STATS506_PS2_code

My Github repository: https://github.com/hzhaoar/Stats_506_PS2

Problem 1

Task (a)

Before we start, we need to define a random number generator to control the randomization in all 4 versions of `play_dice` function.

```
#' Function to create a random number generator function
#'
#' @param seed a numeric value
#' @return a function that receives n as input
#' and generates a n-dimensional random vector using the seed
myRng <- function(seed) {
  set.seed(seed)
  return(function(n) {
    sample(1:6, n, replace = TRUE)
  })
}
```

Then, we try to implement 4 versions of `play_dice`function.

```
#' Version 1: Implement this game using a loop over the die rolls.
#'
#' @param num_rolls an integer indicating the number of rolls
#' @param rng a function as a random number generator
#' @return total net revenue of the dice-rolls
play_dice_v1 <- function(num_rolls, rng) {

  # Generate the rolls using given RNG
```

```r
  rolls <- rng(num_rolls)

  # Iterate over `num_rolls` to calculate total revenue
  revenue <- 0
  for (i in 1:num_rolls) {
    if (rolls[i] %in% c(2, 4, 6)) {
      revenue <- revenue + rolls[i]
    }
  }

  # Calculate net income using net_income = revenue - cost
  cost <- 2*num_rolls
  return(revenue-cost)
}
```

```r
#' Version 2: Implement this game using built-in R vectorized functions.
#'
#' @param num_rolls an integer indicating the number of rolls
#' @param rng a function as a random number generator
#' @return total net revenue of the dice-rolls
play_dice_v2 <- function(num_rolls, rng) {

  # Generate the rolls using given RNG
  rolls <- rng(num_rolls)

  # Calculate revenue using R vectorized function
  revenue <- sum(2*(rolls == 2)+ 4*(rolls == 4) + 6*(rolls == 6))

  # Calculate net income using net_income = revenue - cost
  cost <- 2*num_rolls
  return(revenue-cost)
}
```

```r
#' Version 3: Implement this by collapsing the die rolls into a single table().
#'
#' @param num_rolls an integer indicating the number of rolls
#' @param rng a function as a random number generator
#' @return total net revenue of the dice-rolls
play_dice_v3 <- function(num_rolls, rng) {

  # Generate the rolls using given RNG
```

```
  rolls <- rng(num_rolls)

  # Collapse the die rolls into a single table
  roll_counts <- table(rolls)

  # Iterate over items in the table
  # Notice that the length of the table can be 6 at most.
  # So this is not computationally costly.
  revenue <- 0
  for (i in names(roll_counts)){
    if (as.numeric(i) == 2){
      revenue <- revenue + 2*roll_counts[as.character(i)]
      next
    }
    if (as.numeric(i) == 4){
      revenue <- revenue + 4*roll_counts[as.character(i)]
      next
    }
    if (as.numeric(i) == 6){
      revenue <- revenue + 6*roll_counts[as.character(i)]
    }
  }
  revenue <- as.numeric(revenue)

  # Calculate net income using net_income = revenue - cost
  cost <- 2*num_rolls
  return(revenue-cost)
}
```

```
#' Version 4: Implement this game by using one of the "apply" functions.
#'
#' @param num_rolls an integer indicating the number of rolls
#' @param rng a function as a random number generator
#' @return total net revenue of the dice-rolls
play_dice_v4 <- function(num_rolls, rng) {

  # Generate the rolls using given RNG
  rolls <- rng(num_rolls)

  # Use "apply" to calculate the revenue
  # To use "apply", we choose to convert vector "rolls" into a matrix
```

```
  revenue <- sum(apply(matrix(rolls, ncol = num_rolls), 2, function(row) {
    if (row %in% c(2,4,6)){
      return(row)
    } else{
      return(0)
    }
  }))

  # Calculate net income using net_income = revenue - cost
  cost <- 2*num_rolls
  return(revenue-cost)
}
```

Task (b)

