STATS506 HW2

AUTHOR
Romeo Ruan

```
        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
```

```
        library(ggplot2)
        library(interactions)
        library(emmeans)
```

```
Warning: package 'emmeans' was built under R version 4.4.1

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

Problem 1

a

```
        #' Use the loop for version 1 of dice game
        #' @param n Number of rolls to make
        #' @return Total won/lost in dollars
        v1 <- function(n ,seed = NULL) {
          if (n < 1) {
            return(0)
          }

          dice <- sample(1:6, n, replace = TRUE)
          total <- 0

          for (k in 1:n) {
            total <- total - 2
            if (dice[k] == 3 | dice[k] == 5) {
              total <- total + 2 * dice[k]
            }
          }
```

```
  return(total)
}


#' Use the loop for version 2 of dice game
#' @param n Number of rolls to make
#' @return Total won/lost in dollars
v2 <- function(n) {
  if (n < 1) {
    return(0)
  }

  dice <- sample(1:6, n, replace = TRUE)
  winnings <- rep(0, n)

  winnings[dice == 3 | dice == 5] <- 2 * dice[dice == 3 | dice == 5]

  total_wins <- sum(winnings) - 2 * n

  return(total_wins)
}


#' Use the loop for version 3 of dice game
#' @param n Number of rolls to make
#' @return Total won/lost in dollars
v3 <- function(n) {
  if (n < 1) {
    return(0)
  }

  dice <- sample(1:6, n, replace = TRUE)
  dice <- table(factor(dice, levels = 1:6))

  total <- dice[3]*6 + dice[5]*10 - 2*n
  names(total) <- NULL

  return(total)
}

#' Use the loop for version 4 of dice game
#' @param n Number of rolls to make
#' @return Total won/lost in dollars
v4 <- function(n) {
  if (n < 1) {
    return(0)
  }

  dice <- sample(1:6, n, replace = TRUE)

  total <- -2 * n + sum(vapply(dice, function(x) {
    if (x == 3 | x == 5) {
      return(2*x)
```

```
    } else {
      return(0)
    }
  }, 1))

  return(total)
}
```

b

```
c(v1(3), v2(3), v3(3), v4(3))
```

```
[1]  0  6  4 14
```

```
c(v1(3000), v2(3000), v3(3000), v4(3000))
```

```
[1] 2170 2084 2200 1992
```

c

```
#' Use the loop for version 1 of dice game
#' @param n Number of rolls to make
#' @return Total won/lost in dollars
v1 <- function(n ,seed = NULL) {
  if (n < 1) {
    return(0)
  }

  set.seed(seed)
  dice <- sample(1:6, n, replace = TRUE)
  total <- 0

  for (k in 1:n) {
    total <- total - 2
    if (dice[k] == 3 | dice[k] == 5) {
      total <- total + 2 * dice[k]
    }
  }

  return(total)
}


#' Use the loop for version 2 of dice game
#' @param n Number of rolls to make
#' @return Total won/lost in dollars
v2 <- function(n ,seed = NULL) {
  if (n < 1) {
    return(0)
  }
```

```
  set.seed(seed)
  dice <- sample(1:6, n, replace = TRUE)
  winnings <- rep(0, n)

  winnings[dice == 3 | dice == 5] <- 2 * dice[dice == 3 | dice == 5]

  total_wins <- sum(winnings) - 2 * n

  return(total_wins)
}


#' Use the loop for version 3 of dice game
#' @param n Number of rolls to make
#' @return Total won/lost in dollars
v3 <- function(n, seed = NULL) {
  if (n < 1) {
    return(0)
  }

  set.seed(seed)
  dice <- sample(1:6, n, replace = TRUE)
  dice <- table(factor(dice, levels = 1:6))

  total <- dice[3]*6 + dice[5]*10 - 2*n
  names(total) <- NULL

  return(total)
}

#' Use the loop for version 4 of dice game
#' @param n Number of rolls to make
#' @return Total won/lost in dollars
v4 <- function(n, seed = NULL) {
  if (n < 1) {
    return(0)
  }

  set.seed(seed)
  dice <- sample(1:6, n, replace = TRUE)

  total <- -2 * n + sum(vapply(dice, function(x) {
    if (x == 3 | x == 5) {
      return(2*x)
    } else {
      return(0)
    }
  }, 1))

  return(total)
}

c(v1(3, seed = 1), v2(3, seed = 1), v3(3, seed = 1), v4(3, seed = 1))
```

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

```
        c(v1(3000, seed = 123), v2(3000, seed = 123), v3(3000, seed = 123), v4(3000, s
```

```
[1] 2174 2174 2174 2174
```

d

```
        library(microbenchmark)
```

```
Warning: package 'microbenchmark' was built under R version 4.4.1
```

```
        microbenchmark(loop   = v1(1000, seed = 13),
                       vctrzd = v2(1000, seed = 13),
                       table  = v3(1000, seed = 13),
                       apply  = v4(1000, seed = 13))
```

