roll)
#' @return Total winnings
#' @examples we randomly roll a dice 4 times
#' the number is {3,4,6,5}, the winning should be
#' 2x3-2-2-2+2x5-2 = 8, so the function will return 8
game.table = function(value)
{
  # First we need to check whether the input is valid.
  # we must assure that the input is a positive integer.
  if(!is.numeric(value))
  {
    warning("Input must be a positive integer.")
    stop("This input is not a numeric, please try again.")
  }
  if(abs(value) != value)
  {
    warning("Input must be a positive integer.")
    stop("This input is not a positive number, please try again.")
  }
  if(value != as.integer(value))
  {
    warning("Input must be a positive integer.")
    stop("This input is not an integer, please try again.")
```

```
  }
  # Main Game Part
  roll = sample(1:6, size = value, replace = TRUE)
  # The table can be transfer to a data frame,
  # The first column is roll (the dice number appeared)
  # The second column is the frequency of dice number appeared.
  new.roll = as.data.frame(table(roll))
  new.roll$roll = as.numeric(as.character(new.roll$roll))
  new.roll$Freq = as.numeric(as.character(new.roll$Freq))
  # We count the number of dice number is 3
  win3 = sum(new.roll[which(new.roll$roll==3),2])
  # We count the number of dice number is 5
  win5 = sum(new.roll[which(new.roll$roll==5),2])
  fail = sum(new.roll$Freq)-win3-win5
  winning = 4*win3+8*win5-2*fail
  return(winning)
}
```

```
# Version 4
#' Dice game by using vapply() in r
#' @param value (which is the number of dice to roll)
#' @return Total winnings
#' @examples we randomly roll a dice 4 times
#' the number is {3,4,6,5}, the winning should be
#' 2x3-2-2-2+2x5-2 = 8, so the function will return 8
game.vapply = function(value)
{
  # First we need to check whether the input is valid.
  # we must assure that the input is a positive integer.
  if(!is.numeric(value))
  {
    warning("Input must be a positive integer.")
    stop("This input is not a numeric, please try again.")
  }
  if(abs(value) != value)
  {
    warning("Input must be a positive integer.")
    stop("This input is not a positive number, please try again.")
  }
  if(value != as.integer(value))
  {
```

```
    warning("Input must be a positive integer.")
    stop("This input is not an integer, please try again.")
  }
  # Main Game Part
  roll = sample(1:6, size=value, replace = TRUE)
  # Here we use "vapply" function to calculate the winnings
  winning = sum(vapply(roll, function(x){
    if (x == 3)
    {
      return(4)
    }else if(x == 5)
    {
      return(8)
    }
    else {
      return(-2)
    }
  },numeric(1)
  ))
  return(winning)
}
```

(b)

```
value1  = 3
game.loop(value1)
```

```
[1] -6
```

```
game.vectorized(value1)
```

```
[1] -6
```

```
game.table(value1)
```

```
[1] 0
```

```
game.vapply(value1)
```

```
[1] 0
```

```
value2 = 3000
game.loop(value2)
```

```
[1] 2056
```

```
game.vectorized(value2)
```

```
[1] 2114
```

```
game.table(value2)
```

```
[1] 2150
```

```
game.vapply(value2)
```

```
[1] 2290
```

(c)

```
set.seed(506)
game.loop(3)
```

```
[1] 6
```

```
set.seed(506)
game.vectorized(3)
```

```
[1] 6
```

```
set.seed(506)
game.table(3)
```

```
[1] 6
```

```
  set.seed(506)
  game.vapply(3)
```

```
[1] 6
```

```
  set.seed(5061)
  game.loop(3000)
```

```
[1] 2172
```

```
  set.seed(5061)
  game.vectorized(3000)
```

```
[1] 2172
```

```
  set.seed(5061)
  game.table(3000)
```

```
[1] 2172
```

```
  set.seed(5061)
  game.vapply(3000)
```

```
[1] 2172
```

(d)

```
  library(microbenchmark)
```

```
Warning: package 'microbenchmark' was built under R version 4.3.3
```

```
  microbenchmark(game.loop(1000),
                 game.vectorized(1000),
```

```
                game.table(1000),
                game.vapply(1000))
```

```
Unit: microseconds
                 expr   min     lq    mean median     uq    max neval
       game.loop(1000)  79.3  84.25  95.571  87.80  91.50  437.8   100
 game.vectorized(1000)  66.3  72.95  88.089  78.75  94.45  179.3   100
      game.table(1000) 249.3 283.05 323.564 308.20 340.40  812.8   100
     game.vapply(1000) 386.6 415.15 467.517 424.20 458.75 2279.4   100
```

