HW-2

vrohatgi

Q-1 : Four unique implementations of a function in R

(a-c) Running all methods with varying input [3,3000] and testing with same seed value to check for same result

(i) Using `for` loop & (ii) Using vectorized function `sample()`

```
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(roxygen2)
library(tinytex)

## Random test for set.seed() function
set.seed(2)
sample(1:6, 6)
```

```
[1] 5 1 6 4 2 3
```

```r
## function definition

dice_amount <- function(num_rolls,reps) {

  ##  since it will cost $2 to play, define the starting amount
  ##  case num_rolls = 0 or some other invalid input is ideally checked at input prompt
  ## creating record of multiple iterations of a set of throws (nxm)
  ## browser()

  trial_record = list()

  for (out_loop in 1:reps){

    start_amount <- -2
    current_amount <- 0
    won_amount <- 0

    for (in_loop in 1:num_rolls){

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

      if (dice_face == 3) {
        current_amount <- current_amount + 6
        paste("At roll: ",in_loop)
        paste("current amount: ",sep = "",current_amount)
      }
      else if (dice_face == 5){
        current_amount <- current_amount + 10
        paste("At roll: ",in_loop)
        paste("current amount: ",sep = "",current_amount)
      }
      else {
        paste("At roll: ",in_loop)
        paste("current amount: ",sep = "",current_amount)
        break
      }

      in_loop <- 0 ## reset value of inner loop

    }
    won_amount <- current_amount + start_amount
    ## just setting new variable for won amount
```

```r
    trial_record[out_loop] <- won_amount
  }
  return (trial_record)

}
```

```r
## num_rolls <- NA_integer_
## num_rolls <- as.integer(readline("Enter the number of dice-rolls: "))

(dice_amount(10,10))
```

```
[[1]]
[1] -2

[[2]]
[1] -2

[[3]]
[1] 4

[[4]]
[1] 4

[[5]]
[1] -2

[[6]]
[1] 4

[[7]]
[1] -2

[[8]]
[1] -2

[[9]]
[1] -2

[[10]]
[1] 4
```

- OBSERVATION : By default, when you create a numeric vector using the c() function it will produce a vector of double precision numeric values. To create a vector of integers using c() you must specify explicitly by placing an L directly after each number.

(iii) Using single table to capture all dice throws

(iv) Using an "apply" class function

```
## Considering that "apply()" class functions
## simply help us avoid using for() loop explicitly

# Number of experiments
num_experiments <- 5

# Number of rolls per experiment
num_rolls <- 10

# Function to simulate rolling a die
roll_die <- function(n) {
  sample(1:6, n, replace = TRUE)
}

# Use apply to simulate the experiments
results <- t(apply(matrix(1:num_experiments, nrow = num_experiments), 1, function(x) roll_di

# Print the results
print(results)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    6    5    6    6    3    1    5    5     6
[2,]    6    2    2    3    4    3    1    1    5     1
[3,]    2    4    5    6    5    4    2    5    6     5
[4,]    2    6    4    4    4    4    1    2    2     6
[5,]    6    3    5    3    6    5    5    1    5     6
```

Evaluating computational complexity using `microbenchmark()`

Evaluating fairness using Monte Carlo simulation

Q.2) Linear Regression on "Cars" Data set

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

(a) Rename the Data

1			
2			
3	Dimensions	Height	
4	Dimensions	Length	
5	Dimensions	Width	
6	Engine Information	Driveline	
7	Engine Information	Engine Type	
8	Engine Information	Hybrid	
9	Engine Information	Number of Forward Gears	
10	Engine Information	Transmission	
11	Engine Information	Engine Statistics	Horsepower
12	Engine Information	Engine Statistics	Torque
13	Fuel Information	City mpg	
14	Fuel Information	Fuel Type	
15	Fuel Information	Highway mpg	
16	Identification	Classification	
17	Identification	ID	
18	Identification	Make	
19	Identification	Model Year	
20	Identification	Year	

