Self Reflection

Diksha Shrestha

4/15/2022

Provide a URL for your final project. If you created a Shiny App as your data product, you should include a link to the GitHub repository that contains your code as well as a link to your Shiny App hosted on shinyapp.io (see Chapter 2 of the shinyapps.io User Guide - this is free, but you need to sign up for an account). If you created some other type of data product, you should include a link to the GitHub repository that contains your coce as well as a direct link to your data product.

My final project is NewYork Property Price Analysis. Please find the link to my Github here: [Project] (https://github.com/dikshashrestha/Final_Project) I have also created a Shiny app, please find the link to it. Shiny

Did you work with a group? If so, include the names of your other group members here.

No, I work individually for my project.

A thorough reflection on your work in this class. Talk about the work you've done for the course. Remember that I am interested in the progress have you made towards each course-level learning objectives. Look through your work to determine what you could use to demonstrate (show and discuss) your progress. Provide links directly to your evidence or embed snapshot examples of your work. Be sure to describe how your work demonstrates your progress towards each objectives. Consider the work you did on the final project, your work earlier in the term, the feedback you offered your peers on their work, and how you met your own goals. Feel free to include more links to examples of your work as necessary (again, please point directly to the specific work - not some general document link - so that I can easily review it). Tell me what you are particularly proud of. This is the place to be as honest as possible about your work, both reflecting critically and talking about what you proved capable of in the midst of an incredibly challenging semester. Remember that this is a reflection about your work, not your classmates.

My growth in this course has always been progressive. As I had never worked on R before, I was skeptical about it to be a difficult programming language. However, with each class and the class activities, it boosted my confidence and enhanced my knowledge on this programming language. Although, it has a different syntax than that of the Python language, this course has helped me gain an in-depth knowledge of its syntax and the packages. Similarly, the implementation of all the learning and class activities in my final project boosted my knowledge of R in much more detail. I build a shiny app to analyze the NewYork Property Price from 2015 to 2019. I further did some mini projects to implement my course learning. Hence, I believe I have met all the course objectives. Please refer below for the codes, screenshots, and the github links which shows I have met the course objectives.

Objective 1: Import, manage, and clean data

I have imported, managed and cleaned the data during my final project to build shiny and in my mini projects. I have imported the csv file and excel files into a dataframes to proceed with my project.

Shiny Project

Import

I imported the file from the excel sheet and mapped it in the list. Also, the below code shows the mapping it into the dataframe.

```
library(DT)
library(readxl)
library(plotly)
```

```
## Loading required package: ggplot2

##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(tidyr)
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.89 loaded
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:plotly':
##
##     select
```

```
library(corrr)
library(ggcorrplot)
```

```
#excel_sheets("City_Data.xlsx") %>% map(~read_xlsx("City_Data.xlsx",.))
```

```
#Data <- excel_sheets("City_Data.xlsx") %>% map_df(~read_xlsx("City_Data.xlsx",.))
```

This code chunk extracts each sheet from the City Data excel sheet.

```
excel_sheets("City_Data.xlsx")
```

```
## [1] "2015" "2016" "2017" "2018" "2019"
```

Clean

I executed the below code to clean the data by removing the empty objects, na values, and 0s from the dataset.

```
#Data <- na.omit(Data)
#Data
```

```r
Data <- read.csv('New_Data.csv')
```

```r
Data = Data[!(is.na(Data$Land_Square_Feet) | Data$Land_Square_Feet == "0"),]
```

```r
Data = Data[!(is.na(Data$Gross_Square_Feet) | Data$Gross_Square_Feet == "0"),]
```

```r
Data = Data[!(is.na(Data$Sale_Price) | Data$Sale_Price == "0"),]
```

Manage

Additionally, I also managed my dataset using the dplyr function where I selected only the columns I needed to preform the task and removed the columns that were not needed.

```r
#Data_dr = dplyr::select(New_Data, -c(Tax_Class_At_Present, Apartment_Number, Residential_Units, Commer
 #head (Data_dr)
```

I have also used the dplyr functions such as select, filter summarize to know the average sale price of the NewYork city.

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
New_Data = read.csv("New_Data.csv")
Trend <- New_Data %>%
  dplyr::select(City, Year, Sale_Price) %>%
  group_by(City, Year) %>%
  summarise(Average_Sale_Price = mean(Sale_Price))
```

```
## `summarise()` has grouped output by 'City'. You can override using the `.groups` argument.
```

Furthermore, there are other task for which I have used the dplyr functions.

