506 Problem Set 2

Xiaohan Liu

Table of contents

GitHub repository: https://github.com/EmiiilyLiu/STATS_506

Problem 1

(a)

Version 1:

```r
#' @param n an integer
#' @return the total winnings or loses of 'n'
play_dice_1 <- function(n){
  total = -2*n
  ## use a loop to iterate each roll
  ## add up each wining or lose
  for (i in 1:n){
    result <- sample(1:6,1)
    if(result%%2==0){total = total + result}
    else {total}
  }

  return(total)
}
```

Version 2:

```r
#' @param n an integer
#' @return the total winnings or loses of 'n'
play_dice_2 <- function(n){
  ## use built-in R vectorized functions
  ## get the result of each roll
  result <- sample(1:6, n, replace=TRUE)

  ## then get the corresponding scores
  score <- ifelse(result%%2==0, result-2, -2)

  ## finally sum the scores
  total <- sum(score)
  return(total)
}
```

Version 3:

```r
#' @param n an integer
#' @return the total winnings or loses of 'n'
play_dice_3 <- function(n){
  result <- sample(1:6, n, replace=TRUE)

  ## use factor function to ensure all levels are represented
  roll_factor <- factor(result, levels=1:6)

```

```r
  ## table to store the number of occurrence of each level
  table <- table(roll_factor)

  total <- sum(ifelse(as.numeric(names(table))%%2==0,
                      as.numeric(names(table))-2,-2)*table)
  return(total)
}
```

Version 4:

```r
#' @param n an integer
#' @return the total winnings or loses of 'n'
play_dice_4 <- function(n){
  result <- sample(1:6, n, replace=TRUE)

  ## use sapply function to generate
  ## wining or lose in each roll
  score <- sapply(result,
                  function(result)ifelse(result%%2==0, result-2, -2))

  total <- sum(score)
  return(total)
}
```

(b)

```r
play_dice_1(3)
```

```
[1] 4
```

```r
play_dice_1(3000)
```

```
[1] -4
```

```r
play_dice_2(3)
```

```
[1] 0
```

```r
play_dice_2(3000)
```

```
[1] -70
```

```r
play_dice_3(3)
```

```
[1] 4
```

```r
play_dice_3(3000)
```

```
[1] 48
```

```r
play_dice_4(3)
```

```
[1] -6
```

```r
play_dice_4(3000)
```

```
[1] 134
```

(c)

To get the same result of each version, we modify the functions to add a *seed* parameter. Inputting the same *seed* value when calling functions can make sure each version generating the same random values, so that we can compare their result.

Version 1:

```r
#' @param n an integer
#' @param seed for the generation of random values
#' @return the total winnings or loses of 'n'
play_dice_1 <- function(n, seed=NULL){
  if(!is.null(seed)) {set.seed(seed)}

  total = -2*n
```

```r
  ## use a loop to iterate each roll
  ## add up each wining or lose
  for (i in 1:n){
    result <- sample(1:6,1)
    if(result%%2==0){total = total + result}
    else {total}
  }

  return(total)
}
```

Version 2:

```r
#' @param n an integer
#' @param seed for the generation of random values
#' @return the total winnings or loses of 'n'
play_dice_2 <- function(n, seed=NULL){
  if(!is.null(seed)) {set.seed(seed)}

  ## use built-in R vectorized functions
  ## get the result of each roll
  result <- sample(1:6, n, replace=TRUE)

  ## then get the corresponding scores
  score <- ifelse(result%%2==0, result-2, -2)

  ## finally sum the scores
  total <- sum(score)
  return(total)
}
```

Version 3:

```r
#' @param n an integer
#' @param seed for the generation of random values
#' @return the total winnings or loses of 'n'
play_dice_3 <- function(n, seed=NULL){
  if(!is.null(seed)) {set.seed(seed)}

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

  ## use factor function to ensure all levels are represented
```

```r
  roll_factor <- factor(result, levels=1:6)

  ## table to store the number of occurrence of each level
  table <- table(roll_factor)

  total <- sum(ifelse(as.numeric(names(table))%%2==0,
                      as.numeric(names(table))-2,-2)*table)
  return(total)
}
```