```
Warning in microbenchmark(loop = v1(1000, seed = 13), vctrzd = v2(1000, : less
accurate nanosecond times to avoid potential integer overflows

Unit: microseconds
   expr     min       lq      mean   median       uq      max neval
   loop 163.221 168.1820 180.98671 177.1405 194.2580  229.190   100
 vctrzd  51.783  54.7555  59.16628  58.0765  61.2950   87.904   100
  table 102.705 106.7845 114.36458 113.1600 118.7155  180.605   100
  apply 439.233 463.5255 547.38895 503.1110 570.4945 3603.531   100
```

```
        microbenchmark(loop   = v1(100000, seed = 13),
                       vctrzd = v2(100000, seed = 13),
                       table  = v3(100000, seed = 13),
                       apply  = v4(100000, seed = 13))
```

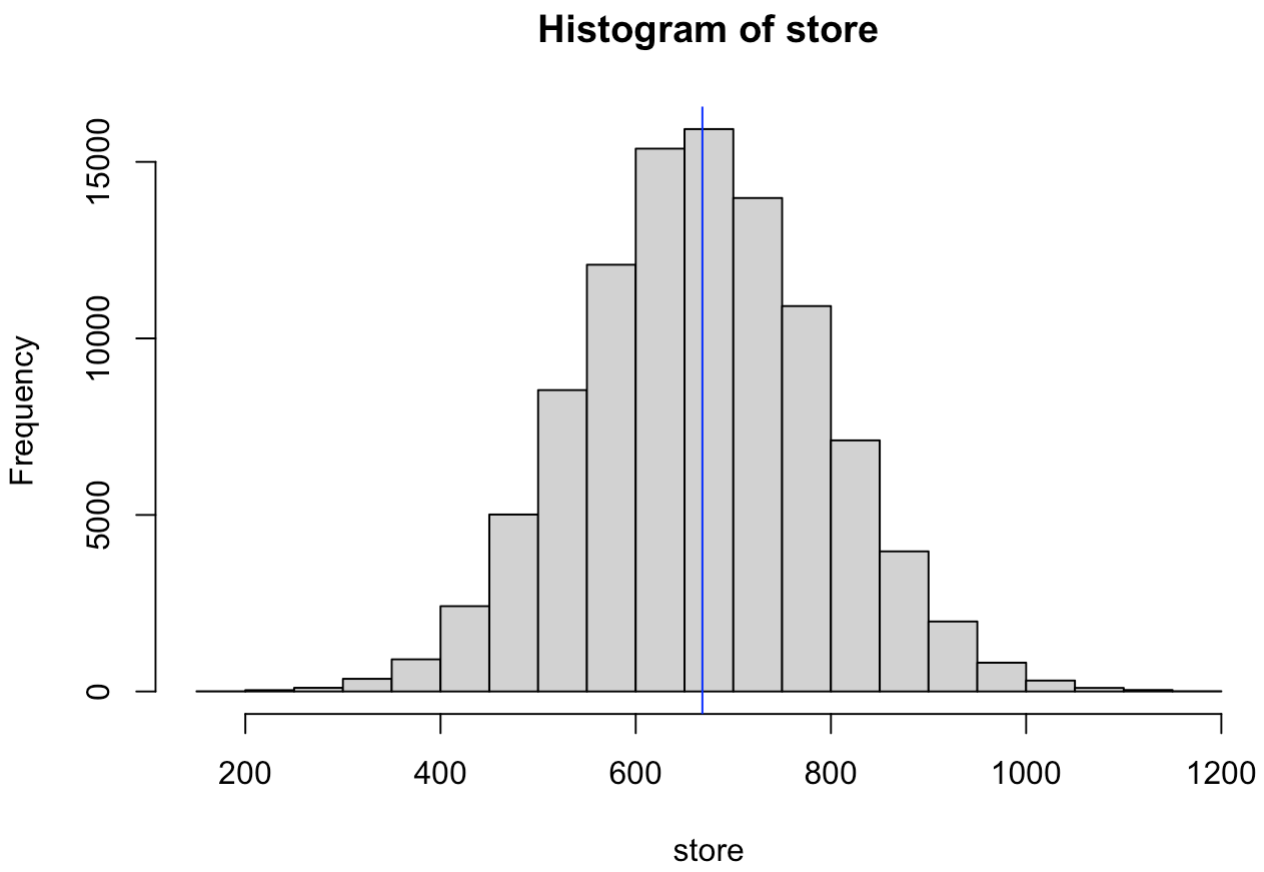
```
Unit: milliseconds
   expr       min        lq      mean    median        uq       max neval
   loop 15.732233 16.033009 16.500239 16.234216 16.648276 19.173240   100
 vctrzd  5.512778  5.857055  6.048265  5.998382  6.194198  7.865235   100
  table  6.793495  7.143471  8.005690  7.284450  7.398184 63.304041   100
  apply 44.416899 45.071935 46.596189 46.109625 46.870401 60.207434   100
```

Conclusion: loop is quick small input, but becomes less efficient for large-scale operations. Vectorized Approach is the most efficient for both small and large inputs. Table performs relatively well for larger inputs but is slower for small inputs due to the overhead of creating the table. Apply is consistently the slowest approach due to the overhead of applying functions based on elements.

e