Upon examining the csv file, we can determine the precise labels and devise a new, shorter name for each column.

```r
colnames(cars_df)
```

```
 [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"
```

```
column_rename <- c("H","L","W","Eng_Dr_Line","Eng_Type","Eng_Hybrid","Eng_Forward_G","Eng_Tra
```

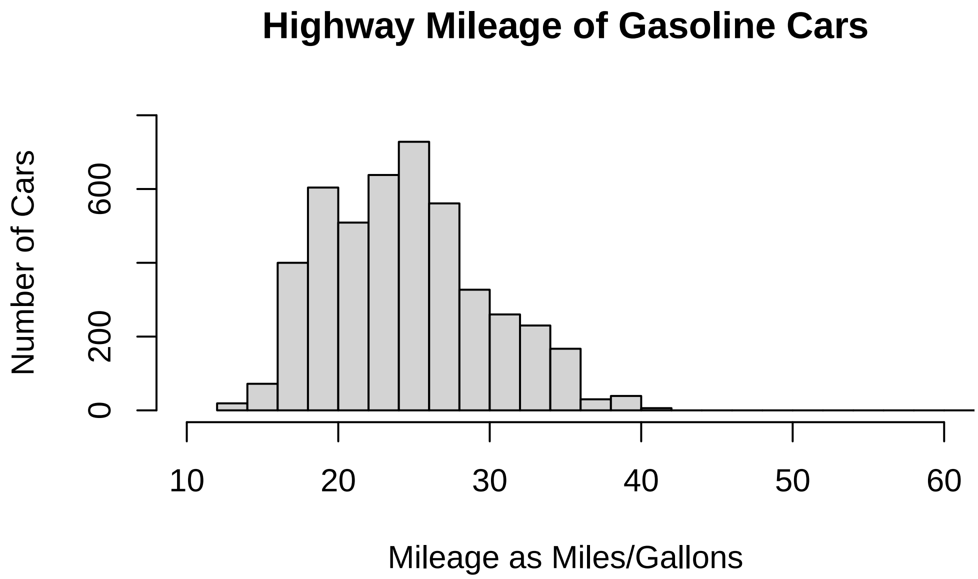
```
colnames(cars_df) <- column_rename
```

```
colnames(cars_df)
```

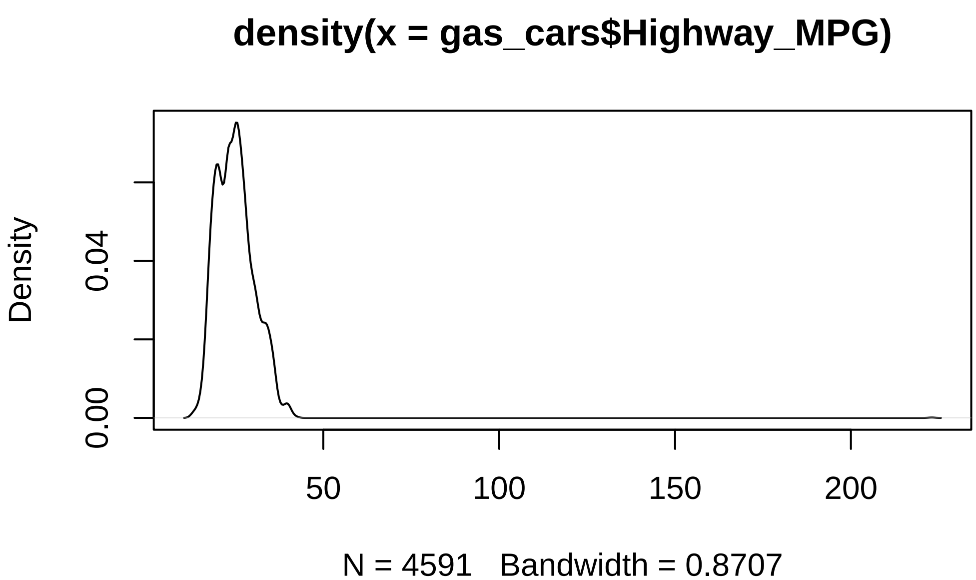
```
 [1] "H"               "L"               "W"               "Eng_Dr_Line"
 [5] "Eng_Type"        "Eng_Hybrid"      "Eng_Forward_G"   "Eng_Trans"
 [9] "City_MPG"        "Fuel_Type"       "Highway_MPG"     "Car_Class"
[13] "Car_ID"          "Car_Make"        "Car_Model_Year"  "Car_Year"
[17] "Eng_Horse_Power" "Eng_Torque"
```

```
## cars_df -- Used to check output

## extracting data for fuel type "gasoline"

gas_cars <- subset(cars_df,Fuel_Type == "Gasoline")

## gas_cars -- used to check output
```

```
## Distribution of highway mileage

hist(gas_cars$Highway_MPG, main = "Highway Mileage of Gasoline Cars", xlab = "Mileage as Mil
```




```r
temp_plot <- plot(density(gas_cars$Highway_MPG))
polygon(temp_plot)
```




```r
sd(gas_cars$Highway_MPG)
```

```
[1] 6.033656
```

```r
mean(gas_cars$Highway_MPG)
```