```r
New_Data = read.csv("New_Data.csv")
Residential_Sum <- New_Data %>%
  dplyr::select(Residential_Units, Year, Borough, City) %>%
  group_by(City, Year)%>%
  summarise(Sum_Residential = sum(Residential_Units))
```

```
## `summarise()` has grouped output by 'City'. You can override using the `.groups` argument.
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
```

```
##     date, intersect, setdiff, union
```

```
New_Data$Sale_Date <- as_date(New_Data$Sale_Date)
New_Data$Years <- as.numeric(format(New_Data$Sale_Date,"%Y"))
head (New_Data)
```

```
##   X Borough Neighborhood           Building_Class_Category Tax_Class_At_Present
## 1 1       2      FORDHAM 08  RENTALS - ELEVATOR APARTMENTS                    2
## 2 2       2    RIVERDALE         01  ONE FAMILY DWELLINGS                    1
## 3 3       3   BATH BEACH   07  RENTALS - WALKUP APARTMENTS                   2A
## 4 4       3   BATH BEACH   07  RENTALS - WALKUP APARTMENTS                   2A
## 5 5       3   BATH BEACH   07  RENTALS - WALKUP APARTMENTS                   2A
## 6 6       3   BATH BEACH   07  RENTALS - WALKUP APARTMENTS                   2A
##   Block Lot Building_Class_At_Present              Address Apartment_Number
## 1  3160  35                       D7 2051 GRAND CONCOURSE               67
## 2  5920 401                       A1 680-06 DELAFIELD WAY                6
## 3  6365  16                       C2   55 BAY 14TH STREET                6
## 4  6365  73                       C2      8674 17TH AVENUE                6
## 5  6374  27                       C3 57 BAY 23RD   STREET                4
## 6  6374  45                       C3 15 BAY 23RD   STREET                4
##   Zip_Code Residential_Units Commercial_Units Total_Units Land_Square_Feet
## 1    10453                67                1          68            15753
## 2    10471                 1                0           1             5194
## 3    11214                 6                0           6             2781
## 4    11214                 6                0           6             2432
## 5    11214                 4                0           4             2417
## 6    11214                 4                0           4             2417
##   Gross_Square_Feet Year_Built Tax_Class_At_Time_Of_Sale
## 1             54360       1922                         2
## 2              3600       1987                         1
## 3              3708       1910                         2
## 4              5082       1926                         2
## 5              3175       1923                         2
## 6              2702       1922                         2
##   Building_Class_At_Time_Of_Sale Sale_Price  Sale_Date Year    City Years
## 1                             D7   10025000 2015-03-11 2015   Bronx  2015
## 2                             A1    1836000 2015-01-15 2015   Bronx  2015
## 3                             C2    1180000 2015-01-21 2015 Broklyn  2015
## 4                             C2    1250000 2015-05-28 2015 Broklyn  2015
## 5                             C3    1329000 2015-02-27 2015 Broklyn  2015
## 6                             C3    1330000 2015-10-07 2015 Broklyn  2015
```

Mini Project

Import

I have also imported the csv file to process my mini project.

```
avocado <- read.csv("Avo.csv")
head(avocado)
```

```
##       Date AveragePrice TotalVolume
## 1 12/27/15         1.33    64236.62
## 2 12/20/15         1.35    54876.98
## 3 12/13/15         0.93   118220.22
## 4  12/6/15         1.08    78992.15
## 5 11/29/15         1.28    51039.60
```

```
## 6 11/22/15         1.26    55979.78
```

Clean

I have also cleaned the data as I have removed the 0s, NAs and empty object in the dataset.

```
avocado = avocado[!(is.na(avocado$AveragePrice) | avocado$AveragePrice == "0"),]
```

```
avocado = avocado[!(is.na(avocado$TotalVolume) | avocado$TotalVolume == "0"),]
```

Manage

I have used the dplyr's mutate function which provided the numerical value as per the MM/DD/YYY format.

```
avo1 <-avocado %>%
  mutate(Date = as.Date(Date, "'%m,%d,%Y"))

head(avo1)
```

```
##   Date AveragePrice TotalVolume
## 1 <NA>         1.33    64236.62
## 2 <NA>         1.35    54876.98
## 3 <NA>         0.93   118220.22
## 4 <NA>         1.08    78992.15
## 5 <NA>         1.28    51039.60
## 6 <NA>         1.26    55979.78
```

Mini Project

Import

I have imported the csv file for the two dataset which consists the food caloreis and the other consist the servings to gram.