```
library(microbenchmark)
microbenchmark(game.loop(100000),
                game.vectorized(100000),
                game.table(100000),
                game.vapply(100000))
```

```
Unit: milliseconds
                   expr     min       lq      mean   median       uq     max
       game.loop(1e+05)  7.3471  7.67015  7.926073  7.78465  8.09000 12.0126
 game.vectorized(1e+05)  5.8406  6.94650  7.261184  7.23335  7.50010 11.7906
      game.table(1e+05)  6.4477  7.63095  8.205578  7.84155  8.28075 15.5999
     game.vapply(1e+05) 39.9320 41.67670 43.769824 43.02380 45.07740 68.1716
 neval
   100
   100
   100
   100
```

We can find the vectorized function performance is always the best, the vapply fucntion is always the slowest. Also, when input value is small, the table function needs more time than the loop function, but when input value increased, this situation reverses, the table function will use less time than the loop function.

(e)

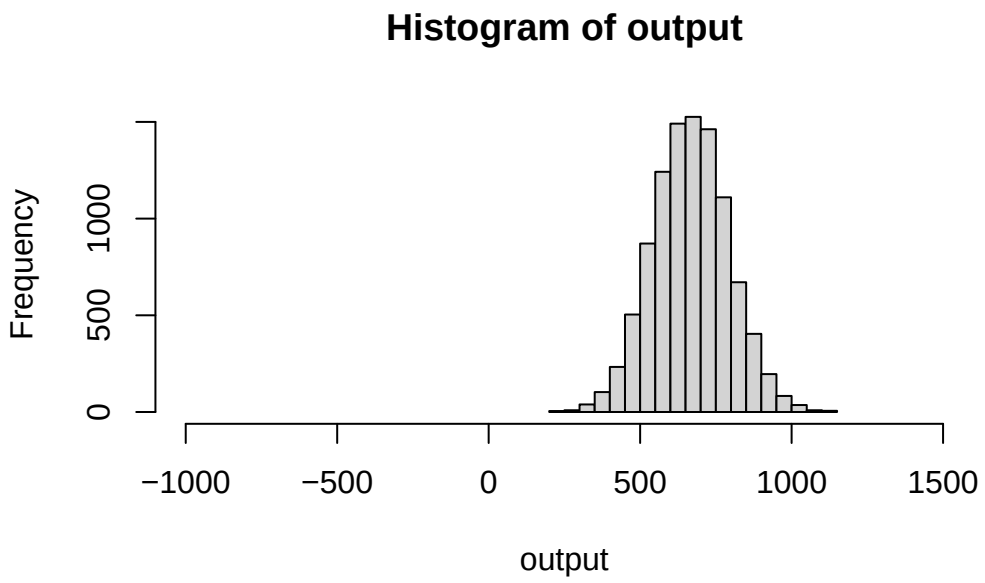
This is not a fair game, the expectation is: # (1/6)x4+(1/6)x8-(4/6)x2 = 2/3 > 0,

so, the expectation is larger than 0, which means if you roll a dice for a large number,

The winning money will larger than lose money. It is not a fair game.

We can us Monte Carlo to do the simulation:

```r
rept = 10000
value = 1000
output = c(1:rept)*0
expectation = c(1:rept)*0
for(i in 1:rept)
{
  # We use vectorized function here, because it is always the fast.
  output[i] = game.vectorized(value)
  expectation[i] = game.vectorized(value)/value
}
hist(output, xlim = c(-1000,1500))
```




```r
sum(expectation)/rept
```

```
[1] 0.6655164
```

From the result, we can find the simulated expectation is very close to 2/3.

Question 2

(a)

```r
data = read.csv("C:/Users/z1883/Desktop/cars.csv",header = TRUE)
colnames(data) = c("Height","Length","Width","Driveline",
                   "Engine.Type","Hybrid","Number.of.Forward.Gears",
                   "Transmission","City.mpg","Fuel.Type","Highway.mpg",
                   "Classification","ID","Make","Model.Year","Year",
                   "Horsepower","Torque")
head(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
                                  Engine.Type Hybrid Number.of.Forward.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)

```r
gasoline.data = data[which(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
                                  Engine.Type Hybrid Number.of.Forward.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
```

```
nrow(gasoline.data)
```

```
[1] 4591
```

(c)

We first check whether the highway MPG data is strictly positive.

```
sum(abs(gasoline.data$Highway.mpg) != gasoline.data$Highway.mpg)
```

```
[1] 0
```

```
sum(gasoline.data$Highway.mpg == 0)
```

```
[1] 0
```

```
sum(is.na(gasoline.data$Highway.mpg))
```

```
[1] 0
```

We can find from the result, the highway MPG data is strictly positive and do not have NA values.

