hw2

1. Dice Game

   a. Game Implementations

      i. Version 1

         ::: {.cell}

```
#' Dice Game V1
#'
#' @param num_dice number of dice to roll
#' @return net profit
dice_game_v1 <- function(num_dice){
    net_profit <- 0
    for (i in 1:num_dice){
        die_roll <- sample(1:6, 1)
        if (die_roll %in% c(2,4,6)){
        net_profit <- net_profit + 2
        } else {
        net_profit <- net_profit - 2
        }
    }
    return(net_profit)
}
```

         :::

      ii. Version 2

         ::: {.cell}

```
#' Dice Game V2
#'
#' @param num_dice number of dice to roll
#' @return net profit
dice_game_v2 <- function(num_dice){
```

```r
    die_rolls <- sample(1:6, num_dice, replace = TRUE)
    net_profit <- sum(ifelse(die_rolls %in% c(2,4,6), 2, -2))
    return(net_profit)
}
```

:::

iii. Version 3

::: {.cell}

```r
#' Dice Game V3
#'
#' @param num_dice number of dice to roll
#' @return net profit
dice_game_v3 <- function(num_dice) {
    die_rolls <- sample(1:6, num_dice, replace = TRUE)  # Simulate dice rolls
    roll_counts <- table(die_rolls)  # Count the occurrences of each roll

    # Ensure all possible outcomes (1 to 6) have counts, replace missing ones
    roll_counts <- ifelse(!1:6 %in% names(roll_counts), 0, roll_counts)

    # Calculate net profit based on the counts using table()
    net_profit <- (roll_counts[2] + roll_counts[4] + roll_counts[6]) * 2
    net_profit <- net_profit - (roll_counts[1] + roll_counts[3] + roll_counts

    return(net_profit)
}
```

:::

iv. Version 4 (Implement this game by using one of the “apply” functions.)

::: {.cell}

```r
#' Dice Game V4
#'
#' @param num_dice number of dice to roll
#' @return net profit
dice_game_v4 <- function(num_dice) {
    die_rolls <- sample(1:6, num_dice, replace = TRUE)  # Simulate dice rolls

    # Define a function to calculate profit/loss for a single roll
    calculate_profit <- function(roll) {
        if (roll %in% c(2, 4, 6)) {
```

```
            return(2)
            } else {
            return(-2)
            }
        }

        # Use sapply to apply the calculate_profit function to each roll
        profits <- sapply(die_rolls, calculate_profit)

        # Calculate the total net profit by summing up the profits
        net_profit <- sum(profits)

        return(net_profit)
    }
```

:::

b. Verfiy

   i. Version 1

      ::: {.cell}

```
num_dice <- 3
print(dice_game_v1(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] 2` :::

```
num_dice <- 3000
print(dice_game_v1(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] -68` ::: :::

   ii. Version 2

      ::: {.cell}

```
num_dice <- 3
print(dice_game_v2(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] 6` :::

```
num_dice <- 3000
print(dice_game_v2(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] -92` ::: :::

iii. Version 3

::: {.cell}

```
num_dice <- 3
print(dice_game_v3(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] 2` :::

```
num_dice <- 3000
print(dice_game_v3(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] 136` ::: :::

iv. Version 4

::: {.cell}

```
num_dice <- 3
print(dice_game_v4(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] -6` :::

```
num_dice <- 3000
print(dice_game_v4(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] 104` ::: :::

c. Same Results

i. Version 1

::: {.cell}

```
set.seed(1234)
num_dice <- 3
print(dice_game_v1(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] 6` :::

```
num_dice <- 3000
print(dice_game_v1(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] -28` ::: :::

ii. Version 2

::: {.cell}

```
set.seed(1234)
num_dice <- 3
print(dice_game_v2(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] 6` :::

```
num_dice <- 3000
print(dice_game_v2(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] -28` ::: :::

iii. Version 3

::: {.cell}

```
set.seed(1234)
num_dice <- 3
print(dice_game_v3(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] 6` :::