```
cal <- read.csv("Calories.csv")
ser <- read.csv("Serving.csv")
```

Manage

I have used a merge function to join the two dataset into one dataset. This performs an inner join where it returns the rows that have matching in left and right table.

```
food <- merge(cal,ser,by="Food")
head(food)
```

```
##          Food Calories             Serving
## 1   Artichoke       60 1 artichoke (128 g)
## 2     Arugula        1         1 leaf (2 g)
## 3   Asparagus        2       1 spear (12 g)
## 4   Aubergine      115 1 aubergine (458 g)
## 5    Beetroot       35        1 beet (82 g)
## 6 Bell Pepper       15      1 pepper (73 g)
```

Mini Project

Import

```
Sale <- read.csv("CitPrice.csv")
head(Sale)
```

```
##      City Average.Sale.Price Year
## 1   Bronx             890000 2015
## 2   Bronx             750000 2016
```

Description: df [25 x 3]

Food <chr>	Calories <int>	Serving <chr>
Artichoke	60	1 artichoke (128 g)
Arugula	1	1 leaf (2 g)
Asparagus	2	1 spear (12 g)
Aubergine	115	1 aubergine (458 g)
Beetroot	35	1 beet (82 g)
Bell Pepper	15	1 pepper (73 g)
Black Olives	2	1 olive (2.7 g)
Broccoli	207	1 bunch (608 g)
Brussels Sprouts	8	1 sprout (19 g)
Cabbage	227	1 head (908 g)

1–10 of 25 rows Previous 1 2 3 Next

Figure 1: merg

```
## 3    Bronx             566790 2017
## 4    Bronx             350000 2018
## 5    Bronx            1000000 2019
## 6 Broklyn             900000 2015
```

Description: df [25 x 3]

City <chr>	Average.Sale.Price <int>	Year <int>
Bronx	890000	2015
Bronx	750000	2016
Bronx	566790	2017
Bronx	350000	2018
Bronx	1000000	2019
Broklyn	900000	2015
Broklyn	2300000	2016
Broklyn	870000	2017
Broklyn	1200000	2018
Broklyn	1250000	2019

1–10 of 25 rows Previous 1 2 3 Next

Figure 2: year

Manage

```
sale_data <- pivot_wider(Sale,names_from = Year, values_from = Average.Sale.Price)
sale_data
```

```
## # A tibble: 5 x 6
##   City          `2015`  `2016` `2017`  `2018`  `2019`
##   <chr>          <int>   <int>  <int>   <int>   <int>
## 1 Bronx         890000  750000 566790  350000 1000000
## 2 Broklyn       900000 2300000 870000 1200000 1250000
## 3 Manhattan    3200000 1000000 990000  658000  867000
## 4 StatenIsland  560000  459000 780000  490000  660000
```

```
## 5 Queens         800000  900000 680000  490000  520000
```

A tibble: 5 x 6

City <chr>	2015 <int>	2016 <int>	2017 <int>	2018 <int>	2019 <int>
Bronx	890000	750000	566790	350000	1000000
Broklyn	900000	2300000	870000	1200000	1250000
Manhattan	3200000	1000000	990000	658000	867000
StatenIsland	560000	459000	780000	490000	660000
Queens	800000	900000	680000	490000	520000

5 rows

Figure 3: pivot

I have used the pivot_wider function which increased the columns for each year and decreased the rows.

Conclusion for Objective 1

I believe I have met the Objective 1 of importing, managing, and cleaning the data as I have imported the excel and csv file to work on my dataset. Also, I have restructured the data using the dplyr function such as pipe, select, filter, groupby, summarise. Furthermore, I have isolated my large dataset of NewYork property prices into a dataframe. Also, I have transformed the information to a numerical value using the mutate function on the date and have combined the information from the multiple dataset into one. Similarly, for my shiny project, I have used the lubridate function to format the date and extract the year only.

Objective 2: Create graphical displays and numerical summaries of data for exploratory analysis and presentations.

I have also used the ggplot2 package to visualize my data from the dataframe. I have used different visualization using ggplot2 and they are line chart, bar graph, histogram, density distribution.