Version 4:

```r
#' @param n an integer
#' @param seed for the generation of random values
#' @return the total winnings or loses of 'n'
play_dice_4 <- function(n, seed=NULL){
  if(!is.null(seed)) {set.seed(seed)}

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

  ## use sapply function to generate
  ## wining or lose in each roll
  score <- sapply(result,
                  function(result)ifelse(result%%2==0, result-2, -2))

  total <- sum(score)
  return(total)
}
```

```r
play_dice_1(3, seed=123)
```

```
[1] 0
```

```r
play_dice_2(3, seed=123)
```

```
[1] 0
```

```r
play_dice_3(3, seed=123)
```

```
[1] 0
```

```r
play_dice_4(3, seed=123)
```

```
[1] 0
```

```r
play_dice_1(3000, seed=123)
```

```
[1] -102
```

```r
play_dice_2(3000, seed=123)
```

```
[1] -102
```

```r
play_dice_3(3000, seed=123)
```

```
[1] -102
```

```r
play_dice_4(3000, seed=123)
```

```
[1] -102
```

```r
## Check the results are equal
(play_dice_1(3, seed=123)==play_dice_2(3, seed=123)) &&
(play_dice_1(3, seed=123)==play_dice_3(3, seed=123)) &&
(play_dice_1(3, seed=123)==play_dice_4(3, seed=123))
```

```
[1] TRUE
```

```r
(play_dice_1(3000, seed=123)==play_dice_2(3000, seed=123)) &&
(play_dice_1(3000, seed=123)==play_dice_3(3000, seed=123)) &&
(play_dice_1(3000, seed=123)==play_dice_4(3000, seed=123))
```

```
[1] TRUE
```

(d)

```r
# Library for timing comparison
library(microbenchmark)

microbenchmark(
  play_dice_1(100, seed = 0),
  play_dice_2(100, seed = 0),
  play_dice_3(100, seed = 0),
  play_dice_4(100, seed = 0),
  play_dice_1(10000, seed = 0),
  play_dice_2(10000, seed = 0),
  play_dice_3(10000, seed = 0),
  play_dice_4(10000, seed = 0)
)
```

```
Unit: microseconds
                         expr     min       lq       mean    median        uq
   play_dice_1(100, seed = 0)   499.5   619.55   1082.332    747.90   1442.65
   play_dice_2(100, seed = 0)    27.9    47.40     85.782     69.15    111.60
   play_dice_3(100, seed = 0)   108.5   142.45    333.639    231.65    362.65
   play_dice_4(100, seed = 0)   227.1   269.85    540.615    362.60    697.55
 play_dice_1(10000, seed = 0) 60230.0 69755.00 117604.159 103305.90 162267.40
 play_dice_2(10000, seed = 0)   867.4  1033.40   1745.748   1337.90   2365.85
 play_dice_3(10000, seed = 0)  1147.9  1417.35   2441.984   1759.00   3242.60
 play_dice_4(10000, seed = 0) 21178.7 26700.70  45802.127  38513.05  63518.80
      max neval
   3073.3   100
    371.5   100
   6669.5   100
   1363.9   100
 212859.0   100
   3454.7   100
   5052.3   100
  93108.9   100
```

Discuss:
1. No matter small or large input, Version 1 has the slowest speed and the running time increases dramatically with the input value increasing.
2. The speed of Version 4 is higher than that of Version 1, but is still lower than that of Version 2 and Version 3. Its running time also shows a significant increase when the input value increases.
3. Version 2 is the most efficient one no matter small or large input. 4. Version 3 is more

efficient than Version 1 and Version 4, but spends longer running time than Version 2. *Conclusion*: Vectorization in R demonstrates higher efficiency especially with large input or repeatedly calculation, while loop runs considerably slowly.