```
num_dice <- 3000
print(dice_game_v3(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] -28` ::: :::

iv. Version 4

::: {.cell}

```
set.seed(1234)
num_dice <- 3
print(dice_game_v4(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] 6` :::

```
num_dice <- 3000
print(dice_game_v4(num_dice))
```

::: {.cell-output .cell-output-stdout} `[1] -28` ::: :::

d. Benchmark

i. Version 1

::: {.cell}

```r
library(microbenchmark)
num_dice <- 100
microbenchmark(dice_game_v1(num_dice))
```

::: {.cell-output .cell-output-stderr} `Warning in microbenchmark(dice_game_v1(num_dice)): less accurate nanosecond  times to avoid potential integer overflows` :::

::: {.cell-output .cell-output-stdout} `Unit: microseconds                    expr min       lq     mean  median       uq      max  dice_game_v1(num_dice) 310.78 331.7105 374.1607 343.498 357.4995 2850.238   neval 100` :::

```r
num_dice <- 10000
microbenchmark(dice_game_v1(num_dice))
```

::: {.cell-output .cell-output-stdout} `Unit: milliseconds                    expr min       lq     mean   median      uq      max  dice_game_v1(num_dice) 35.41851 36.52237 38.96493 37.86973 39.8259 60.36012   neval 100` ::: :::

ii. Version 2

::: {.cell}

```r
library(microbenchmark)
num_dice <- 100
microbenchmark(dice_game_v2(num_dice))
```

::: {.cell-output .cell-output-stdout} `Unit: microseconds                    expr min     lq     mean median     uq    max neval  dice_game_v2(num_dice) 11.111 12.464 14.81986 12.956 13.776 53.997   100` :::

```r
num_dice <- 10000
microbenchmark(dice_game_v2(num_dice))
```

::: {.cell-output .cell-output-stdout} `Unit: microseconds                    expr min      lq     mean  median       uq      max  dice_game_v2(num_dice) 560.716 584.824 632.3036 595.443 611.4945 3902.995   neval 100` ::: :::

iii. Version 3

::: {.cell}

```r
library(microbenchmark)
num_dice <- 100
microbenchmark(dice_game_v3(num_dice))
```

::: {.cell-output .cell-output-stdout} `Unit: microseconds                              expr min     lq     mean  median      uq     max neval   dice_game_v3(num_dice) 40.959 43.46 47.16107 44.8745 48.2775 150.224   100` :::

```r
num_dice <- 10000
microbenchmark(dice_game_v3(num_dice))
```

::: {.cell-output .cell-output-stdout} `Unit: microseconds                              expr min       lq      mean    median       uq      max   dice_game_v3(num_dice) 598.723 640.6455 762.1687 654.1755 670.7395 10405.14   neval 100` ::: :::

iv. Version 4

::: {.cell}

```r
library(microbenchmark)
num_dice <- 100
microbenchmark(dice_game_v4(num_dice))
```

::: {.cell-output .cell-output-stdout} `Unit: microseconds                              expr min lq     mean median     uq     max neval   dice_game_v4(num_dice) 79.048 82 89.10243 86.223 92.414 147.477   100` :::

```r
num_dice <- 10000
microbenchmark(dice_game_v4(num_dice))
```

::: {.cell-output .cell-output-stdout} `Unit: milliseconds                              expr min       lq      mean   median       uq      max   dice_game_v4(num_dice) 7.490372 8.017079 9.106001 8.79821 9.674237 19.95605   neval 100` ::: :::

e. Monte Carlo Simulation

::: {.cell}

```r
num_dice <- 10000
num_sims <- 10000
set.seed(1234)
results <- replicate(num_sims, dice_game_v2(num_dice))
print(mean(results))
```

::: {.cell-output .cell-output-stdout} `[1] 0.3376` ::: :::
This seems to be a fair game because the average net profit is close to 0 over 10000 simulations of 10000 dice rolls.

2. Linear Regression

   a. Renaming - Modified CSV headers in place locally

   b. Restrict to "Gasoline"

      ::: {.cell}

```
cars <- read.csv("cars.csv")
cars <- cars[cars$Fuel_Type == "Gasoline",]
cars$Year <- as.factor(cars$Year)
```

      :::

   c. Fit Linear Regression for "Highway MPG"

      ::: {.cell}