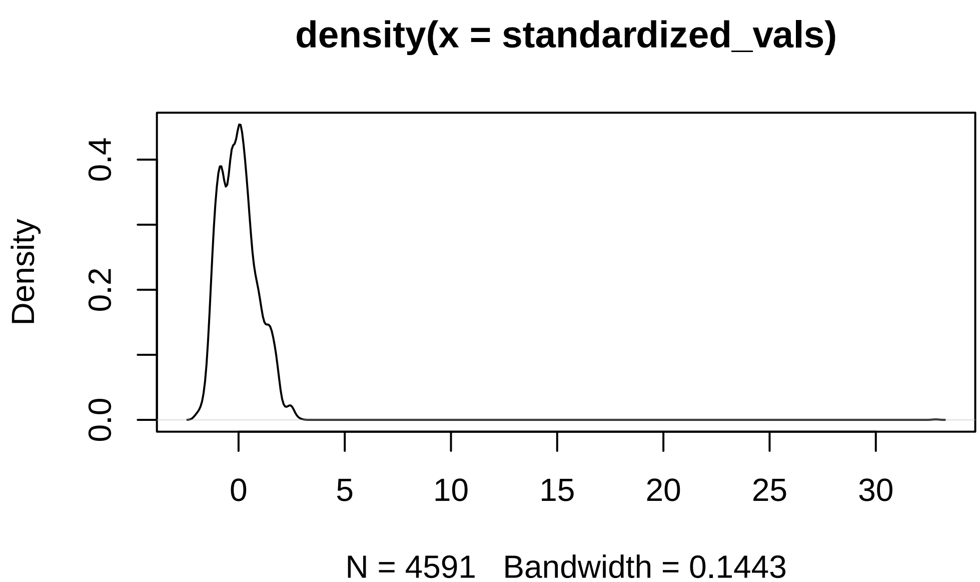
```
[1] 24.96689
```

```r
glimpse(gas_cars$Highway_MPG)
```

```
 int [1:4591] 25 28 30 28 28 27 26 18 20 30 ...
```

Based on the above plots, highway mileage appears to have a normal distribution. The mean mileage is 24 while the standard deviation is roughly 6. We can attempt to Z-transform the scores to control for the deviation around the mean.

```r
standardized_vals <- scale(gas_cars$Highway_MPG)
plot(density(standardized_vals))
```




```r
sd(standardized_vals)
```

```
[1] 1
```

```r
integer(mean(standardized_vals))
```

```
integer(0)
```

```
glimpse(standardized_vals)
```

```
 num [1:4591, 1] 0.00549 0.5027 0.83417 0.5027 0.5027 ...
 - attr(*, "scaled:center")= num 25
 - attr(*, "scaled:scale")= num 6.03
```