In this task, we will show that all versions work. Notice that we will pass a random number into the function `myRng`, since we do not need to fix the result at this moment.

```
for (t in c(3,3000)) {
  cat("Result for ", t, " using v1 is ",
      play_dice_v1(t, myRng(sample.int(1000, 1))), "\n")
  cat("Result for ", t, " using v2 is ",
      play_dice_v2(t, myRng(sample.int(1000, 1))), "\n")
  cat("Result for ", t, " using v3 is ",
      play_dice_v3(t, myRng(sample.int(1000, 1))), "\n")
  cat("Result for ", t, " using v4 is ",
      play_dice_v4(t, myRng(sample.int(1000, 1))), "\n")
}
```

```
Result for  3  using v1 is  -6
Result for  3  using v2 is  4
Result for  3  using v3 is  -6
Result for  3  using v4 is  2
Result for  3000  using v1 is  -36
Result for  3000  using v2 is  -22
Result for  3000  using v3 is  126
Result for  3000  using v4 is  -162
```

Task (c)

In this task, we will show that the four versions give the same result. We will control the randomization by putting the same seed to RNG.

```
seed <- 114
for (t in c(3,3000)) {
  cat("Result for ", t, " using v1 is ", play_dice_v1(t, myRng(seed)), "\n")
  cat("Result for ", t, " using v2 is ", play_dice_v2(t, myRng(seed)), "\n")
  cat("Result for ", t, " using v3 is ", play_dice_v3(t, myRng(seed)), "\n")
  cat("Result for ", t, " using v4 is ", play_dice_v4(t, myRng(seed)), "\n")
}
```

```
Result for  3  using v1 is  2
Result for  3  using v2 is  2
Result for  3  using v3 is  2
Result for  3  using v4 is  2
Result for  3000  using v1 is  -146
Result for  3000  using v2 is  -146
Result for  3000  using v3 is  -146
Result for  3000  using v4 is  -146
```

It is clear that the four versions give the same result.

Task (d)

First, we need to import `microbenchmark` library.

```
library(microbenchmark)
```

Then, we test the performance with a low input.

```
# Benchmark with low input (100)
seed <- 896
benchmark_low <- microbenchmark(
  v1 = play_dice_v1(100, myRng(seed)),
  v2 = play_dice_v2(100, myRng(seed)),
  v3 = play_dice_v3(100, myRng(seed)),
  v4 = play_dice_v4(100, myRng(seed)),
  times = 100
)
```

```
  print(benchmark_low)
```

```
Unit: microseconds
 expr   min     lq    mean median     uq    max neval
   v1  67.6  72.20  88.938  76.10  87.50  395.5   100
   v2  10.6  11.35  17.190  12.30  14.60  117.6   100
   v3  64.6  74.30 118.159  81.95 101.05 1676.1   100
   v4 138.6 148.45 180.564 159.60 182.25  563.7   100
```