Now, let's check the skewness.

```
library(e1071)
```

```
Warning: package 'e1071' was built under R version 4.3.3
```

```
skewness(gasoline.data$Highway.mpg)
```

```
[1] 7.990895
```

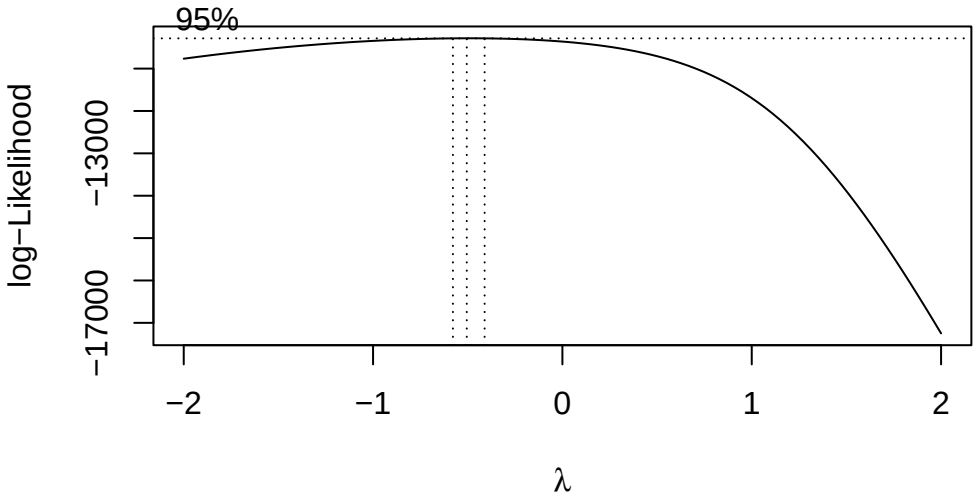
From the result, we can find the data is highly positive skewed. So, we can use Box-Cox Transformation here, box-cox method will also automatically find the optimal power transformation that minimizes deviations from normality and heteroscedasticity.

```
library(MASS)
```

```
Warning: package 'MASS' was built under R version 4.3.2
```

```
highwayMPG = 0
model = lm(Highway.mpg~Torque+Horsepower+Height+Length+Width
           +as.factor(Year), data=gasoline.data)

b = boxcox(model)
```




```
# Exact the best lambda
lambda <- b$x[which.max(b$y)]
lambda
```

```
[1] -0.5050505
```

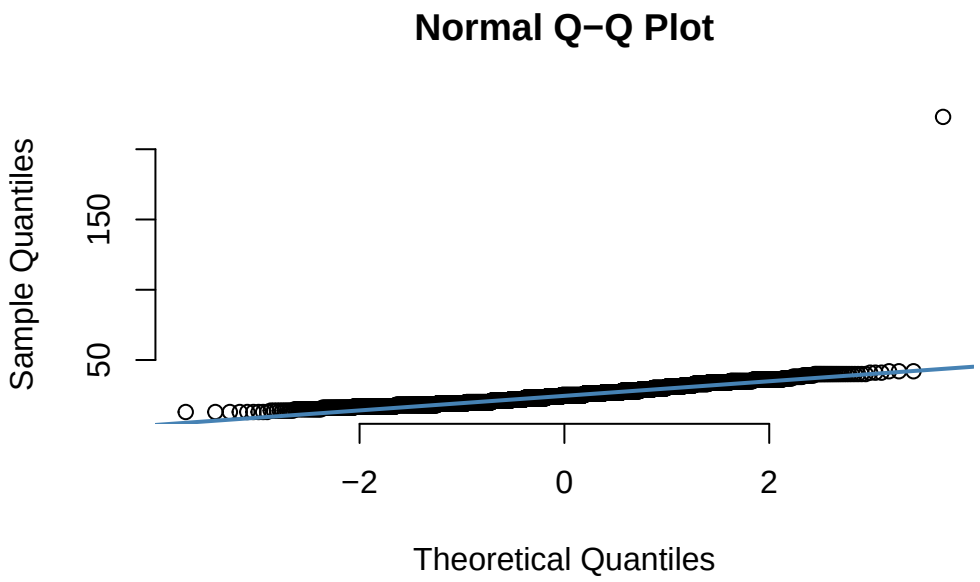
```
if(lambda != 0)
{
  highwayMPG=(gasoline.data$Highway.mpg^lambda - 1)/lambda
```

```
}else{
  highwayMPG=log(gasoline.data$Highway.mpg)
}
# Check the skewness again
skewness(highwayMPG)
```