```
#ggplot(Trend, aes(x=Year, y = Avg_Sale_Price, colour = City)) +
   #geom_line()+
   #labs(y = 'Avg Sale Price(In Thousands)', title = 'Average Sale Price across cities')
```

```
ggplot(Residential_Sum, aes(x = Year, y = Sum_Residential))+
  geom_bar(stat = "identity", fill = "aquamarine3")+
  labs(y = 'Residential Unit ', title = "Number of Residential Unit sold over the Year")
```

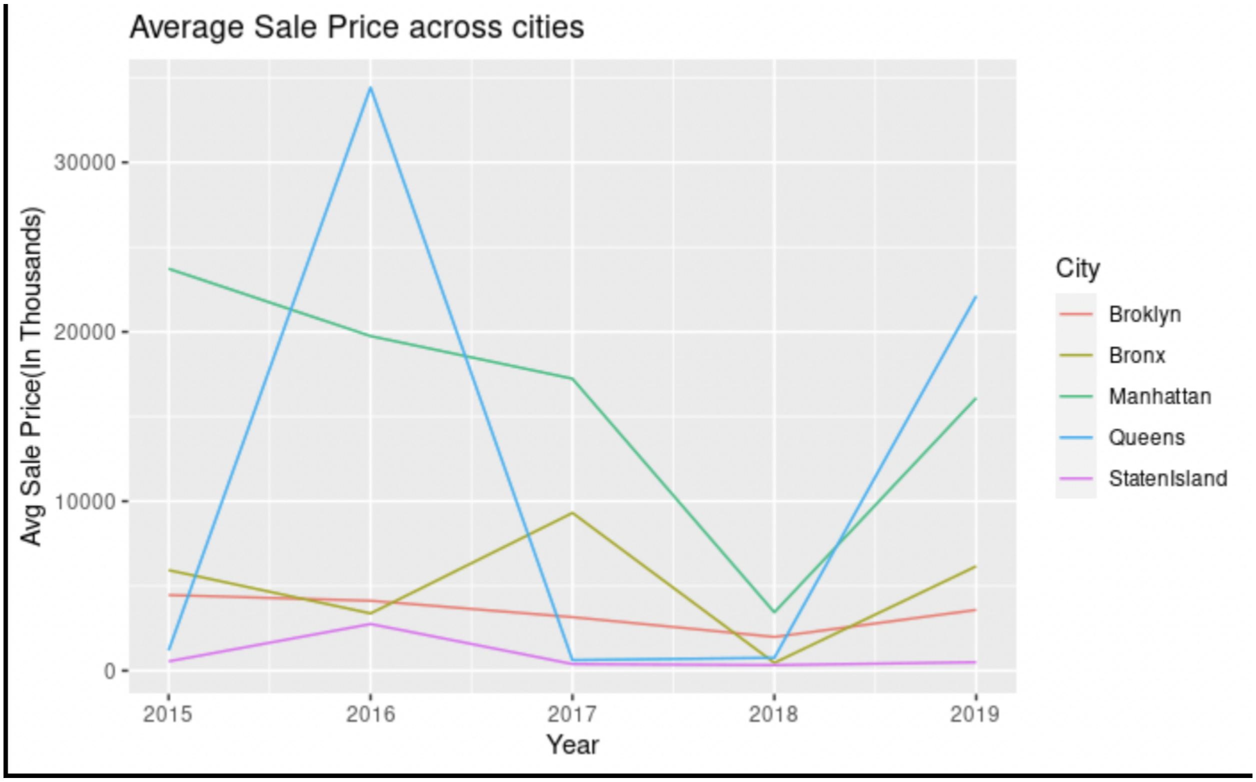
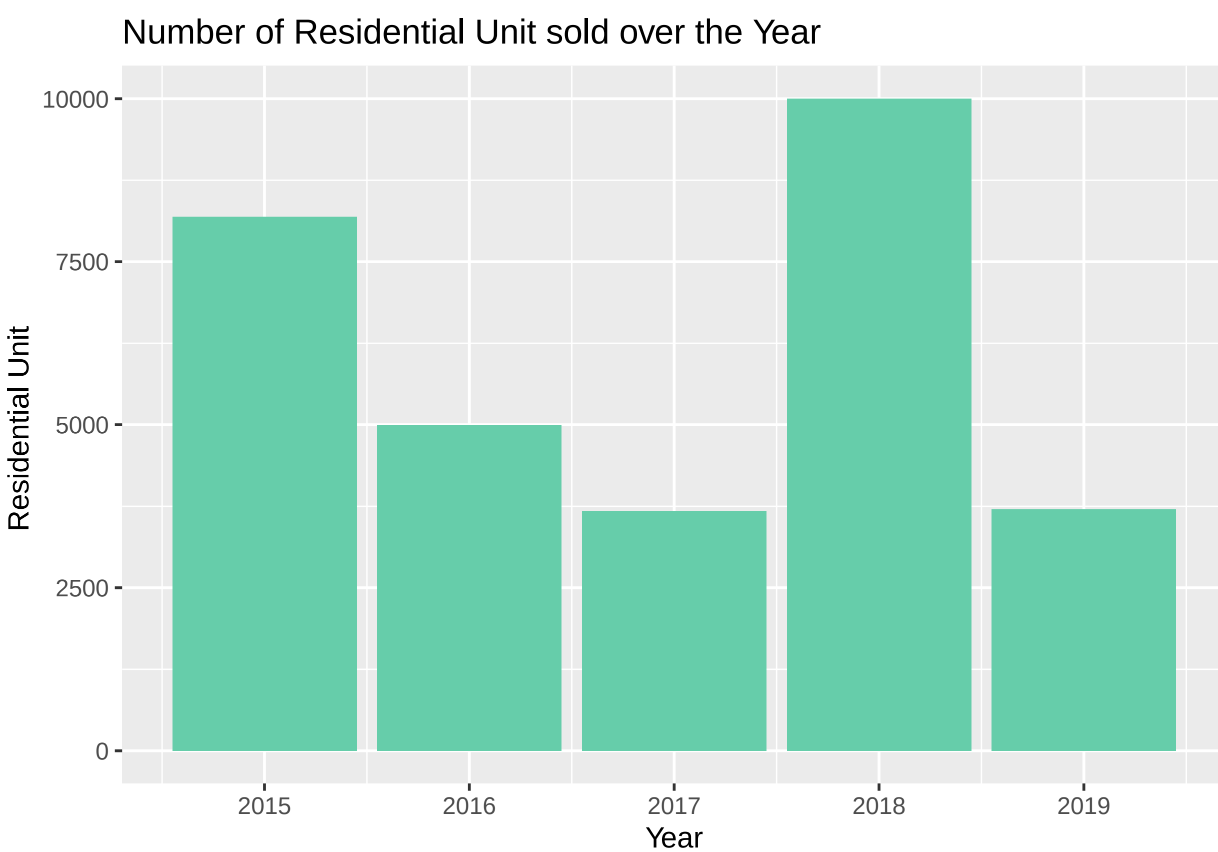
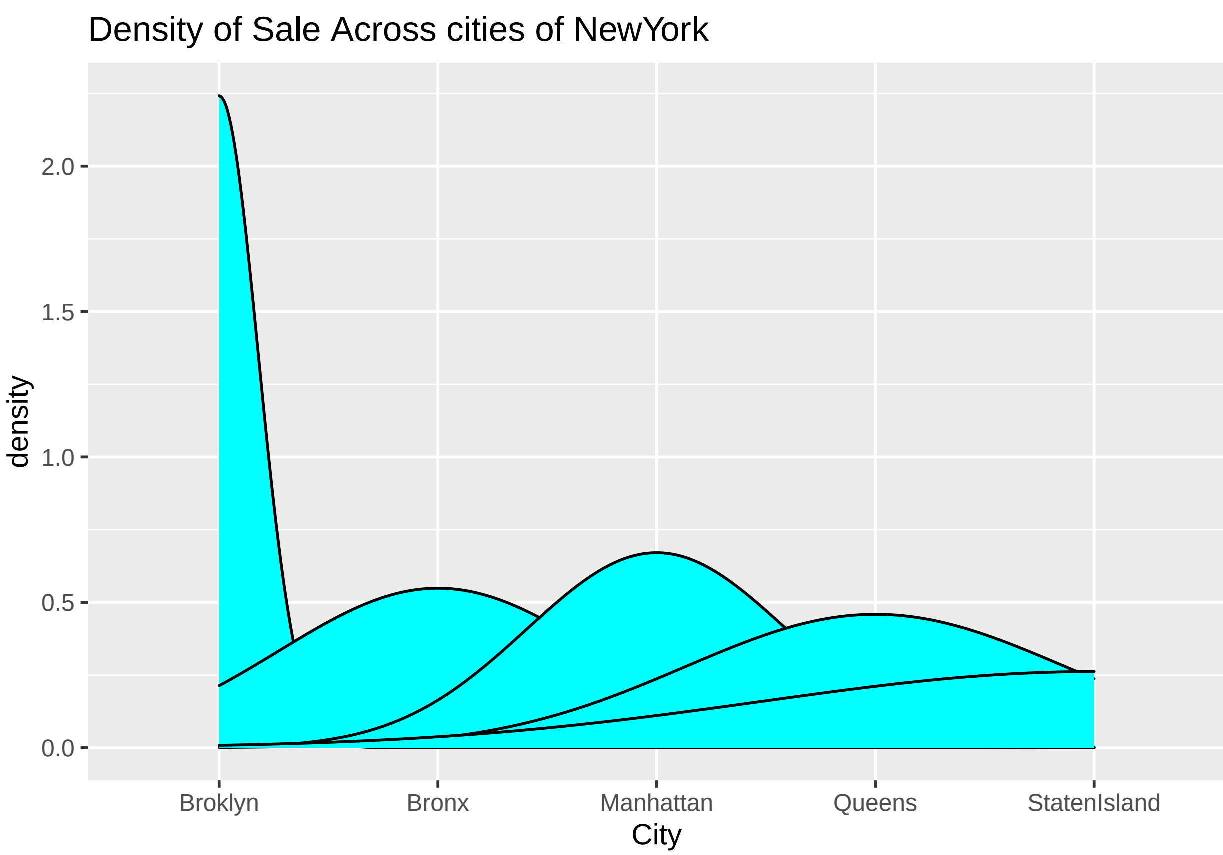


Figure 4: Line