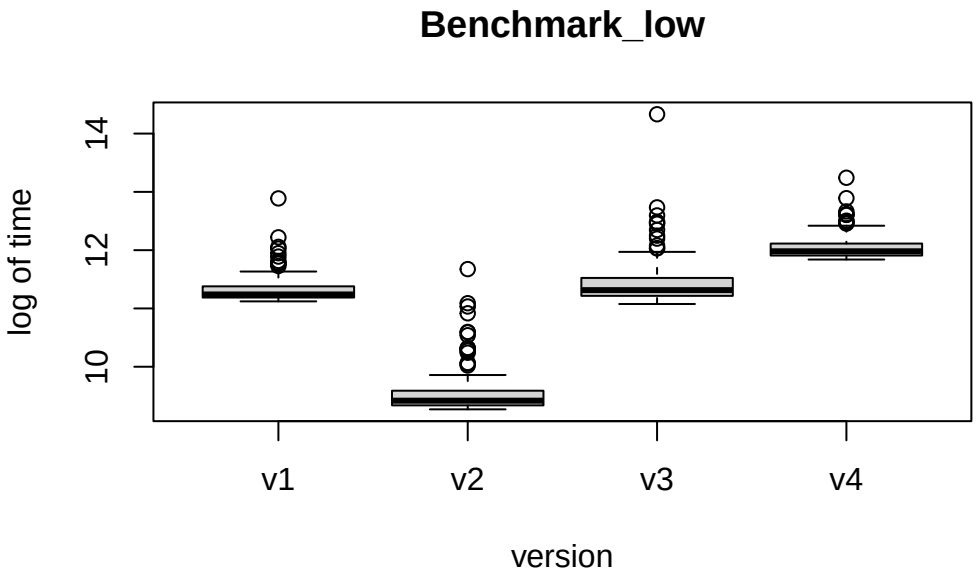
And we test the performance with a large input.

```
  # Benchmark with large input (10000)
  benchmark_large <- microbenchmark(
    v1 = play_dice_v1(10000, myRng(seed)),
    v2 = play_dice_v2(10000, myRng(seed)),
    v3 = play_dice_v3(10000, myRng(seed)),
    v4 = play_dice_v4(10000, myRng(seed)),
    times = 100
  )
  print(benchmark_large)
```

```
Unit: microseconds
 expr     min       lq      mean   median       uq     max neval
   v1  6855.0  7566.70  9262.357  8853.65 10644.15 13759.3   100
   v2   480.8   519.80   583.760   561.30   624.90  1014.1   100
   v3   746.6   878.25   994.217   934.00  1059.10  1875.6   100
   v4 13948.3 16477.40 18892.856 18546.35 20330.60 33759.9   100
```

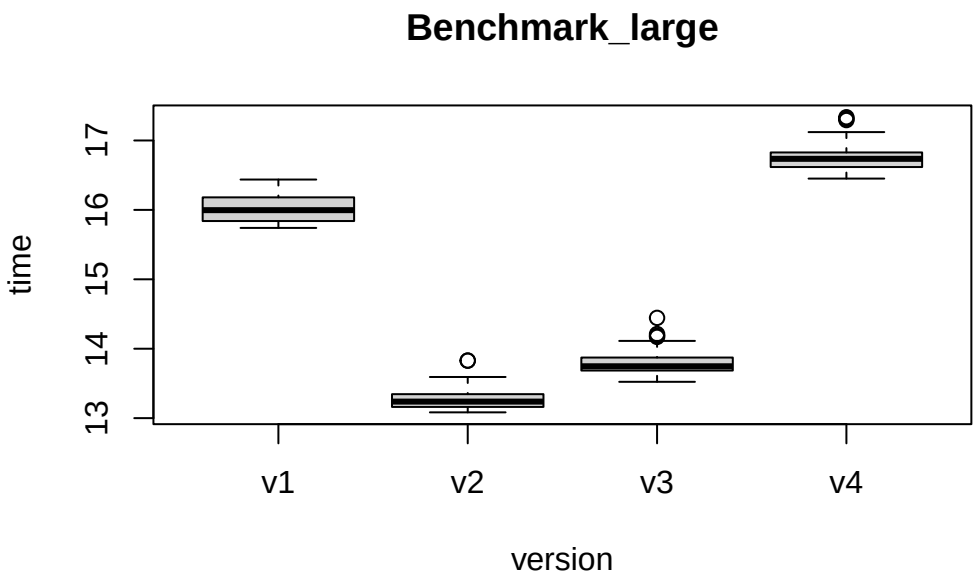
Notice that the scale of time using each version differs largely. So, we may take log-transformation of time to make the plot.

```
  log_time_low <- benchmark_low
  log_time_low$time <- log(benchmark_low$time)
  plot(log_time_low, xlab = "version", ylab = "log of time", main = "Benchmark_low")
```



By the boxplot, we can see that for low input, the running time for v1 and v3 are close to each other. And v2 is the fastest one while v4 is the lowest one on average.

```
log_time_large <- benchmark_large
log_time_large$time <- log(benchmark_large$time)
plot(log_time_large, xlab = "version", ylab = "time", main = "Benchmark_large")
```



By the boxplot, we can see that for large input, running time for 4 versions can be ranked as $v2 < v3 < v1 < v4$ on average.

