STATS506-Problem Set 2

Shenyi Tang

2024-09-17

Shenyi Tang's GitHub Repo For STATS 506 FA 2024

https://github.com/shenyi-tang/stats506-computing-methods-and-tools.git

```r
library("moments")
library("microbenchmark")
library("MASS")
library("ggplot2")
library("emmeans")
```

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

```r
library("interactions")
```

Problem 1 - Dice Game

a. Version 1. Implement this game using a loop.

```r
#' @title play_dice1
#' @param n, a positive integer, refers to the number of dice rolls
#' @description implement the game using a loop.
#' In each iteration randomly generates an integer
#' between 1 and 6 (include)
#' to simulate the outcome of rolling one dice
#'
play_dice1 <- function(n) {

  if(!is.numeric(n) || n <= 0 || floor(n) != n) {
    stop("The argument should be a positive integer!")
  }

  x <- 0

  for (i in 1:n) {
```

```
    x <- x - 2
    roll <- sample(1:6, 1, replace = TRUE)
    if (roll == 3 || roll == 5) {
      x <- x + 2 * roll
    }
  }
  return(x)
}
```

a. Version 2. Implement this game using built-in R vectorized functions.

```
#' @title play_dice2
#' @param n, a positive integer, refers to the number of dice rolls
#' @description implement the game using vectorized functions
#' generates n positive integers between 1 and 6 at one time with replacement
#' to simulate the result of n dice rolls
#'
play_dice2 <- function(n) {

  if(!is.numeric(n) || n <= 0 || floor(n) != n) {
    stop("The argument should be a positive integer!")
  }

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

  winnings <- ifelse(all_roll %in% c(3,5), 2 * all_roll - 2, -2)

  return(sum(winnings))
}
```

a. Version 3. Implement this by rolling all the dice into one and collapsing the die rolls into a single `table()`. (Hint: Be careful indexing the table - what happens if you make a table of a single dice roll? You may need to look to other resources for how to solve this.)

```
#' @title play_dice3
#' @param n, a positive integer, refers to the number of dice rolls
#' @description using the table to show the frequency of n dice rolls
#'

play_dice3 <- function(n) {

  if(!is.numeric(n) || n <= 0 || floor(n) != n) {
    stop("The argument should be a positive integer!")
  }

  all_roll <- sample(1:6, n, replace = TRUE)
  tab <- table(factor(all_roll, levels = 1:6))
  tab <- as.numeric(tab)
  winnings <- tab[3] * 2 * 3 + tab[5] * 2 * 5
```

```r
  return(winnings - 2 * n)
}
```

a. Version 4: Implement this game by using one of the `apply` functions.

```r
#' @title play_dice4
#' @param n, a positive integer, refers to the number of dice rolls
#' @description using apply functions to simulate n rolls of dice
#'
play_dice4 <- function(n) {

  if(!is.numeric(n) || n <= 0 || floor(n) != n) {
    stop("The argument should be a positive integer!")
  }

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

  roll_cost <- vapply(all_roll,
                      \(x){
                        ifelse(x %in% c(3,5), 2 * x - 2, -2)
                      },
                      3)
  return(sum(roll_cost))
}
```

b. Demonstrate that all versions work. Do so by running each a few times, once with an input a 3, and once with an input of 3,000.

```r
# loop method
play_dice1(3)
## [1] -6
play_dice1(3000)
## [1] 1832
```

```r
# built-in R vectorized
play_dice2(3)
## [1] 10
play_dice2(3000)
## [1] 1728
```

```r
#table view
play_dice3(3)
## [1] -6
play_dice3(3000)
## [1] 1808
```

```r
# apply method
play_dice4(3)
## [1] -6
```

```r
play_dice4(3000)
## [1] 1832
```

c. Demonstrate that the four versions give the same result. Test with inputs 3 and 3,000. (You will need to add a way to control the randomization.)

```r
set.seed(6)
play_dice1(3)
```

```
[1] 14
```

```r
set.seed(6)
play_dice2(3)
```

```
[1] 14
```

```r
set.seed(6)
play_dice3(3)
```

```
[1] 14
```

```r
set.seed(6)
play_dice4(3)
```

```
[1] 14
```

```r
set.seed(1234)
play_dice1(300)
```

```
[1] 196
```

```r
set.seed(1234)
play_dice2(300)
```

```
[1] 196
```

```r
set.seed(1234)
play_dice3(300)
```

```
[1] 196
```

```r
set.seed(1234)
play_dice4(300)
```

```
[1] 196
```