```
        rep <- 100000
        store <- vector(length = rep)
        for (i in 1:rep) {
```

```
  store[i] <- v2(1000)
}
hist(store)
abline(v = mean(store), col = "blue")
```



If we calculate its expected value, we have: (6 * 1/6)+(10 * 1/6)+(0 * 4/6)-2=0.67. Also from our simulation, we can see that it is not a fair game in that the blue line does not indicate at 0.

#Problem 2

a

```
cars_data <- read.csv("/Users/yining/Downloads/cars.csv")
names(cars_data) <- c(
"Height", "Length", "Width", "Driveline", "Engine_Type", "Hybrid",
"Fwd_Gears", "Transmission", "City_MPG", "Fuel_Type", "Highway_MPG",
"Classification", "ID", "Make", "Model_Year", "ID_Year", "HP", "Torque")
head(cars_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 Fwd_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 ID_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
   HP Torque
1 250    236
2 200    207
3 200    207
4 200    207
5 200    207
6 265    243
```

b

```r
table(cars_data$Fuel_Type)
```

```
Compressed natural gas            Diesel fuel                    E85
                     2                     27                     456
             Gasoline
                  4591
```

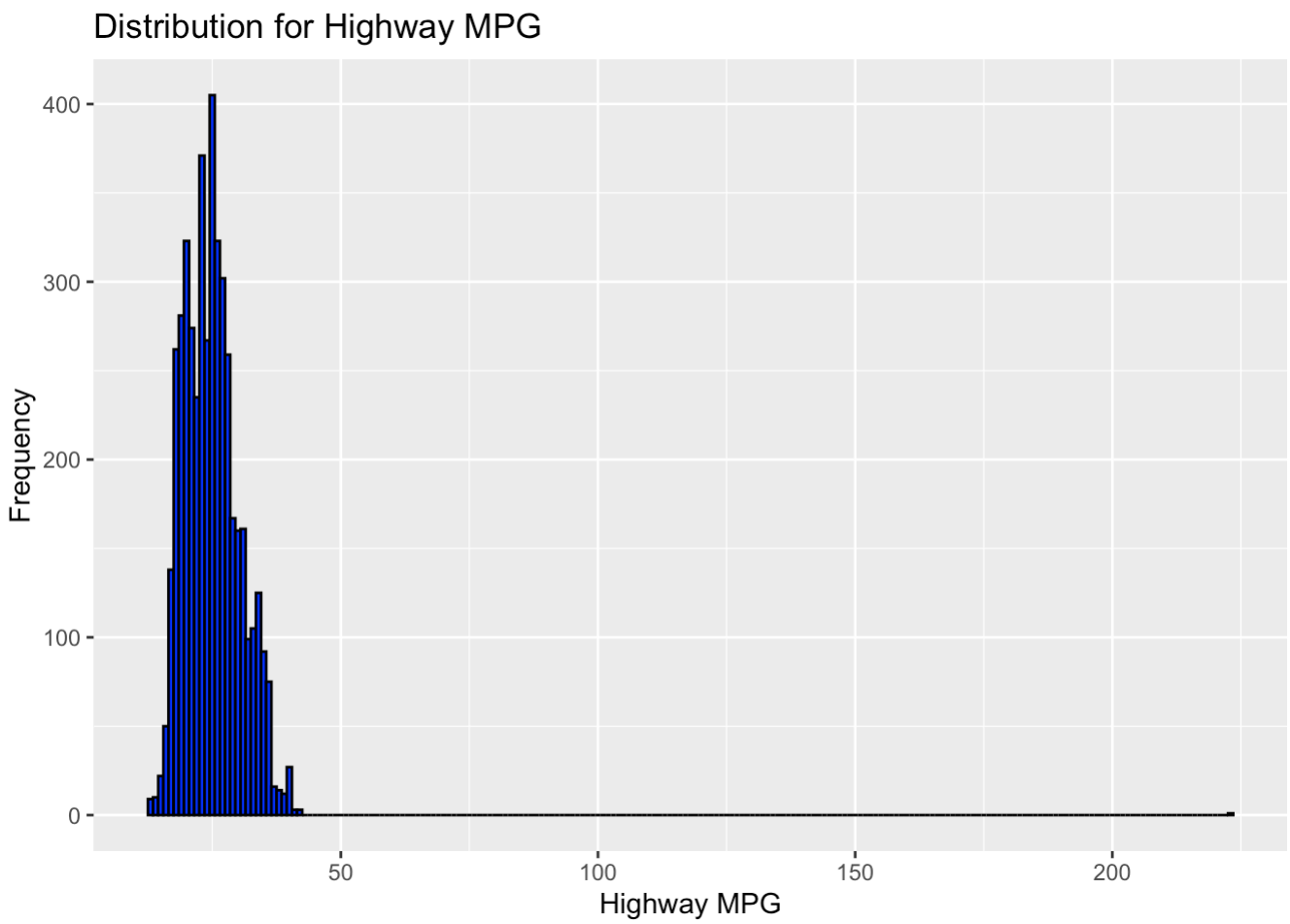
```r
filtered_data <- subset(cars_data, Fuel_Type == "Gasoline")
nrow(filtered_data)
```