(e)

```
MC_simulation <- function(num_simulation, num_roll){
  simulation_result <- replicate(num_simulation, play_dice_2(num_roll))
  avg_score <- mean(simulation_result)
  return(avg_score)
}
num_simulation = 1e6
num_roll = 1
MC_simulation(num_simulation, num_roll)
```

```
[1] -0.00133
```

In theory, the expected wining for this game is $\frac{1}{6}*((-2)+(2-2)+(-2)+(4-2)+(-2)+(6-2)) = 0$, it is a fair game. By Monte Carlo simulation, the expectation of the game is 0, proving the answer.

Problem 2

```
data <- read.csv("F:/Desktop/STATS 506/STATS_506/cars.csv")
```

(a)

```
colnames(data)
```

```
 [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"
```

```r
colnames(data) <- c("Height", "Length", "Width", "Driveline",
                    "EngineType", "Hybrid", "GearsNum",
                    "Transmission", "CityMPG", "FuelType",
                    "HighwayMPG", "Classification", "ID", "Make",
                    "ModelYear", "ReleaseYear", "Horsepower", "Torque")
```

(b)

```r
## New dataset only with FuelType is Gasoline
gasoline_cars <- subset(data, FuelType=='Gasoline')
```

(c)

```r
## Linear model
model <- lm(HighwayMPG ~ Horsepower +  Torque + Height + Length +
              Width + as.factor(ReleaseYear), data = gasoline_cars)
summary(model)
```

```
Call:
lm(formula = HighwayMPG ~ Horsepower + Torque + Height + Length +
    Width + as.factor(ReleaseYear), data = gasoline_cars)

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 ***
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
as.factor(ReleaseYear)2010 -0.4539681  0.6768246  -0.671   0.5024
as.factor(ReleaseYear)2011  0.1711016  0.6757043   0.253   0.8001
as.factor(ReleaseYear)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
```

Controlling all other predictors constant, 1 unit increase in *Horsepower* will cause 0.0164 unit increase in *HighwayMPG*. The $p - value$ is $7.96e - 13 << 0.05$, so that the coefficient is statistically significant.

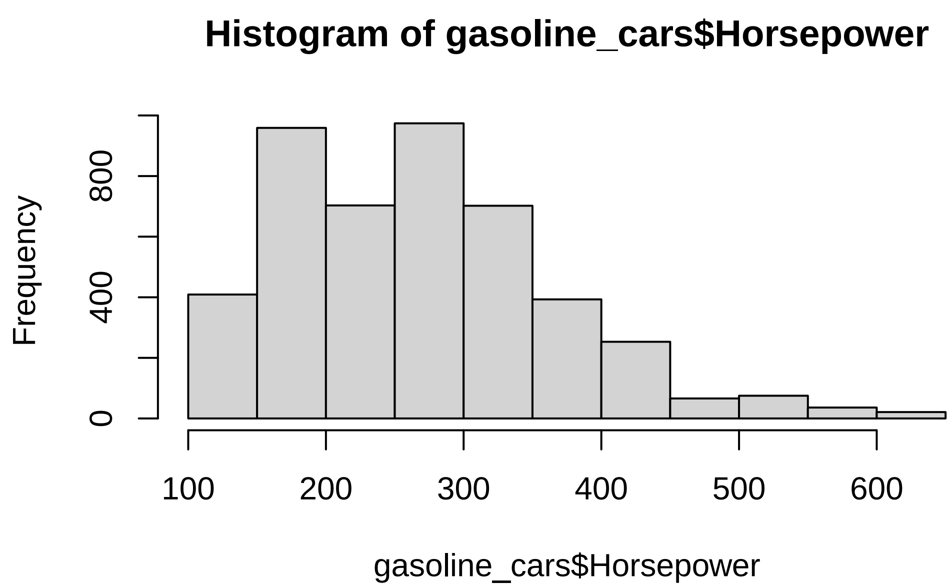
(d)

```r
## Linear model with interaction term
interact_model <- lm(HighwayMPG ~ Horsepower*Torque + Height +
                          Length + Width + as.factor(ReleaseYear),
                        data = gasoline_cars)
summary(interact_model)
```