d. Use the microbenchmark package to clearly demonstrate the speed of the implementations. Compare performance with a low input (1,000) and a large input (100,000). Discuss the results

```r
set.seed(999)
microbenchmark(
  play_dice1(1000),
  play_dice2(1000),
  play_dice3(1000),
  play_dice4(1000),
  play_dice1(100000),
  play_dice2(100000),
  play_dice3(100000),
  play_dice4(100000),
  times = 10
)
```

```
Unit: microseconds
              expr        min         lq        mean     median         uq
  play_dice1(1000)   3248.348   3343.099   3368.4288   3363.435   3397.629
  play_dice2(1000)     98.974    103.033    117.8668    114.841    129.888
  play_dice3(1000)    106.928    116.932    186.1441    179.908    229.600
  play_dice4(1000)   1983.129   2011.337   2040.9718   2038.827   2050.861
 play_dice1(1e+05) 341345.131 342511.581 347368.8100 344623.942 352417.099
 play_dice2(1e+05)   7896.764   7932.311   8080.7023   8136.676   8209.389
 play_dice3(1e+05)   6911.042   6978.979   7092.2210   7073.566   7163.725
 play_dice4(1e+05) 203436.506 206073.585 207948.2649 207873.997 210036.727
        max neval  cld
   3471.429    10 a
    147.190    10 a
    329.230    10 a
   2147.047    10 a
 358177.599    10  b
   8228.249    10   c
   7338.303    10   c
 213183.600    10    d
```

e. Do you think this is a fair game? Defend your decision with evidence based upon a Monte Carlo simulation.

```r
#' @title monte_carlo_simu
#' @param n, a positive integer, refers to the number of dice rolls
#' @param n_simu, a positive integer, refers to the number of simulations
#' @description simulate n rolls of dice for n_simu times
```

```r
monte_carlo_simu <- function(n, n_simu) {
  results <- rep(0,n_simu)
  for (i in 1:n_simu) {
    results[i] <- play_dice2(n)
  }
  return(mean(results))
}

monte_carlo_simu(100,10000)
```

```
[1] 66.6502
```

- As the average result of 10000 times simulation is bigger than zero, the game might be biased.

Problem 2 - Liner Regression

a. The names of the variables in this data are way too long. Rename the columns of the data to more reasonable lengths.

```r
cars <- read.csv2("cars.csv",
                  sep = ',')
names(cars) <- c("height", "length", "width", "driveline", "engine_type", "hybrid",
                 "n_gears", "transmission", "city_mpg", "fuel_type", "hw_mpg", "class",
                 "id", "maker", "myear", "year", "horsepower", "torque")
```

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

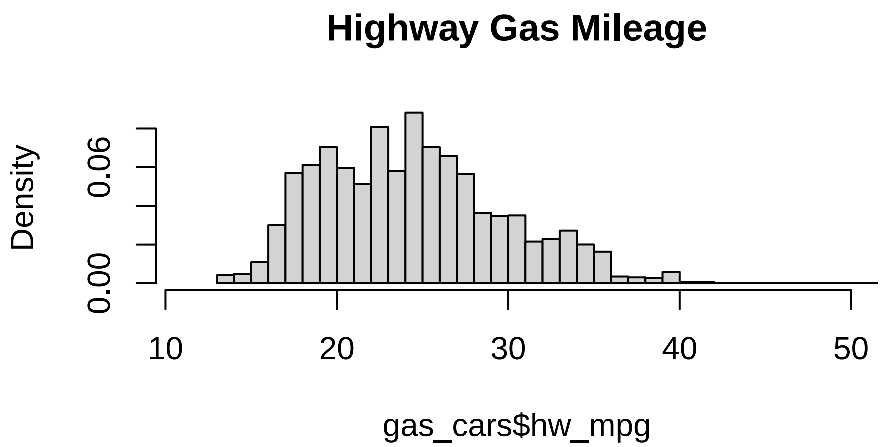
```r
gas_cars <- cars[cars$fuel_type == "Gasoline", ]
```

c. Examine the distribution of highway gas mileage. Consider whether a transformation could be used. If so, generate the transformed variable and use this variable going forward. If not, provide a short justification.

```r
# histogram for the original data
summary(gas_cars$hw_mpg)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  13.00   21.00   25.00   24.97   28.00  223.00 
```