It appears that, upon Z-transformation, the standard deviation reduces from 6 to 1 while the mean becomes zero. In order to check whether transformation of Highway Mileage variable is needed, we can later run the computations with both versions of the feature. But for the time being, if we use the standardized values, we might lose some of the detail captured in the relatively large variance in the original values.

```
gas_cars<- na.omit(gas_cars) ## check to remove any rows with a missing value
```

Computing some general statistics :

```
gas_cars_num <- gas_cars %>% select(where(is.numeric)) ## create a numeric columns only slic

## drop the non-comparable columns

gas_cars_2 <- data.frame(gas_cars$Eng_Forward_G,gas_cars$City_MPG,gas_cars$Highway_MPG,gas_ca

cor(gas_cars_2, method = c("pearson", "kendall", "spearman"))
```

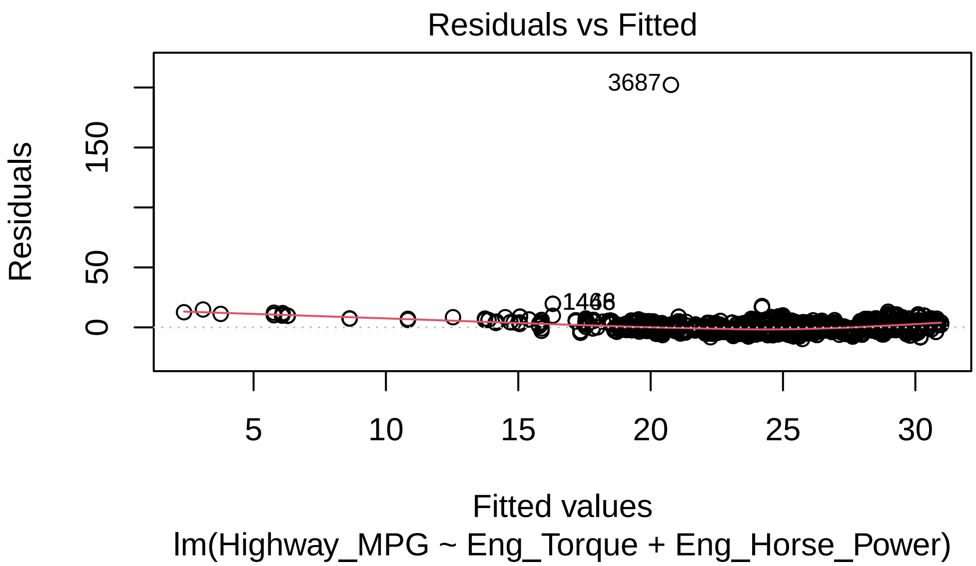
```
                         gas_cars.Eng_Forward_G gas_cars.City_MPG
gas_cars.Eng_Forward_G               1.00000000        -0.0695871
gas_cars.City_MPG                   -0.06958710         1.0000000
gas_cars.Highway_MPG                 0.03274557         0.8271942
gas_cars.Eng_Horse_Power             0.33588453        -0.7409432
gas_cars.Eng_Torque                  0.23254242        -0.7878203
                         gas_cars.Highway_MPG gas_cars.Eng_Horse_Power
gas_cars.Eng_Forward_G             0.03274557                0.3358845
gas_cars.City_MPG                  0.82719420               -0.7409432
gas_cars.Highway_MPG               1.00000000               -0.5566069
gas_cars.Eng_Horse_Power          -0.55660688                1.0000000
gas_cars.Eng_Torque               -0.62114741                0.9474896
                         gas_cars.Eng_Torque
gas_cars.Eng_Forward_G             0.2325424
gas_cars.City_MPG                 -0.7878203
gas_cars.Highway_MPG              -0.6211474
gas_cars.Eng_Horse_Power           0.9474896
gas_cars.Eng_Torque                1.0000000
```