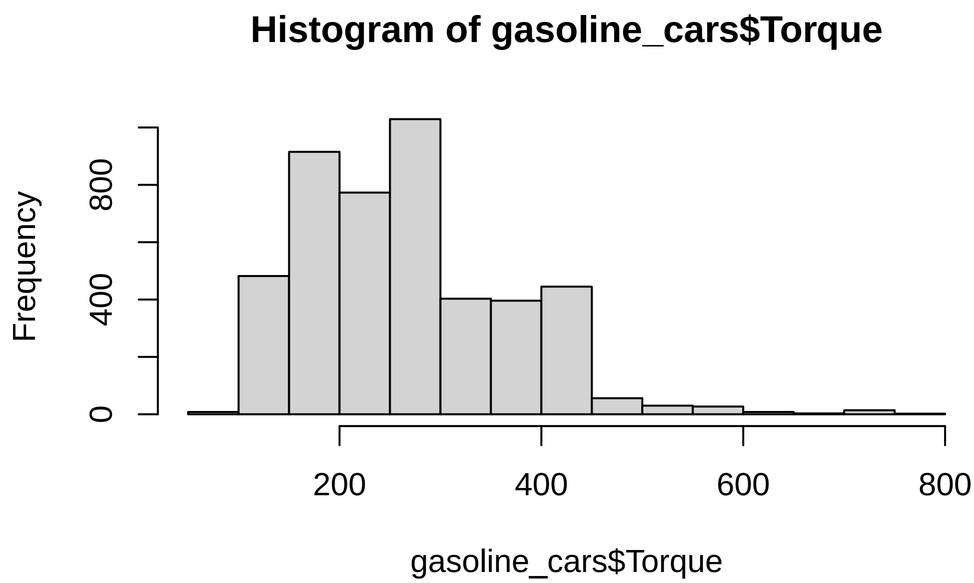
```
Call:
lm(formula = HighwayMPG ~ Horsepower * Torque + Height + Length +
    Width + as.factor(ReleaseYear), data = gasoline_cars)

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 ***
```

```
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
as.factor(ReleaseYear)2010 -5.628e-01  6.372e-01  -0.883   0.3771
as.factor(ReleaseYear)2011  7.254e-02  6.361e-01   0.114   0.9092
as.factor(ReleaseYear)2012  1.197e+00  6.411e-01   1.867   0.0619 .
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
```

```r
library(interactions)

## choose reasonable values of horsepower
hist(gasoline_cars$Horsepower)
```



```r
horsepower_values <- c(150, 200, 250, 300, 350, 400)

## choose reasonable values of torque
hist(gasoline_cars$Torque)
```