```
library(plotly)
library(tidyr)
New_Data <- read.csv("New_Data.csv")
#Finding the density of sale across cities
City_Density <- New_Data%>%
  dplyr::select(City) %>%
  gather(metric, value) %>%
  ggplot(aes(value, fill = metric)) +
  geom_density(show.legend = FALSE) +
  facet_wrap(~ metric, scales = "free")

ggplot(New_Data, aes(x = City)) +
  geom_density(fill = 'cyan')+
  labs(title = 'Density of Sale Across cities of NewYork')
```



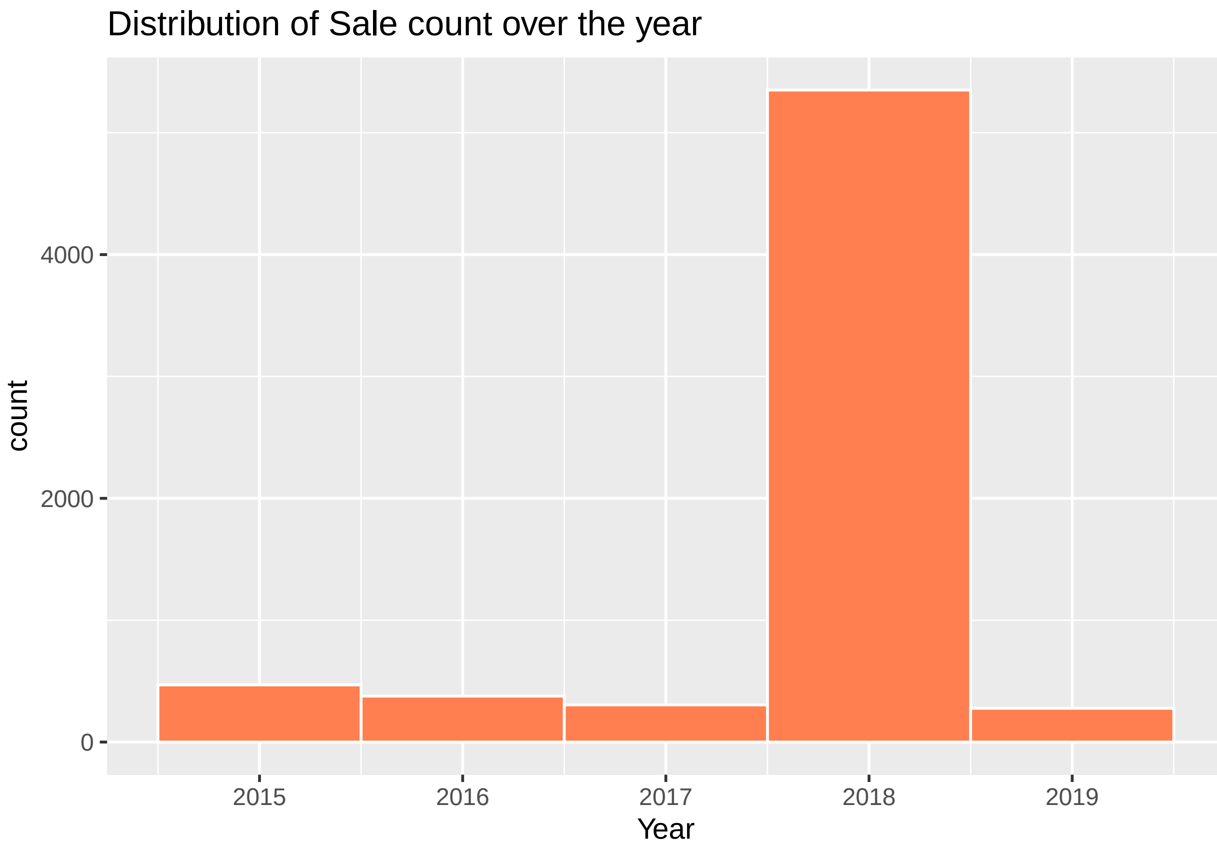
I have used the ggplot 2 and facet wrap here to find out the density. The facet_wrap is used to make a long ribbon and a 2D overlap visualization.