```
model <- lm(
    Highway_MPG
    ~ Horsepower
    + Torque
    + Height
    + Width
    + Length
    + Year
    , data = cars
)
print(coefficients(model)[2])
```

      ::: {.cell-output .cell-output-stdout} `Horsepower   0.01635563` ::: :::

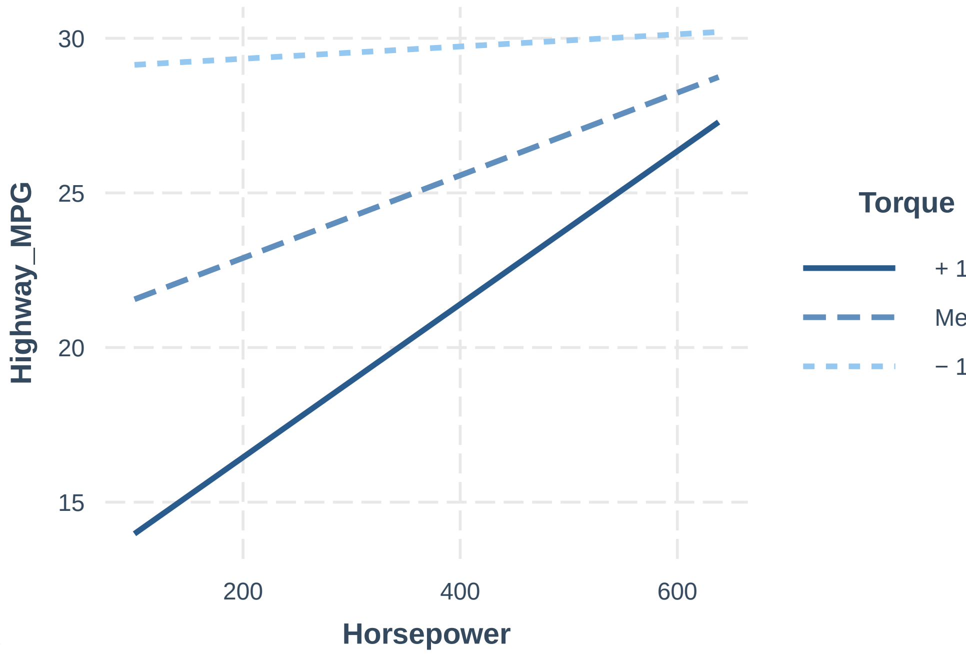
      The coeffecient for Horsepower is approximately 0.016. This means that for every 1 unit increase in Horsepower, Highway MPG increases by 0.016. The coeffecient is so low that it is almost negligible. This means that Horsepower has almost no effect on Highway MPG.

   d. Horsepower and Torque Interaction Plot

      ::: {.cell}

```
library(interactions)
model <- lm(
    Highway_MPG
    ~ Horsepower * Torque
```

```
    + Height
    + Width
    + Length
    + Year
    , data = cars
)
interact_plot(model, pred = Horsepower, modx = Torque, data = cars)
```



::: {.cell-output-display}
::: :::
For lower torque values, we see the lowest amount of increase in Highway MPG as Horsepower increases. For higher torque values, we see the largest amount of increase in Highway MPG as Horsepower increases.

e. Manually Calculate Coefficients
::: {.cell}

```
x <- model.matrix(model)
x_t <- t(x)
y <- cars$Highway_MPG
b_hat <- solve(x_t %*% x) %*% x_t %*% y
print(b_hat[, 1])
```

::: {.cell-output .cell-output-stdout} `(Intercept)         Horsepower             Torque`
`Height       42.1879478687      -0.0166633227      -0.0860592704`
`0.0065603903                  Width             Length          Year2010`

`Year2011 -0.0011694485 0.0017767232 -0.5627857770 0.0725356431 Year2012 Horsepower:Torque 1.1970329986 0.0001123567` :::