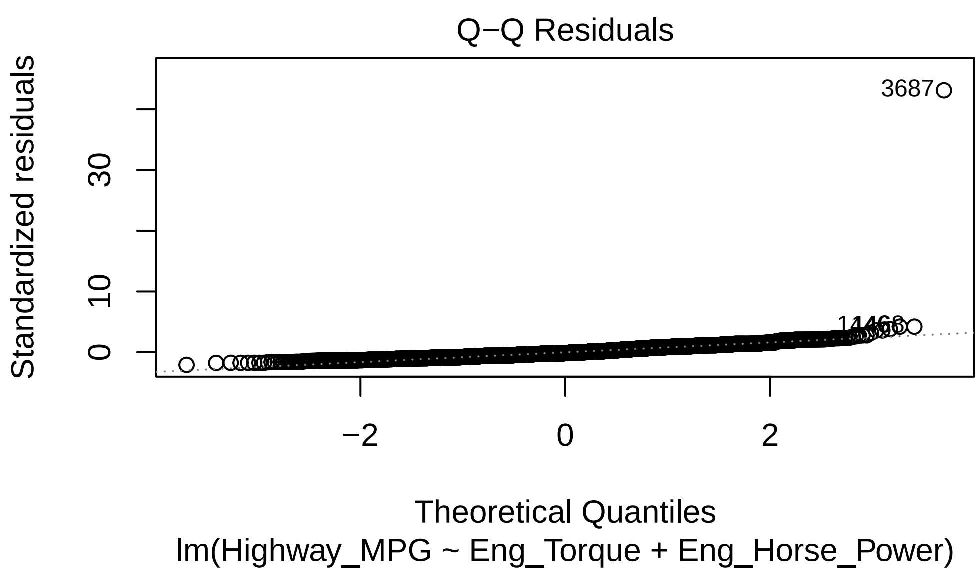
```r
gas_cars_num <- data.matrix(gas_cars_2)
```

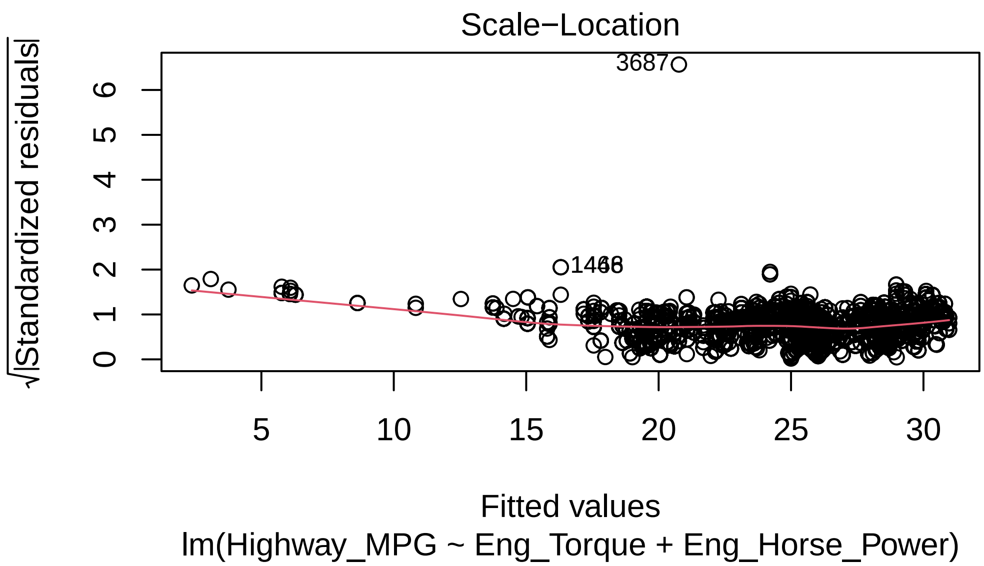
```r
Linear_regress <- lm(Highway_MPG~Eng_Torque+Eng_Horse_Power, data=gas_cars)
summary(Linear_regress)
```