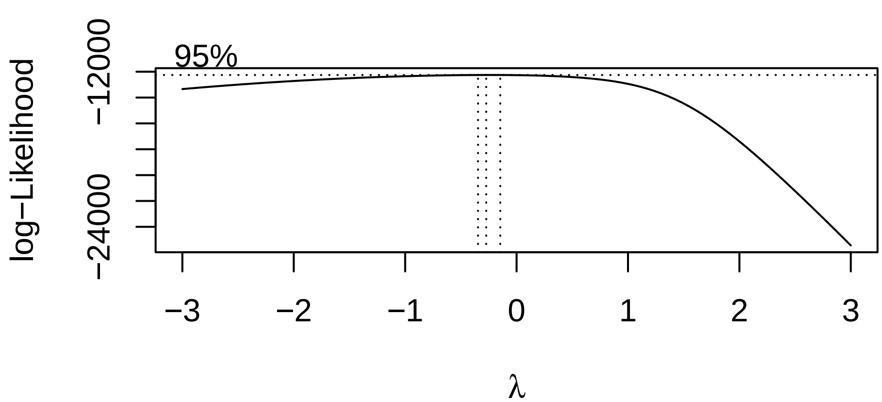
```r
hist(gas_cars$hw_mpg, breaks = 200, probability = TRUE, xlim = c(11.0, 50),
     main = "Highway Gas Mileage")
```




```r
paste("The skewness of Highway Gas Mileage: ",skewness(cars$hw_mpg))
```

```
[1] "The skewness of Highway Gas Mileage:  5.86263796206443"
```

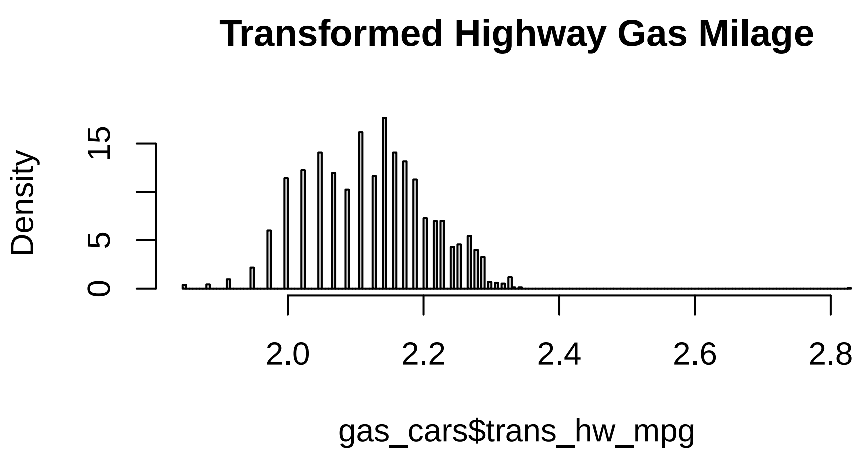
```r
# Box-Cox transformation
y <- gas_cars$hw_mpg
model <- lm(y~1)
lambda <- boxcox(model, lambda = seq(-3, 3, by = 0.1))
```




```r
best_lambda <- lambda$x[which.max(lambda$y)]
paste("The best lambda for Box-Cox Transformation: ",best_lambda)
```

```
[1] "The best lambda for Box-Cox Transformation:  -0.272727272727272"
```

```r
# histogram for the transformed data
gas_cars$trans_hw_mpg <- (gas_cars$hw_mpg^best_lambda - 1)/best_lambda
hist(gas_cars$trans_hw_mpg, breaks = 200, probability = TRUE,
     main = "Transformed Highway Gas Milage")
```



- According to the histogram of original highway gas mileage data, the original data is right-skewed. Therefore, I attempt to use Box-Cox Transformation to make the data closer to the normal distribution.

d. Fit a linear regression model predicting MPG on the highway. The predictor of interest is torque. Control for:

   - The horsepower of the engine
   - All three dimensions of the car
   - The year the car was released, as a categorical variable.