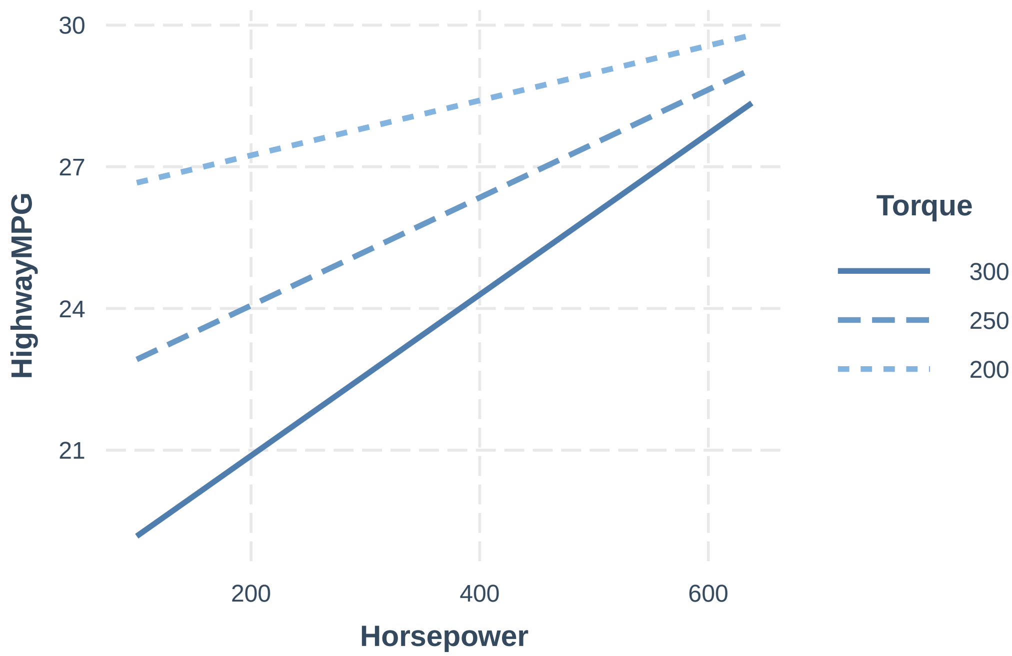



```r
torque_values <- c(200, 250, 300)

## choose a reasonable year
table(gasoline_cars$ReleaseYear)
```

```
2009 2010 2011 2012
  48 1633 1794 1116
```

```r
interact_plot(interact_model, pred = Horsepower, modx = Torque,
              at = list(Horsepower = horsepower_values, ReleaseYear = 2011),
              modx.values = torque_values)
```

```
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.
```



(e)

```
form <- HighwayMPG ~ Horsepower*Torque + Height +
                     Length + Width + as.factor(ReleaseYear)
X<-model.matrix(form, data = gasoline_cars)
y <- gasoline_cars$HighwayMPG
beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y

## results are equal to linear model in 2(d)
beta_hat
```

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

```
Horsepower:Torque           0.0001123567
```

```
coef(interact_model)
```

```
              (Intercept)                Horsepower
            42.1879478687             -0.0166633227
                   Torque                    Height
            -0.0860592704              0.0065603903
                   Length                     Width
             0.0017767232             -0.0011694485
as.factor(ReleaseYear)2010 as.factor(ReleaseYear)2011
            -0.5627857770              0.0725356431
as.factor(ReleaseYear)2012         Horsepower:Torque
             1.1970329986              0.0001123567
```

Problem 3

```
. do "C:\Users\xhliuu\Documents\PS2 Q3.do"

. * Import data
. import delimited "K:\cars.csv"
(encoding automatically selected: ISO-8859-1)
(18 vars, 5,076 obs)

.
```

(a)

```
. * (a) Rename columns
. rename dimensionsheight Height

. rename dimensio~gth Length

. rename dimensio~dth Width

. rename engineinf~ne Driveline

. rename engineinf~pe EngineType
```

```
. rename engineinf~id Hybrid

. rename engineinf~rd GearsNum

. rename engineinfo~n Transmission

. rename fuelin~tympg CityMPG

. rename fuelinform~e FuelType

. rename fuelin~aympg HighwayMPG

. rename identifica~n Classification

. rename identifica~d ID

. rename identifica~e Make

. rename identi~lyear ModelYear

. rename identi~nyear ReleaseYear

. rename engineinfo~c Horsepower

. rename v18 Torque
```

(b)

```
. * (b) Restrict the data to cars whose Fuel Type is "Gasoline"
. keep if FuelType == "Gasoline"
(485 observations deleted)
```

(c)

```
.
. * (c)
. regress HighwayMPG Horsepower Torque Height Length Width i.ReleaseYear

      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
             |
 ReleaseYear |
       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
------------------------------------------------------------------------------
```