Task (e)

For convenience, we will only use `play_dice_v2` in this problem, since the results of 4 versions of function are the same. In a single call of `play_dice_v2` function, we will roll the dice `num_rolls` of times. And we will record the average net income per roll among `num_rolls` times of dice rolls.

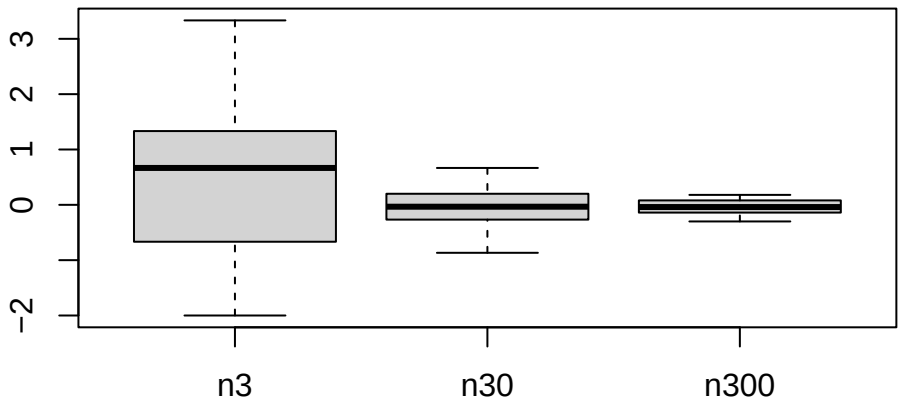
In the following Monte Carlo simulation, we will choose 3 different values for `num_rolls`, which are $3, 30, 300$. And we will repeat this experiment for 100 times.

```
reps <- 100

# Create 3 empty vectors to record the result
n3 <- vector(length = reps)
n30 <- vector(length = reps)
n300 <- vector(length = reps)

# Monte Carlo simulation
for (i in seq_len(reps)) {
```

```
  n3[i] <- play_dice_v2(3,myRng(sample.int(1000, 1)))/3
  n30[i] <- play_dice_v2(30,myRng(sample.int(1000, 1)))/30
  n300[i] <- play_dice_v2(300,myRng(sample.int(1000, 1)))/300
}
```

Here, we can plot our result on using boxplot.

```
boxplot(data.frame(n3, n30, n300))
```



We can see for n_{30} and n_{300}, the average net income per roll is very close to 0. This is not surprising because theoretically,

$$\mathbb{E}(net\ income) = \mathbb{E}(revenue) - \mathbb{E}(cost) = \frac{1}{6}(0+2+0+4+0+6) - 2 = 0$$

Therefore, we can conclude that this game is a fair game.

Problem 2

First, we read the data.

```r
data <- read.csv("./cars.csv")
data <- data.frame(data)
```

Task (a)

Rename the columns of the data to more reasonable lengths.

```r
colnames(data) <- c("height", "length", "width", "driveLine", "engineType",
                    "isHybrid", "numGears", "transmission", "cityMPG",
                    "fuelType", "highwayMPG", "classification", "ID",
                    "make", "modelYear", "IDYear", "horsepower", "torque")
```

Task (b)

Restrict the data to cars whose Fuel Type is "Gasoline"

```r
data <- data[which(data$fuelType == "Gasoline"),]
```

Task (c)

We will fit a linear regression model predicting MPG on the highway in this task. In the formula of this linear model, we should treat "highwayMPG" as the response variable, horsepower as the predictor, and "torque", "height", "length", "width", and factorized "IDYear" as the control variable.

```r
# Fit the linear model
# "factor" treats "IDYear" as a categorical variable,
# where "2009" is chosen to be the base class by default
data$IDYear <- as.factor(data$IDYear)
M1 <- lm(highwayMPG~IDYear+horsepower+torque+height+length+width, data = data)
summary(M1)
```

```
Call:
lm(formula = highwayMPG ~ IDYear + horsepower + torque + height +
    length + width, data = 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 ***
IDYear2010  -0.4539681  0.6768246  -0.671   0.5024
IDYear2011   0.1711016  0.6757043   0.253   0.8001
IDYear2012   1.3029279  0.6810076   1.913   0.0558 .
horsepower   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
---
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
```

By the summary of this linear model, we can find that the p-values for "horsepower", "torque", and "height" are significant, which means these three variables are significant in our model.