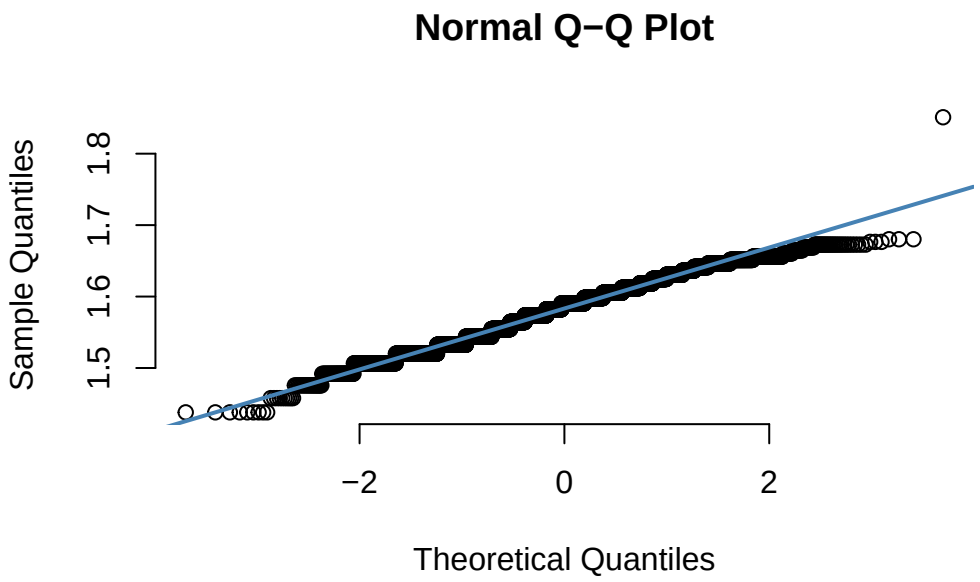
```
[1] -0.1916226
```

Let's compare the data before and after transformation by using QQ plot.

```
# Data before transformation
qqnorm(gasoline.data$Highway.mpg, pch = 1, frame = FALSE)
qqline(gasoline.data$Highway.mpg, col = "steelblue", lwd = 2)
```




```
# Data after transformation
qqnorm(highwayMPG, pch = 1, frame = FALSE)
qqline(highwayMPG, col = "steelblue", lwd = 2)
```



From the plot, we can find the data after the transformation is much better.

Combine the transformed data with the original data set.

```
new.gasoline = cbind(gasoline.data, highwayMPG)
head(new.gasoline)
```

```
  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
                                  Engine.Type Hybrid Number.of.Forward.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 highwayMPG
1        250    236   1.590386
2        200    207   1.612060
3        200    207   1.624660
4        200    207   1.612060
5        200    207   1.612060
6        265    243   1.605239
```

(d)

```r
# We used the transformed highwayMPG data
model1 = lm(highwayMPG~Torque+Horsepower+Height+Length+Width
            +as.factor(Year), data=new.gasoline)
summary(model1)
```

```
Call:
lm(formula = highwayMPG ~ Torque + Horsepower + Height + Length +
    Width + as.factor(Year), data = new.gasoline)

Residuals:
      Min        1Q    Median        3Q       Max
-0.121645 -0.017585  0.000378  0.019087  0.310402

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)          1.647e+00  4.342e-03 379.369  < 2e-16 ***
Torque              -4.838e-04  1.324e-05 -36.554  < 2e-16 ***
Horsepower           2.116e-04  1.368e-05  15.464  < 2e-16 ***
```

```
Height               8.075e-05  6.770e-06  11.929  < 2e-16 ***
Length               5.223e-06  5.309e-06   0.984   0.3252
Width               -2.354e-05  5.434e-06  -4.331 1.52e-05 ***
as.factor(Year)2010 -4.719e-03  4.067e-03  -1.160   0.2459
as.factor(Year)2011 -1.230e-03  4.060e-03  -0.303   0.7619
as.factor(Year)2012  6.790e-03  4.092e-03   1.659   0.0971 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02765 on 4582 degrees of freedom
Multiple R-squared:  0.5782,    Adjusted R-squared:  0.5775
F-statistic: 785.3 on 8 and 4582 DF,  p-value: < 2.2e-16
```

Controlling other variables fixed, we can find the torque has a significant negative relationship with the highwayMPG, which means holding other variables constant, the more the value of torque increased, the less the predict highwayMPG.