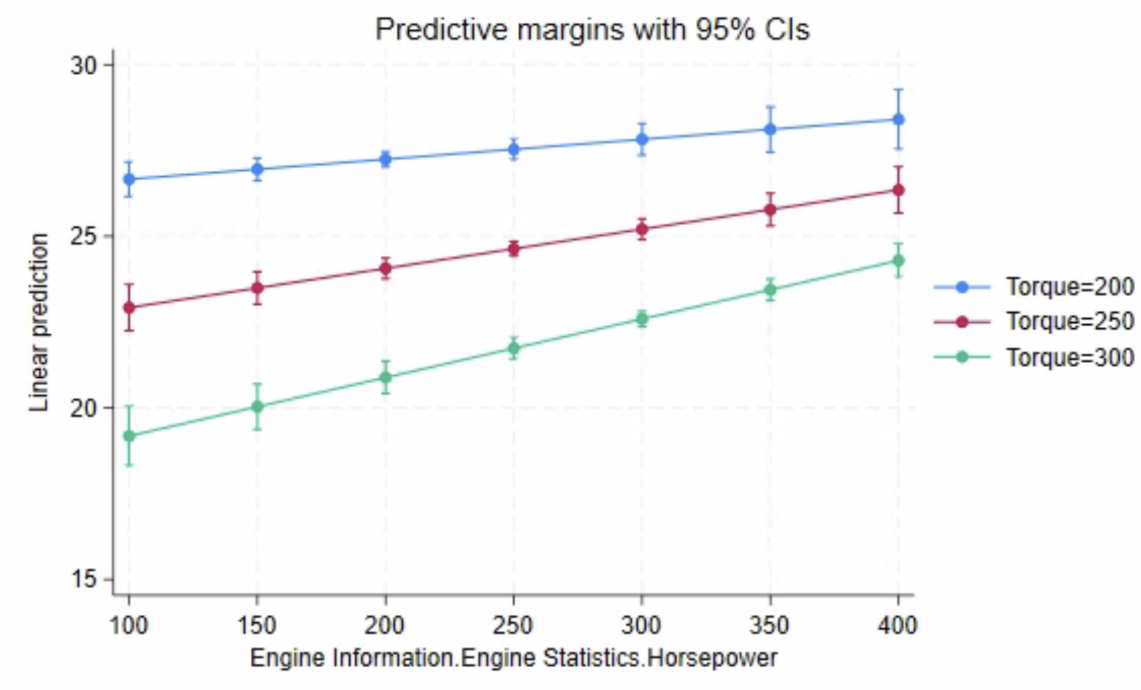
(d)

```
.
. * (d)
. regress HighwayMPG c.Horsepower##c.Torque Height Length Width i.ReleaseYear

      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
                     |
         ReleaseYear |
               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
------------------------------------------------------------------------------------------

.
. margins, at(Torque=(200(50)300) Horsepower=(100(50)400) ReleaseYear=2011)

Predictive margins                                          Number of obs = 4,591
Model VCE: OLS

Expression: Linear prediction, predict()
1._at:  Horsepower  =  100
        Torque      =  200
        ReleaseYear = 2011
2._at:  Horsepower  =  100
        Torque      =  250
        ReleaseYear = 2011
3._at:  Horsepower  =  100
        Torque      =  300
        ReleaseYear = 2011
4._at:  Horsepower  =  150
        Torque      =  200
        ReleaseYear = 2011
5._at:  Horsepower  =  150
        Torque      =  250
        ReleaseYear = 2011
6._at:  Horsepower  =  150
        Torque      =  300
        ReleaseYear = 2011
```

```
7._at:  Horsepower  =  200
        Torque      =  200
        ReleaseYear = 2011
8._at:  Horsepower  =  200
        Torque      =  250
        ReleaseYear = 2011
9._at:  Horsepower  =  200
        Torque      =  300
        ReleaseYear = 2011
10._at: Horsepower  =  250
        Torque      =  200
        ReleaseYear = 2011
11._at: Horsepower  =  250
        Torque      =  250
        ReleaseYear = 2011
12._at: Horsepower  =  250
        Torque      =  300
        ReleaseYear = 2011
13._at: Horsepower  =  300
        Torque      =  200
        ReleaseYear = 2011
14._at: Horsepower  =  300
        Torque      =  250
        ReleaseYear = 2011
15._at: Horsepower  =  300
        Torque      =  300
        ReleaseYear = 2011
16._at: Horsepower  =  350
        Torque      =  200
        ReleaseYear = 2011
17._at: Horsepower  =  350
        Torque      =  250
        ReleaseYear = 2011
18._at: Horsepower  =  350
        Torque      =  300
        ReleaseYear = 2011
19._at: Horsepower  =  400
        Torque      =  200
        ReleaseYear = 2011
20._at: Horsepower  =  400
        Torque      =  250
        ReleaseYear = 2011
```

```
21._at: Horsepower  =  400
        Torque      =  300
        ReleaseYear = 2011

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
         _at |
          1  |   26.66169   .2564192   103.98   0.000     26.15898    27.16439