```
[1] 4591
```

c

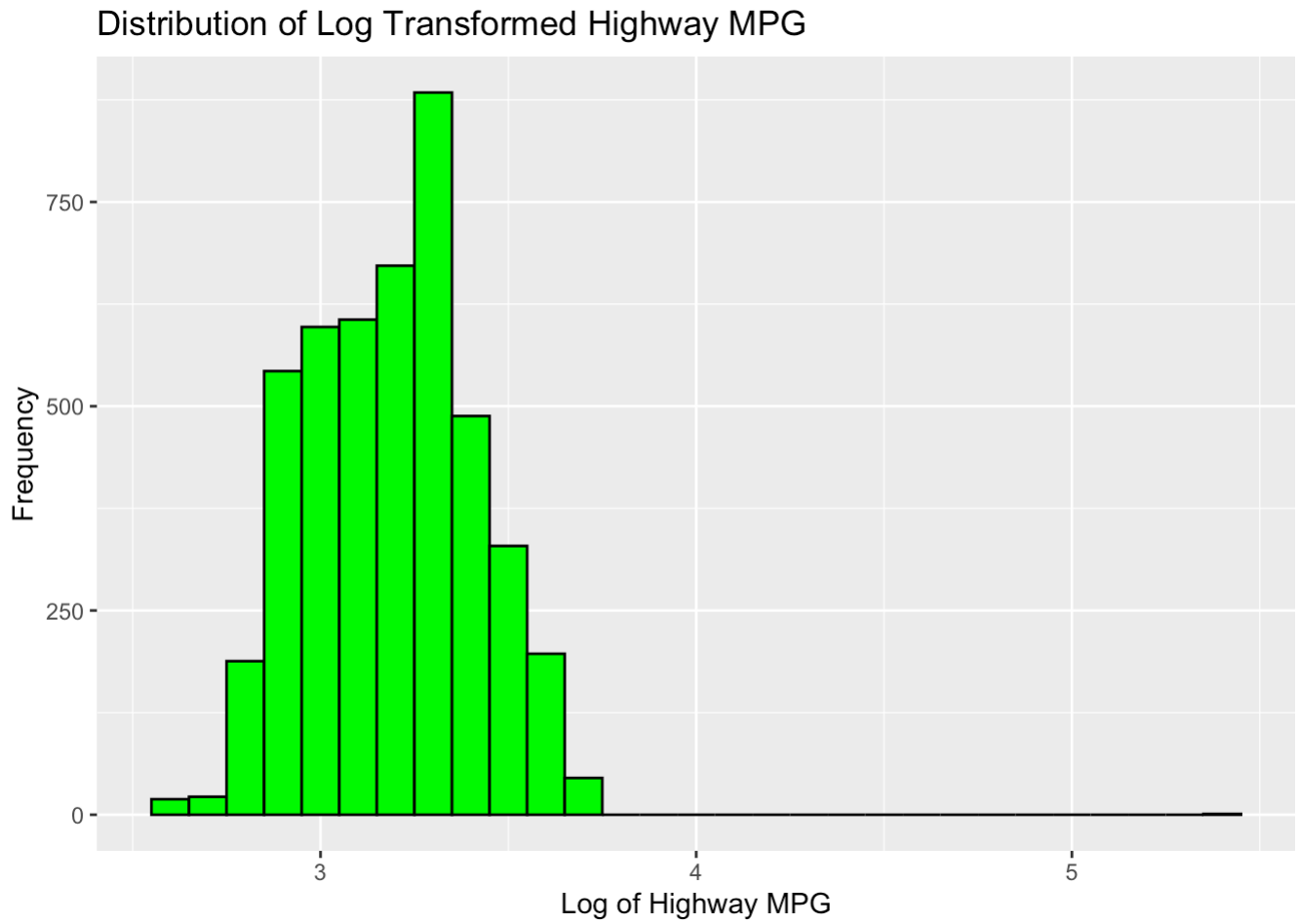
```r
ggplot(filtered_data, aes(x = Highway_MPG)) +
  geom_histogram(binwidth = 1, color = "black", fill = "blue") +
  ggtitle("Distribution for Highway MPG") +
```