```
Sale_Dist <- New_Data['Year']

# library
library(ggplot2)

# basic histogram
Histo <- ggplot(Sale_Dist, aes(x=Year)) +
  geom_histogram(binwidth = 1, colour = "white", fill = "coral")+
  labs(title = 'Distribution of Sale count over the year')
```

```
Histo
```



Furthermore to using ggplot2, I have also performed a exploratory analysis by executing the correlation matrix. The correlation matrix helped me to find out the correlated variables.

```
# library(MASS)
# library(corrplot)
# library(corrr)
# #corelation=cor(Correl)
# corelation
# library(ggcorrplot)
# ggcorrplot(corelation)
```

Conclusion for Objective 2 I have met the objective 2 where I have used the ggplot2 to visualize my data. Also, with the help of different visualization, I can analyze the data and extract the findings from each visualization. Similarly, I have performed the exploratory analysis to find the correlation coefficient among the variables. From this correlation matrix, we can see which variable has the correlation with the other variable and we can analyze that the sale price and the residential unit shows a correlation coefficient of **0.5**.

Objective 3: Write R programs for simulations from probability models and randomization-based experiments.

I have executed the bootstrapping, simulation, regression and descriptive statistics method for this objective.

Bootstrapping

```
library(boot)

avo <- function(avocado, i){
 d2 <- avocado[i,]
```

Figure 5: corr

	Year	Year_Built	Residential_Units	Commercial_Units	Sale_Price	Borough
Year	1.00000000	-0.057525140	-0.19361364	-0.011955947	-0.058133121	-0.094264266
Year_Built	-0.05752514	1.000000000	0.03293567	0.027342454	0.004272665	0.068010171
Residential_Units	-0.19361364	0.032935669	1.00000000	0.017362607	0.664111600	0.032082817
Commercial_Units	-0.01195595	0.027342454	0.01736261	1.000000000	0.031725169	0.006055243
Sale_Price	-0.05813312	0.004272665	0.66411160	0.031725169	1.000000000	-0.072924982
Borough	-0.09426427	0.068010171	0.03208282	0.006055243	-0.072924982	1.000000000

Figure 6: core

```r
 return(cor(d2$AveragePrice, d2$TotalVolume))
}
```

The above function uses the dataset from the avocado and i refers to the vector which means that the rows from the avocado dataset will be chosen to perform the bootstrap sample.

```r
set.seed(1)
bootstrap_correlation <- boot(avocado,avo,R=10000)
```

The set seed function is used to select the same random number every time we run the same code chunk to ensure we get the same result.