(e)

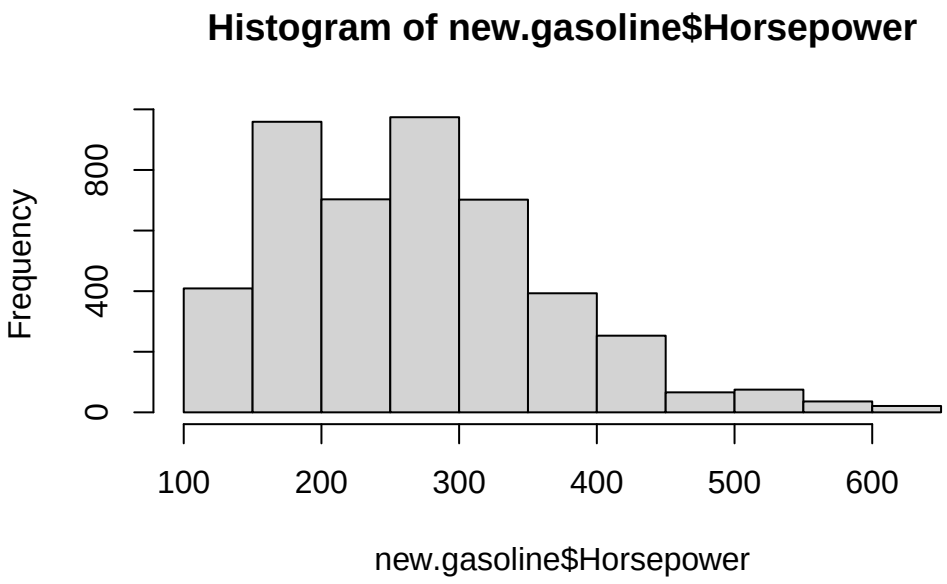
```r
summary(new.gasoline$Horsepower)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  100.0   185.0   263.0   267.5   317.0   638.0
```

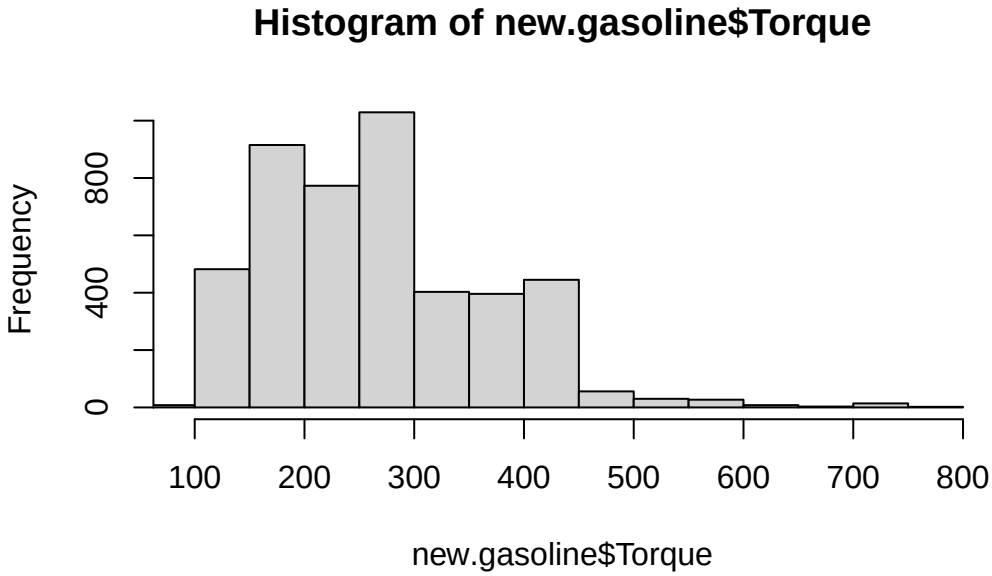
```r
summary(new.gasoline$Torque)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   98.0   177.0   257.0   267.2   332.0   774.0
```

```r
hist(new.gasoline$Horsepower,xlim = c(100,650))
```




```r
hist(new.gasoline$Torque,xlim = c(90,780))
```