```
  xlab("Highway MPG") +
  ylab("Frequency")
```



From the graph, we can see the diagram is especially right-skewed, so a transformation should be used, so let's do a log transformation and generate a more suitable graph:

```
filtered_data$log_Highway_MPG <- log(filtered_data$Highway_MPG)
ggplot(filtered_data, aes(x = log_Highway_MPG)) +
  geom_histogram(binwidth = 0.1, color = "black", fill = "green") +
  ggtitle("Distribution of Log Transformed Highway MPG") +
  xlab("Log of Highway MPG") +
  ylab("Frequency")
```



d

```
model <- lm(Highway_MPG ~ HP + Torque + Height + Length + Width +
              as.factor(ID_Year), data = filtered_data)
summary(model)
```

```
Call:
lm(formula = Highway_MPG ~ HP + Torque + Height + Length + Width +
    as.factor(ID_Year), data = filtered_data)

Residuals:
    Min      1Q  Median      3Q     Max
-10.824  -2.550  -0.452   2.372 202.639

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)             32.2926630  0.7225982  44.690  < 2e-16 ***
HP                       0.0163556  0.0022772   7.182 7.96e-13 ***
Torque                  -0.0507425  0.0022030 -23.034  < 2e-16 ***
Height                   0.0099079  0.0011267   8.794  < 2e-16 ***
Length                   0.0017290  0.0008836   1.957   0.0504 .
Width                   -0.0003343  0.0009045  -0.370   0.7117
as.factor(ID_Year)2010 -0.4539681  0.6768246  -0.671   0.5024
as.factor(ID_Year)2011  0.1711016  0.6757043   0.253   0.8001
```

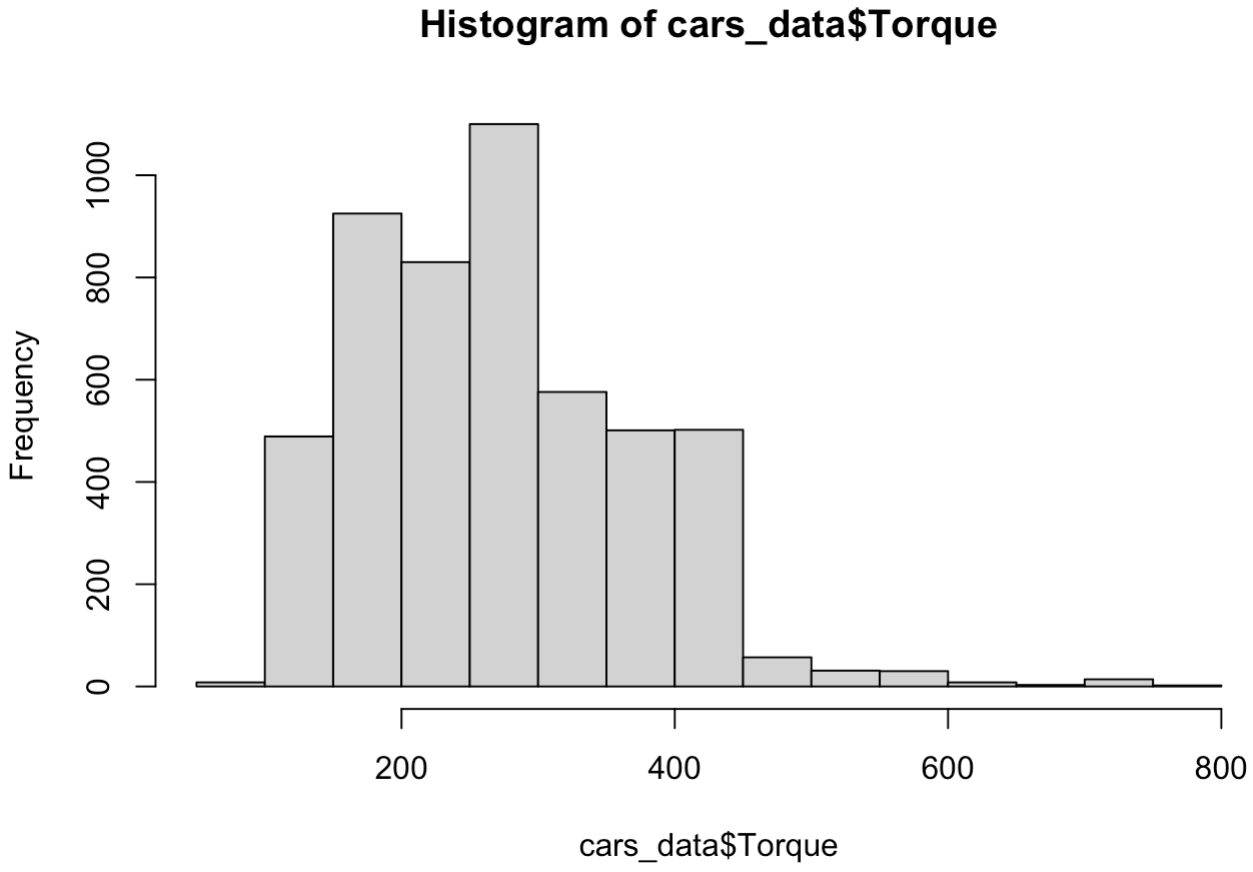
```
as.factor(ID_Year)2012  1.3029279  0.6810076   1.913   0.0558 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.602 on 4582 degrees of freedom
Multiple R-squared:  0.4192,    Adjusted R-squared:  0.4182
F-statistic: 413.3 on 8 and 4582 DF,  p-value: < 2.2e-16
```

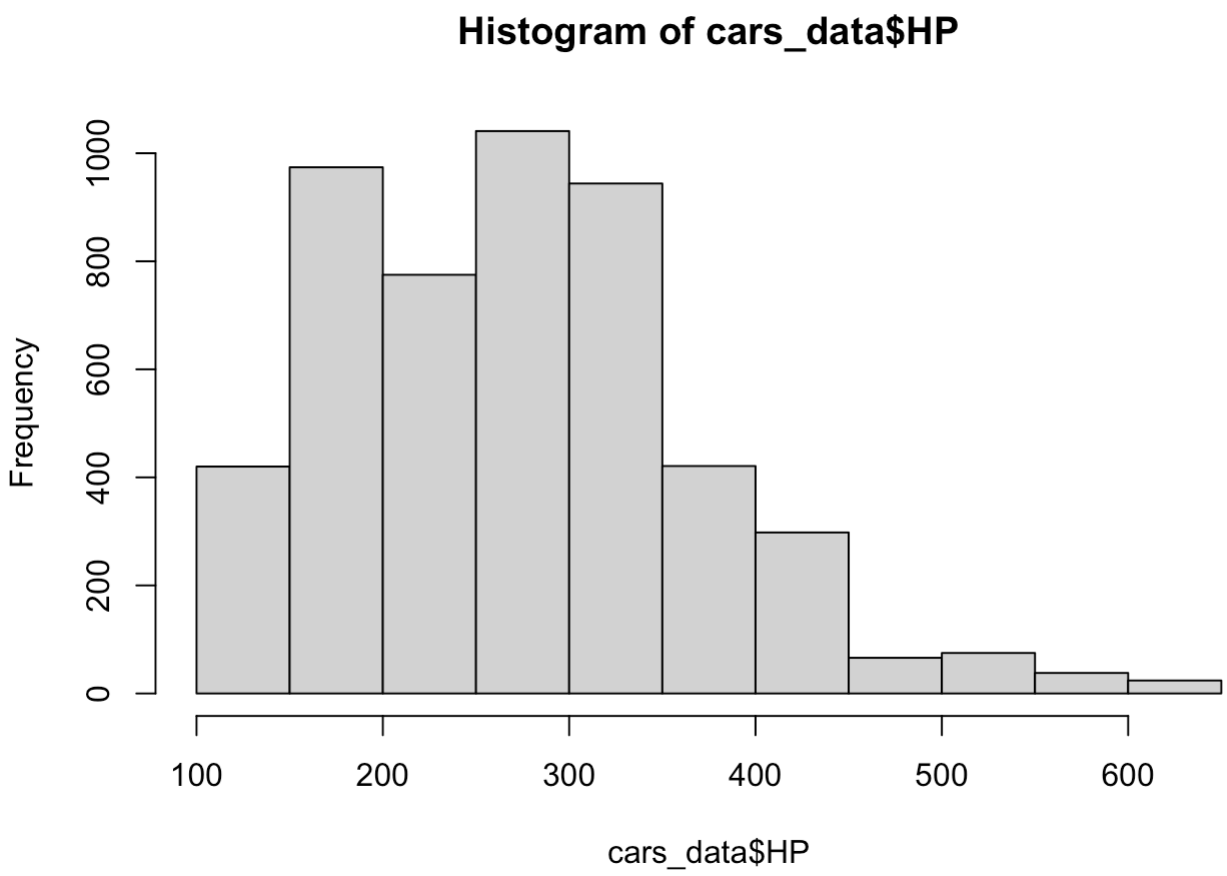
The coefficient for Torque is -0.0507425, and this indicates that higher torque results in lower highway mpg, on average.

e