And noticing that the estimate for the coefficient of "horsepower" is positive, which implies that as horsepower increases, highway MPG tends to increase. To be more precise, a 1-unit increase in the horsepower tend to cause 0.0163556 units of increase in highway MPG for a car, if other variables remain the same.

Lastly, the multiple R-squared is only 0.4192, which means our model should be incomplete and we should include more variables in this model.

Task (d)

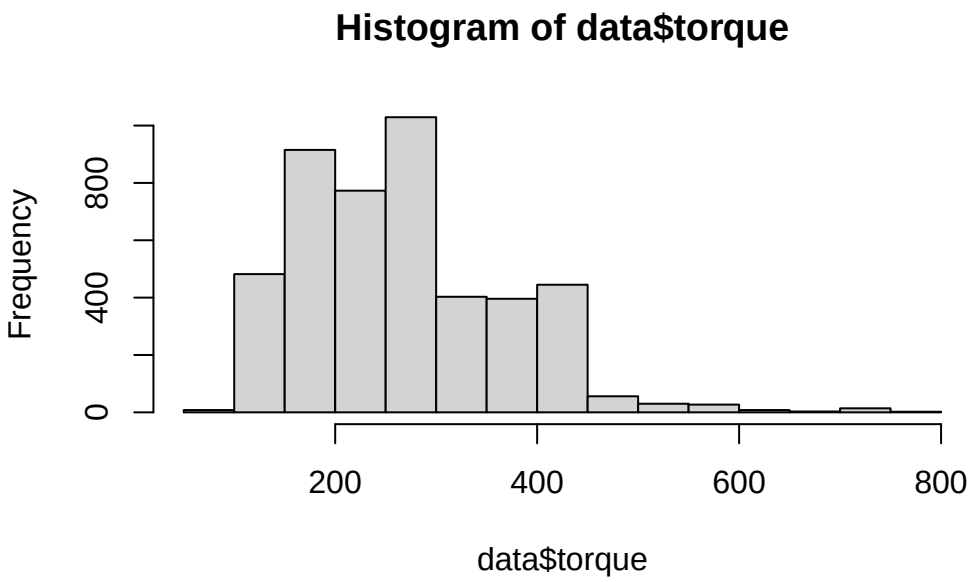
In this task, we will first obtain the model with intersection. Noticing that by `horsepower * torque`, R will automatically include `horsepower` and `torque` in the model, then we do not need to include them manually.

```r
M2 <- lm(highwayMPG~IDYear+horsepower*torque+height+length+width, data = data)
summary(M2)
```

```
Call:
lm(formula = highwayMPG ~ IDYear + horsepower * torque + height +
    length + width, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-11.109  -2.313  -0.258   2.062 203.540

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)        4.219e+01  7.930e-01  53.199  < 2e-16 ***
IDYear2010        -5.628e-01  6.372e-01  -0.883   0.3771
IDYear2011         7.254e-02  6.361e-01   0.114   0.9092
IDYear2012         1.197e+00  6.411e-01   1.867   0.0619 .
horsepower        -1.666e-02  2.539e-03  -6.563 5.84e-11 ***
torque            -8.606e-02  2.533e-03 -33.972  < 2e-16 ***
height             6.560e-03  1.070e-03   6.133 9.32e-10 ***
length             1.777e-03  8.318e-04   2.136   0.0327 *
width             -1.169e-03  8.521e-04  -1.372   0.1700
horsepower:torque  1.124e-04  4.628e-06  24.276  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.333 on 4581 degrees of freedom
Multiple R-squared:  0.4854,    Adjusted R-squared:  0.4844
F-statistic: 480.1 on 9 and 4581 DF,  p-value: < 2.2e-16
```