   Briefly discuss the estimated relationship between torque and highway MPG. Be precise about the interpretation of the estimated coefficient.

```r
lm.car <- lm(trans_hw_mpg ~ torque + horsepower + height + length + width + factor(year),
             data = gas_cars)
summary(lm.car)
```

```
Call:
lm(formula = trans_hw_mpg ~ torque + horsepower + height + length +
    width + factor(year), data = gas_cars)

Residuals:
     Min       1Q   Median       3Q      Max
-0.24292 -0.03788 -0.00026  0.04124  0.78574

Coefficients:
```

```
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       2.262e+00  9.146e-03 247.339  < 2e-16 ***
torque           -9.892e-04  2.788e-05 -35.474  < 2e-16 ***
horsepower        4.171e-04  2.882e-05  14.472  < 2e-16 ***
height            1.694e-04  1.426e-05  11.878  < 2e-16 ***
length            1.240e-05  1.118e-05   1.109 0.267552
width            -4.358e-05  1.145e-05  -3.807 0.000143 ***
factor(year)2010 -9.542e-03  8.567e-03  -1.114 0.265401
factor(year)2011 -1.875e-03  8.553e-03  -0.219 0.826442
factor(year)2012  1.541e-02  8.620e-03   1.788 0.073918 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05825 on 4582 degrees of freedom
Multiple R-squared:  0.5734,    Adjusted R-squared:  0.5726
F-statistic: 769.8 on 8 and 4582 DF,  p-value: < 2.2e-16
```

- The slope of torque is -9.892×10^{-4}, indicating that there's a negative relationship between torque and transformed highway MPG. All else equal, two individual cars who differ in torque in 1 unit are expectect to differ in transformed highway MPG by -9.892×10^{-4}

e. It seems reasonable that there may be an interaction between torque and horsepower. Refit the model (with `lm`) and generate an interaction plot, showing how the relationship between torque and MPG changes as horsepower changes. Choose reasonable values of torque, and show lines for three different reasonable values of horsepower.

   (Hint: If you choose to use the interactions package for this, look at the `at` = argument to help with how year comes into play - choose a reasonable single value for year.

```r
gas_cars$year <- as.factor(gas_cars$year)
lm.cars2 <- lm(trans_hw_mpg ~ torque*horsepower +
                 height + length + width + year,
             data = gas_cars)
summary(lm.cars2)
```

```
Call:
lm(formula = trans_hw_mpg ~ torque * horsepower + height + length +
    width + year, data = gas_cars)

Residuals:
     Min       1Q   Median       3Q      Max
-0.24693 -0.03429  0.00032  0.03436  0.79841

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)        2.401e+00  9.886e-03 242.896   <2e-16 ***
torque            -1.486e-03  3.158e-05 -47.040   <2e-16 ***
horsepower        -4.694e-05  3.165e-05  -1.483   0.1381
height             1.224e-04  1.333e-05   9.176   <2e-16 ***
```

```
length             1.307e-05  1.037e-05   1.261   0.2075
width             -5.532e-05  1.062e-05  -5.207    2e-07 ***
year2010          -1.107e-02  7.943e-03  -1.394   0.1634
year2011          -3.261e-03  7.930e-03  -0.411   0.6809
year2012           1.392e-02  7.992e-03   1.742   0.0816 .
torque:horsepower  1.579e-06  5.770e-08  27.369   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05401 on 4581 degrees of freedom
Multiple R-squared:  0.6333,    Adjusted R-squared:  0.6326
F-statistic: 879.2 on 9 and 4581 DF,  p-value: < 2.2e-16
```