```
model <- lm(Highway_MPG ~ Torque*HP + Height + Length + Width +
            as.factor(ID_Year), data = filtered_data)
hist(cars_data$Torque)
```




```
hist(cars_data$HP)
```




```
summary(cars_data$HP)
```

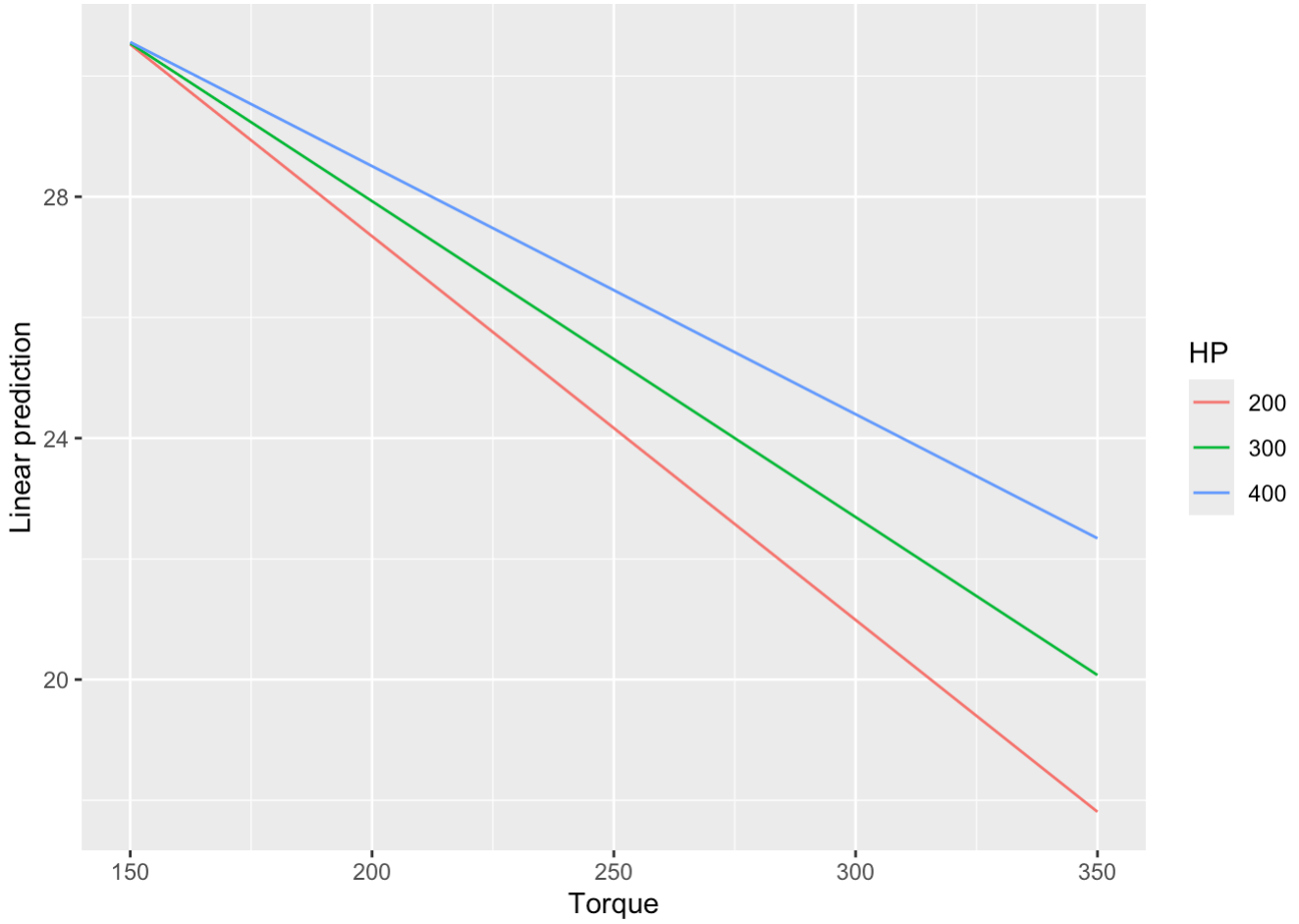
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  100.0   190.0   266.0   270.5   317.0   638.0 
```

```
summary(cars_data$Torque)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   98.0   187.0   260.0   272.7   335.0   774.0 
```