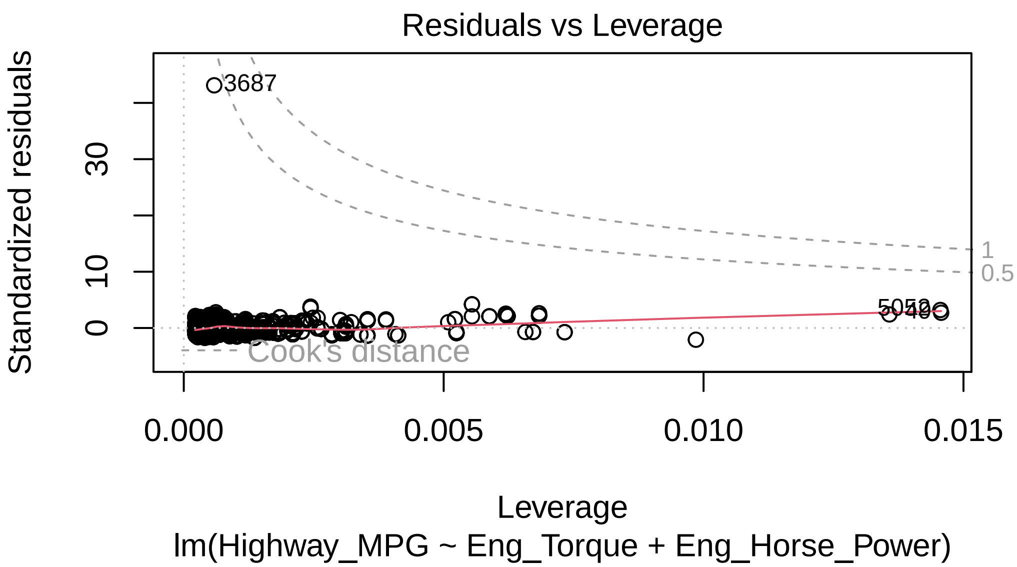
```
Call:
lm(formula = Highway_MPG ~ Eng_Torque + Eng_Horse_Power, data = gas_cars)

Residuals:
    Min      1Q  Median      3Q     Max 
 -9.729  -2.632  -0.661   2.489 202.234 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)     34.386412   0.202445 169.856   <2e-16 ***
Eng_Torque      -0.054600   0.002137 -25.551   <2e-16 ***
Eng_Horse_Power  0.019332   0.002222   8.699   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.691 on 4588 degrees of freedom
Multiple R-squared:  0.3958,    Adjusted R-squared:  0.3955 
F-statistic:  1503 on 2 and 4588 DF,  p-value: < 2.2e-16
```

```r
plot(Linear_regress)
```


Residuals vs Fitted
3687
1468
Residuals
0
50
150
5
10
15
20
25
30
Fitted values
lm(Highway_MPG ~ Eng_Torque + Eng_Horse_Power)


Q−Q Residuals
3687
Standardized residuals
0
10
30
−2
0
2
Theoretical Quantiles
lm(Highway_MPG ~ Eng_Torque + Eng_Horse_Power)






```
ANOVA_COMP <- aov(Highway_MPG~Eng_Torque * Eng_Horse_Power, data=gas_cars)
summary(ANOVA_COMP)
```

```
                             Df Sum Sq Mean Sq F value Pr(>F)    
Eng_Torque                    1  64471   64471 3323.96 <2e-16 ***
Eng_Horse_Power               1   1665    1665   85.86 <2e-16 ***
Eng_Torque:Eng_Horse_Power    1  11995   11995  618.41 <2e-16 ***
Residuals                  4587  88968      19                   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To control for other variables in analyzing the relationship between Highway Mileage and Torque, we can start by making make separate data frames

```
# Basic Interaction Plot
interaction.plot(x.factor = gas_cars$Eng_Torque,
                 trace.factor = gas_cars$Highway_MPG,
                 response = gas_cars$Eng_Horse_Power)
```