          2  |   22.92051   .3407517    67.26   0.000     22.25247    23.58854
          3  |   19.17933   .4392009    43.67   0.000     18.31828    20.04037
          4  |   26.95209   .1653544   163.00   0.000     26.62791    27.27626
          5  |    23.4918   .2413064    97.35   0.000     23.01872    23.96488
          6  |   20.03151    .336916    59.46   0.000     19.37099    20.69203
          7  |   27.24249   .1133793   240.28   0.000     27.02021    27.46477
          8  |   24.06309   .1533936   156.87   0.000     23.76237    24.36382
          9  |    20.8837   .2389747    87.39   0.000     20.41519     21.3522
         10  |   27.53289   .1492954   184.42   0.000      27.2402    27.82558
         11  |   24.63439   .1096092   224.75   0.000      24.4195    24.84927
         12  |   21.73588   .1539059   141.23   0.000     21.43415    22.03761
         13  |   27.82329   .2358917   117.95   0.000     27.36083    28.28575
         14  |   25.20568   .1538913   163.79   0.000     24.90398    25.50738
         15  |   22.58806   .1150943   196.26   0.000     22.36242     22.8137
         16  |   28.11369   .3360389    83.66   0.000     27.45489    28.77249
         17  |   25.77697   .2419393   106.54   0.000     25.30265    26.25129
         18  |   23.44025   .1605863   145.97   0.000     23.12542    23.75508
         19  |   28.40409   .4405915    64.47   0.000     27.54032    29.26786
         20  |   26.34826   .3414242    77.17   0.000     25.67891    27.01762
         21  |   24.29243   .2476102    98.11   0.000       23.807    24.77787
------------------------------------------------------------------------------

.
. marginsplot, xdim(Horsepower)

Variables that uniquely identify margins: Torque Horsepower
```



(e)

```
. * (e)
. gen constant = 1

. gen interact_col = Horsepower*Torque

. gen Year_2010 = (ReleaseYear == 2010)

. gen Year_2011 = (ReleaseYear == 2011)

. gen Year_2012 = (ReleaseYear == 2012)

. mkmat constant interact_col Horsepower Torque Height Length Width Year_2010 Year_2011
> Year_2012, matrix(X)

. mkmat HighwayMPG, matrix(y)

. matrix beta_hat = invsym(X'*X)*(X'*y)
```

```
. matrix list beta_hat

beta_hat[10,1]
               HighwayMPG
    constant    42.187948
interact_col    .00011236
  Horsepower   -.01666332
      Torque   -.08605927
      Height    .00656039
      Length    .00177672
       Width   -.00116945
   Year_2010   -.56278578
   Year_2011    .07253564
   Year_2012     1.197033

.
.
.
.
.
.
.
.
.
.
.
.
end of do-file
```