```
print(coefficients(model))
```

::: {.cell-output .cell-output-stdout} `(Intercept) Horsepower Torque Height 42.1879478687 -0.0166633227 -0.0860592704 0.0065603903 Width Length Year2010 Year2011 -0.0011694485 0.0017767232 -0.5627857770 0.0725356431 Year2012 Horsepower:Torque 1.1970329986 0.0001123567` ::: :::

The coefficients are the same as the ones calculated by lm

3. Linear Regression

   a. Renaming - Modified CSV headers in place locally

   b. Restrict to "Gasoline"

      ::: {.cell}

      ```
      import delimited "cars.csv"
      (encoding automatically selected: ISO-8859-1)
      (18 vars, 5,076 obs)

      keep if fuel_type == "Gasoline"
      (485 observations deleted)
      ```

      :::

   c. Fit Linear Regression for "Highway MPG" (output truncated for display reasons)

      ::: {.cell}

      ```
      gen model_year_numeric = .
      (4,591 missing values generated)

      regress highway_mpg horsepower torque height width length i.year
            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

      ------------------------------------------------------------------------------
      ```

```
highway_mpg | 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
       width |  -.0003343   .0009045    -0.37   0.712    -.0021075    .0014388
      length |    .001729   .0008836     1.96   0.050    -3.36e-06    .0034613
             |
        year |
       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
------------------------------------------------------------------------------
```

:::

Same answer as above. Coeffecient is approximately 0.016. This means that for every 1 unit increase in Horsepower, Highway MPG increases by 0.016.

d. Horsepower and Torque Interaction Plot

::: {.cell}

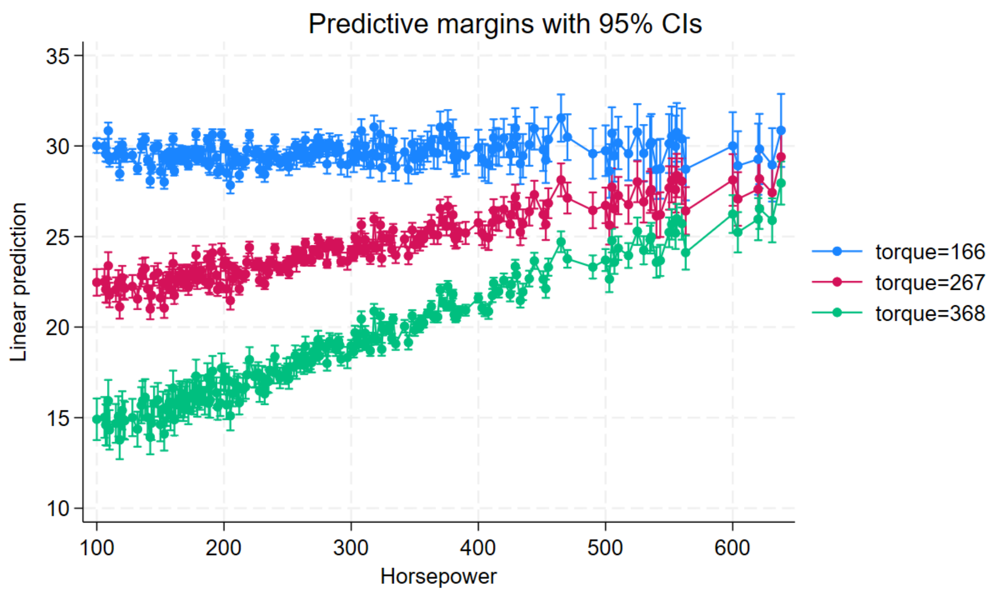
```
regress highway_mpg c.horsepower##c.torque height width length i.year

    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

------------------------------------------------------------------------------
 highway_mpg | 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
```

```
       width |  -.0011694   .0008521    -1.37   0.170     -.00284    .0005011
      length |   .0017767   .0008318     2.14   0.033    .0001459    .0034075
             |
        year |
       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=(166 267 368)) over(horsepower)
marginsplot, xdim(horsepower)
```

:::



Same answer as above for graph. For lower torque values, we see the lowest amount of increase in Highway MPG as Horsepower increases. For higher torque values, we see the largest amount of increase in Highway MPG as Horsepower increases.