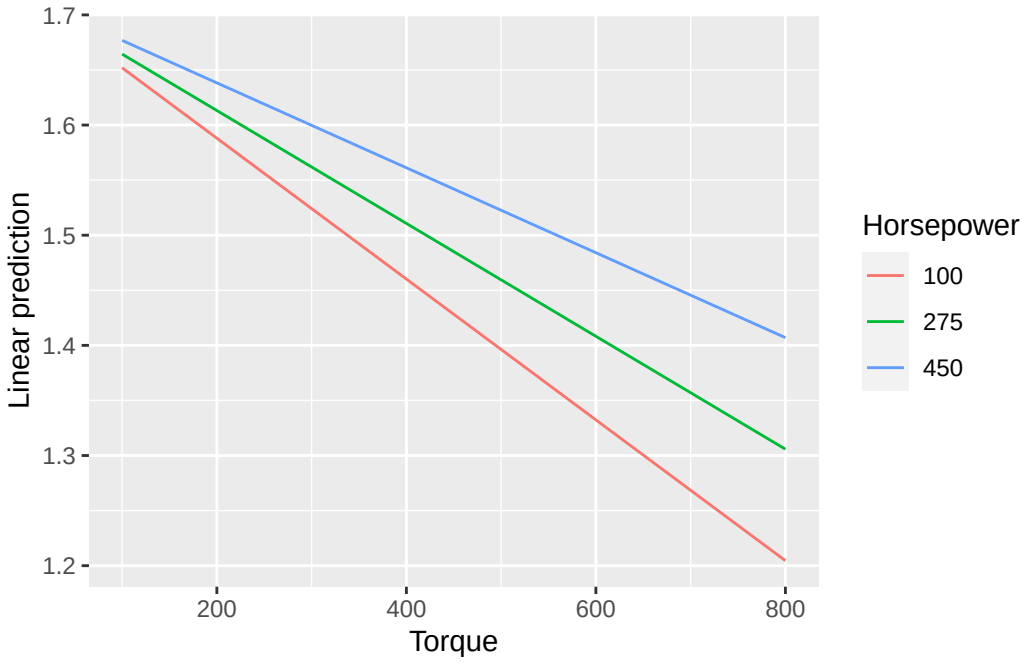


```
library(emmeans)
```

```
Warning: package 'emmeans' was built under R version 4.3.3
```

```
Welcome to emmeans.
Caution: You lose important information if you filter this package's results.
See '? untidy'
```

```
mod = lm(highwayMPG~Torque*Horsepower+Height+Length+Width
           +as.factor(Year), data=new.gasoline)
# From the hist plot above, we can choose three different value for
# Horsepower which is 100, 275, 450.
# As for torque, the range is 98-774, so, we set the sequence
# from 100 - 800 with 100 distance.
emmip(mod,Horsepower~Torque, at = list(Horsepower = c(100,275,450),
                                       Torque = seq(100,800,100)))
```



From the result, when the horsepower decreased, the line for torque is more and more steep, which means, in higher horsepower, the torque has less negative effect on highway MPG.

(f)

We can use this beta.hat = (X'X)^-1X'Y, where X is the designed matrix

```
new.gasoline$Year = as.factor(new.gasoline$Year)
# Find the design matrix
X = model.matrix(highwayMPG~Torque+Horsepower+Height+Length+Width
                 +Year, data = new.gasoline)
head(X)
```

```
  (Intercept) Torque Horsepower Height Length Width Year2010 Year2011 Year2012
1           1    236        250    140    143   202        0        0        0
2           1    207        200    140    143   202        0        0        0
3           1    207        200    140    143   202        0        0        0
4           1    207        200    140    143   202        0        0        0
5           1    207        200    140    143   202        0        0        0
6           1    243        265     91     17    62        0        0        0
```

```
beta.hat = solve(t(X)%*%X)%*%(t(X)%*%new.gasoline$highwayMPG)
beta.hat
```

```
                    [,1]
(Intercept)  1.647078e+00
Torque      -4.838350e-04
Horsepower   2.115799e-04
Height       8.075517e-05
Length       5.223458e-06
Width       -2.353568e-05
Year2010    -4.719219e-03
Year2011    -1.230372e-03
Year2012     6.789545e-03
```

From the result, we can find the beta hat is the same as the estimated coefficient obtained by using lm() function.