Before we draw the interaction plot, we can take a look at the distribution of `torque`.

```
hist(data$torque)
```

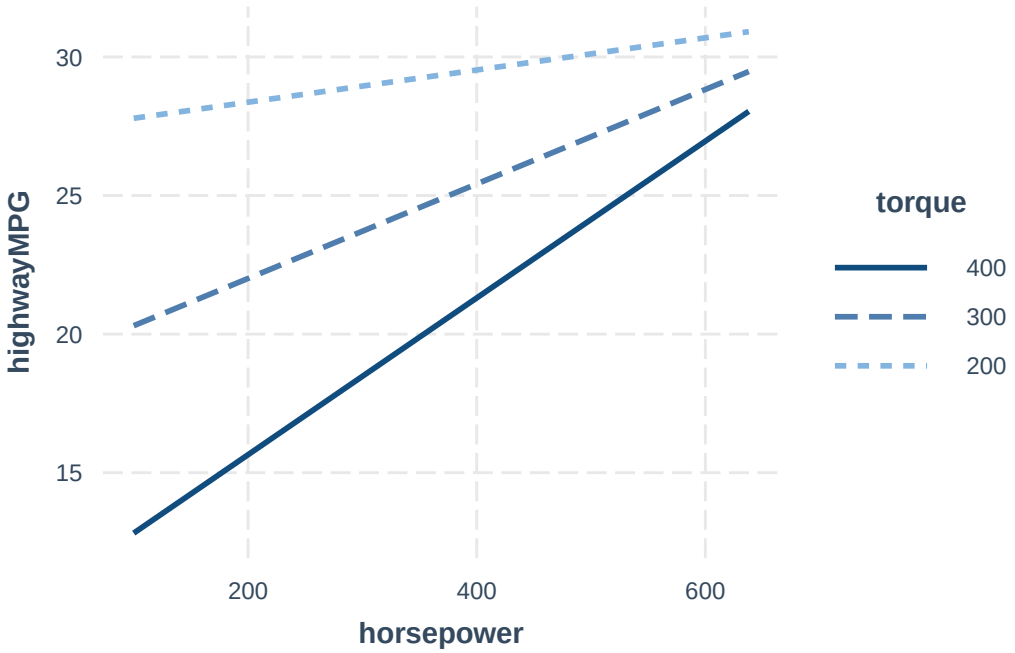


We can see that most of the observations are in the interval $[100, 500]$. So, we may choose 3 different values $(200, 300, 400)$ in this interval.

And we will use package "interactions" for the interaction plot. And for the value of "IDYear", since it is categorical and it is meaningless to define its mean, we will just pick $IDYear = 2012$.

```
library(interactions)
```

```
interact_plot(M2, pred = horsepower, modx = torque, modx.values = c(200, 300, 400),
              at = list(IDYear = as.factor(2012)))
```



Task (e)

In this task, we are to fit a model

$$highwayMPG = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4 + \hat{\beta}_5 x_5 + \hat{\beta}_6 x_1 x_2 + \hat{\beta}_7 \mathbb{I}_{2010} + \hat{\beta}_8 \mathbb{I}_{2011} + \hat{\beta}_9 \mathbb{I}_{2012}$$

Here, x_1, x_2, x_3, x_4, x_5 are horsepower, torque, height, length, and width, respectively. And $\mathbb{I}_{2010} = 1$ if this observation have $IDYear = 2010$, and $\mathbb{I}_{2010} = 0$ otherwise.

Then, we need to construct a vector for response and a matrix for predictors.

```
# Construct the data matrix
y <- data$highwayMPG
X <- data.frame(horsepower = data$horsepower, torque = data$torque,
                height = data$height, length = data$length , width = data$width)
X$intersection <- data$horsepower*data$torque

# Set the categorical variable
X$ID2010 <- 0
X$ID2011 <- 0
X$ID2012 <- 0
```

```r
X$ID2010[which(data$IDYear == 2010)] <- 1
X$ID2011[which(data$IDYear == 2011)] <- 1
X$ID2012[which(data$IDYear == 2012)] <- 1

# Set a column for the intercept
X$intercept <- 1
```