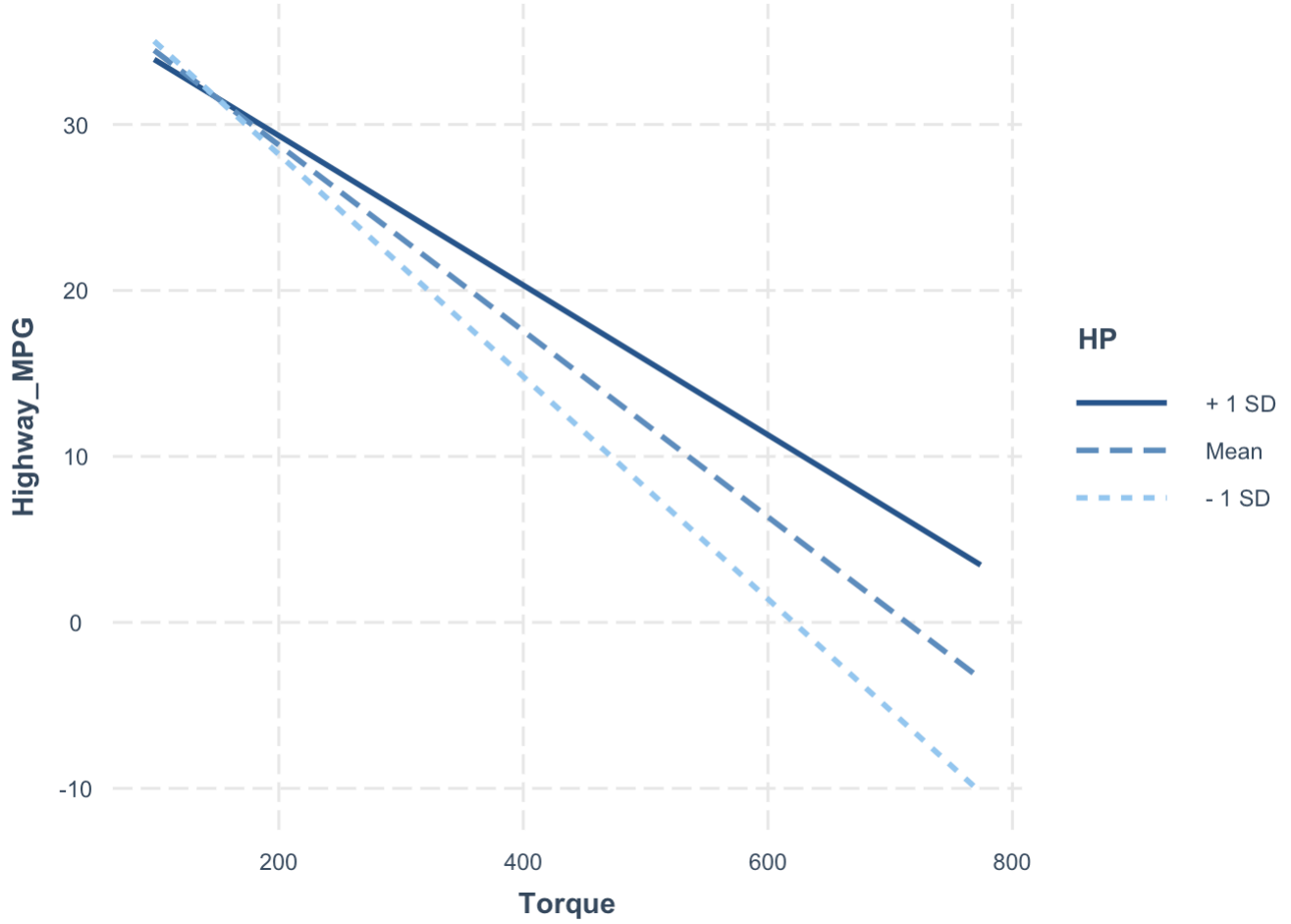
From the tables, we can see that horsepower ranges from 100 to 638, and torque ranges from 98 to 774. Horsepower ranges from 100 to 600, and most torque does not exceed 400. Based on this, we can visualizae the interaction:

```
emmip(model, HP ~ Torque, at = list(Torque = seq(150, 400, 100), HP = c(200, 3
```




```
        interact_plot(model, pred = Torque, modx = HP,
                      at = list(ID_Year = 2012))
```

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



f

```
X <- model.matrix(Highway_MPG ~ Torque*HP + Height + Length + Width +
                    as.factor(ID_Year), data = filtered_data)
y <- filtered_data$Highway_MPG
betahat <- solve(t(X)%*%X)%*%t(X)%*%y
cbind(model$coef, betahat)
```

```
                                [,1]          [,2]
(Intercept)            42.1879478687 42.1879478687
Torque                 -0.0860592704 -0.0860592704
HP                     -0.0166633227 -0.0166633227
Height                  0.0065603903  0.0065603903
Length                  0.0017767232  0.0017767232
Width                  -0.0011694485 -0.0011694485
as.factor(ID_Year)2010 -0.5627857770 -0.5627857770
as.factor(ID_Year)2011  0.0725356431  0.0725356431
as.factor(ID_Year)2012  1.1970329986  1.1970329986
Torque:HP               0.0001123567  0.0001123567
```

Github Rep Link: https://github.com/romeor26/STATS-506