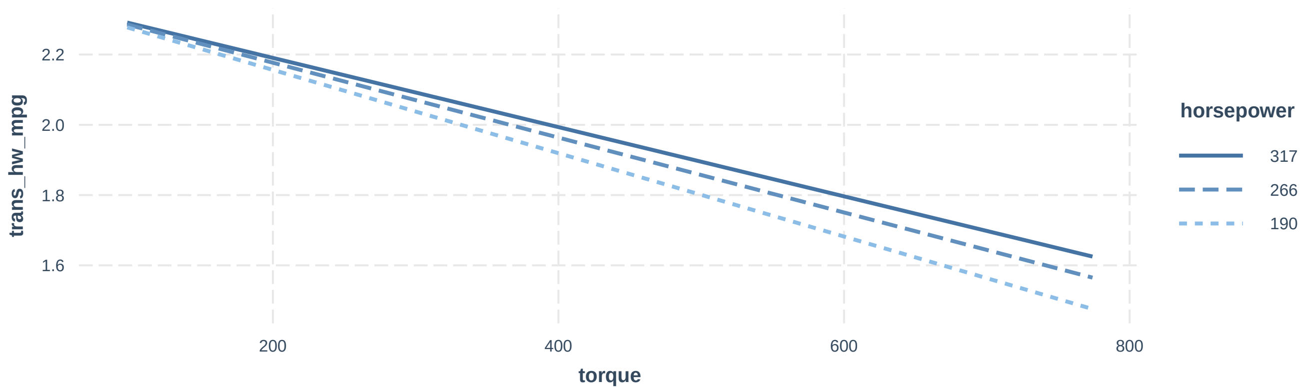
```
 # Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 #   98.0   187.0   260.0   272.7   335.0   774.0
summary(gas_cars$torque)
```

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

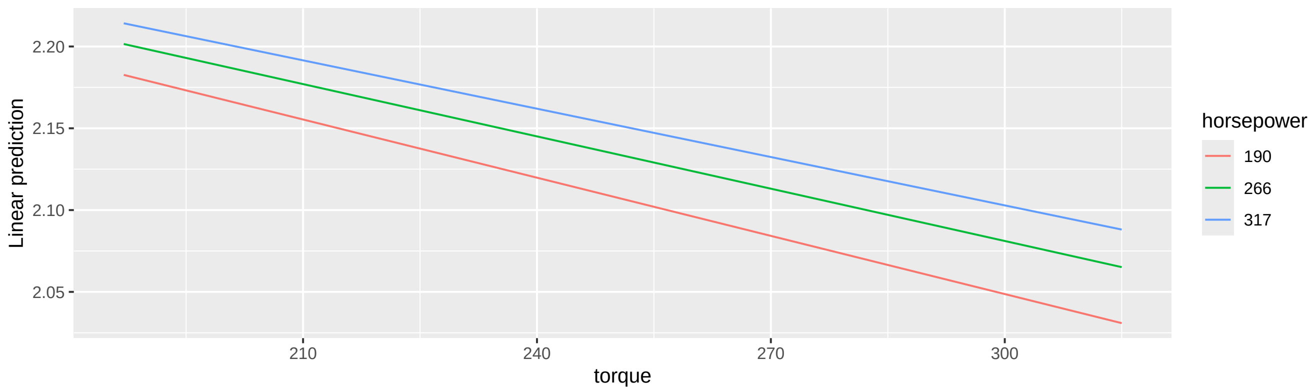
```
# Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   100.0   190.0   266.0   270.5   317.0   638.0
summary(gas_cars$horsepower)
```

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

```
interact_plot(lm.cars2, pred = "torque",
              modx = "horsepower",
              modx.values = c(190,266,317),
              at = list(year = c("2010")),
              data = gas_cars)
```




```
emmip(lm.cars2, horsepower ~ torque,
      at = list(horsepower = c(190,266,317),
                torque = c(187, 260, 315)))
```



f. Calculate $\hat{\beta}$ from d. manually (without using `lm`) by first creating a proper design matrix, then using matrix algebra to estimate β. Confirm that you get the same result as `lm` did prior

```
X <- model.matrix(~ torque + horsepower + height + length + width + year,
             data = gas_cars)
y <- gas_cars$trans_hw_mpg
beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y
beta_hat
```

```
                     [,1]
(Intercept)  2.262216e+00
torque      -9.891529e-04
horsepower   4.171382e-04
height       1.694003e-04
length       1.240187e-05
width       -4.357972e-05
year2010    -9.542194e-03
year2011    -1.875421e-03
year2012     1.540812e-02
```

The result of manual calculation is same as the result as `lm` did