After constructing data matrix, we can apply the formula

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

```r
# Get prepared for matrix manipulation
X <- as.matrix(X)
y <- as.matrix(y)

# Use the formula
solve(t(X)%*%X)%*%t(X)%*%y
```

```
                    [,1]
horsepower   -0.0166633227
torque       -0.0860592704
height        0.0065603903
length        0.0017767232
width        -0.0011694485
intersection  0.0001123567
ID2010       -0.5627857770
ID2011        0.0725356431
ID2012        1.1970329986
intercept    42.1879478687
```

We can see our result is identical to the result we obtained form `lm()` numerically.

Problem 3

Task (a)

First, we read the data.

```
clear
import delimited "C:\Users\hzhaoar\Downloads\cars.csv"
```

And we rename each column

```
rename dimensionsheight height
rename dimensionslength length
rename dimensionswidth width
rename engineinformationdriveline driveLine
rename engineinformationenginetype engineType
rename engineinformationhybrid isHybrid
rename engineinformationnumberofforward numGears
rename engineinformationtransmission transmission
rename fuelinformationcitympg cityMPG
rename fuelinformationfueltype fuelType
rename fuelinformationhighwaympg highwayMPG
rename identificationclassification classification
rename identificationid ID
rename identificationmake make
rename identificationmodelyear modelYear
rename identificationyear IDyear
rename engineinformationenginestatistic horsepower
rename v18 torque
```

Task (b)

Restrict the data to cars whose Fuel Type is "Gasoline"

```
keep if fuelType == "Gasoline"
```

Task (c)

Fit a linear regression model predicting MPG on the highway.

```
regress highwayMPG horsepower torque height length width i.IDyear
```

Here is the result.

```
      Source |       SS           df       MS      Number of obs   =     4,591
-------------+----------------------------------   F(8, 4582)      =    413.35
       Model |  70043.6695         8  8755.45869   Prob > F        =    0.0000
    Residual |   97055.298     4,582  21.1818634   R-squared       =    0.4192
-------------+----------------------------------   Adj R-squared   =    0.4182
       Total |  167098.968     4,590  36.4050038   Root MSE        =    4.6024

------------------------------------------------------------------------------
  highwayMPG | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
  horsepower |   .0163556   .0022772     7.18   0.000     .0118913      .02082
      torque |  -.0507425    .002203   -23.03   0.000    -.0550614   -.0464236
      height |   .0099079   .0011267     8.79   0.000      .007699    .0121168
      length |    .001729   .0008836     1.96   0.050    -3.36e-06    .0034613
       width |  -.0003343   .0009045    -0.37   0.712    -.0021075    .0014388
             |
      IDyear |
        2010 |  -.4539681   .6768246    -0.67   0.502     -1.78087    .8729342
        2011 |   .1711016   .6757043     0.25   0.800    -1.153604    1.495808
        2012 |   1.302928   .6810076     1.91   0.056    -.0321751    2.638031
             |
       _cons |   32.29266   .7225982    44.69   0.000     30.87602     33.7093
------------------------------------------------------------------------------
```

We can see it is identical to the result we have obtained using R.

Task (d)

Refit the model by adding the interaction between horsepower and torque.

```
regress highwayMPG c.horsepower##c.torque height length width i.IDyear
```

Here is the result.

```
      Source |       SS           df       MS      Number of obs   =     4,591
-------------+----------------------------------   F(9, 4581)      =    480.07
       Model |  81105.8715         9  9011.76351   Prob > F        =    0.0000
    Residual |  85993.096     4,581  18.7716865   R-squared       =    0.4854
-------------+----------------------------------   Adj R-squared   =    0.4844
       Total |  167098.968     4,590  36.4050038   Root MSE        =    4.3326
```

```
------------------------------------------------------------------------------
  highwayMPG | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
  horsepower |  -.0166633   .0025388    -6.56   0.000    -.0216406    -.011686
      torque |  -.0860593   .0025333   -33.97   0.000    -.0910257   -.0810928
             |
c.horsepower#|
    c.torque |   .0001124   4.63e-06    24.28   0.000     .0001033    .0001214
             |
      height |   .0065604   .0010696     6.13   0.000     .0044634    .0086573
      length |   .0017767   .0008318     2.14   0.033     .0001459    .0034075
       width |  -.0011694   .0008521    -1.37   0.170      -.00284    .0005011
             |
      IDyear |
        2010 |  -.5627858   .6371716    -0.88   0.377    -1.811949    .6863777
        2011 |   .0725356   .6361142     0.11   0.909    -1.174555    1.319626
        2012 |   1.197033   .6411085     1.87   0.062    -.0598488    2.453915
             |
       _cons |   42.18795   .7930274    53.20   0.000     40.63323    43.74266
------------------------------------------------------------------------------
```

We can see this is also identical to the result we have obtained using R.

Then, we can use `margins` and `marginsplot` to create the interaction plot.

```
margins, at(torque=(200,300,400) horsepower=(200,300,400,500,600) IDyear=2012)
marginsplot, xdim(horsepower)
```

Here is the interaction plot.

We can see the three lines we obtained using stata is the same as that we obtained using R.

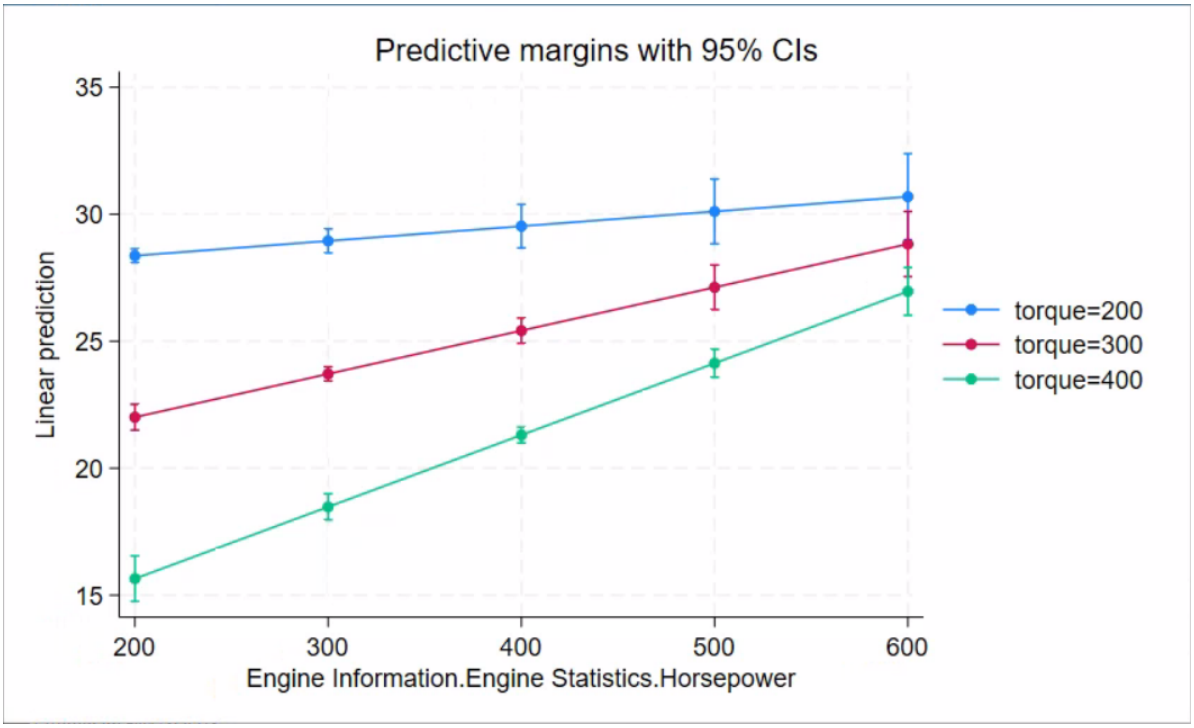


Figure 1: Interaction plot