```r
bootstrap_correlation
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = avocado, statistic = avo, R = 10000)
##
##
## Bootstrap Statistics :
##      original       bias    std. error
## t1* -0.533907 0.0008245426  0.06130988
```

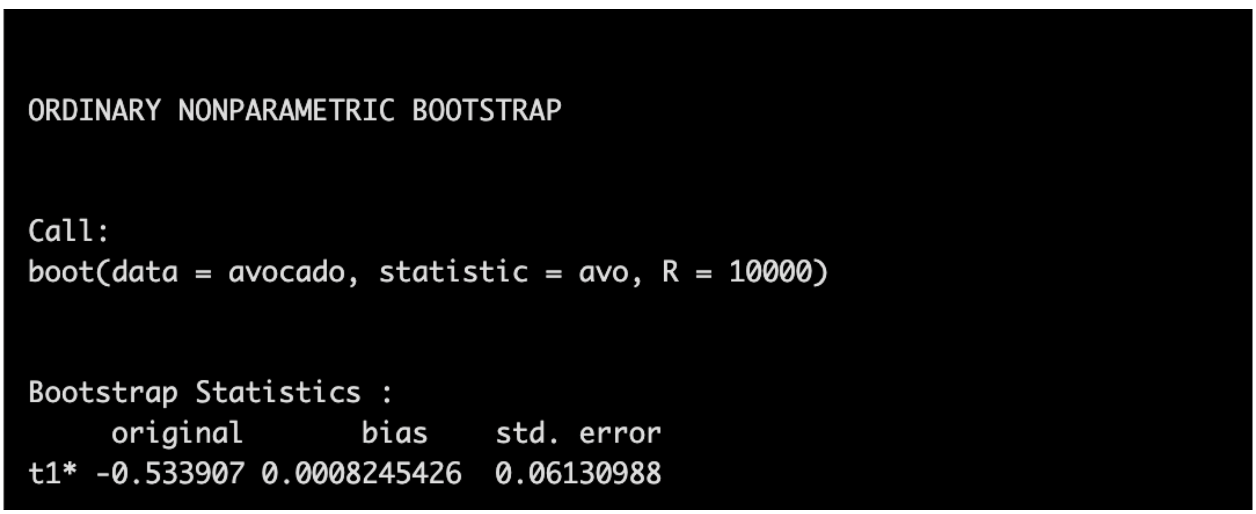


Figure 7: co

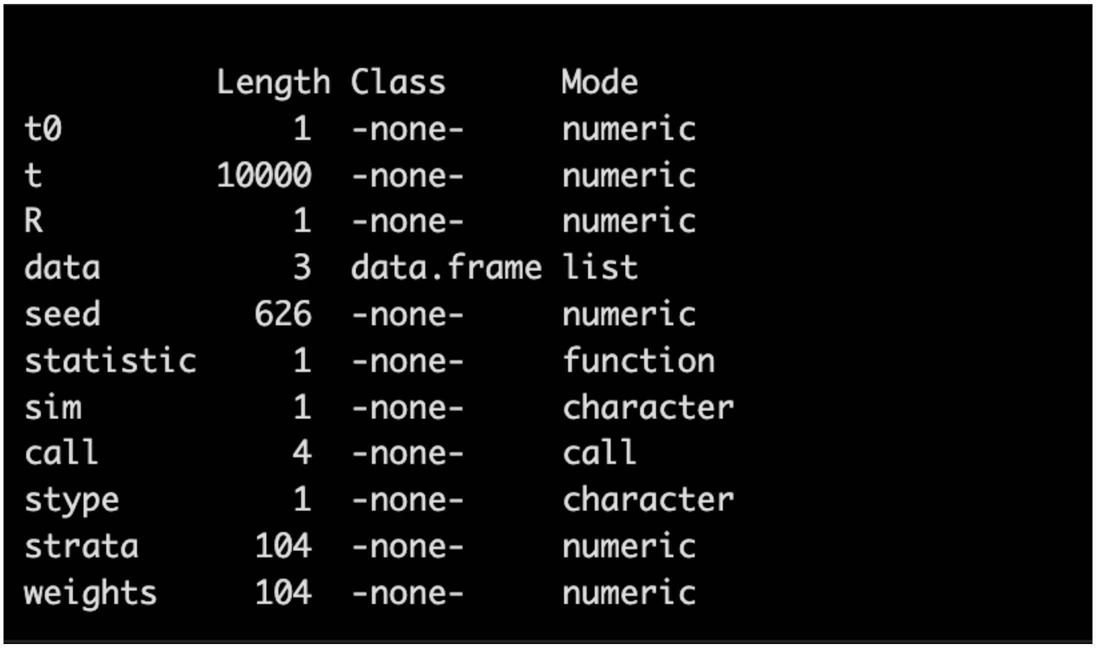
From the above code, we can see that the original correlation between the average price and the total volume is **-0.533** and the standard error between them is **0.061**. This negative correlation shows that the both the variable moves in different direction ad has a negative relationship.

From the above code, we can see that the original correlation between the average price and the total volume is **-0.533** and the standard error between them is **0.061**. This negative correlation shows that the both the variable moves in different direction ad has a negative relationship.

```r
summary(bootstrap_correlation)
```

```
##           Length Class      Mode
## t0            1  -none-     numeric
```

```
## t         10000  -none-     numeric
## R             1  -none-     numeric
## data          3  data.frame list
## seed        626  -none-     numeric
## statistic     1  -none-     function
## sim           1  -none-     character
## call          4  -none-     call
## stype         1  -none-     character
## strata      104  -none-     numeric
## weights     104  -none-     numeric
```



```
           Length Class      Mode
t0             1  -none-     numeric
t          10000  -none-     numeric
R              1  -none-     numeric
data           3  data.frame list
seed         626  -none-     numeric
statistic      1  -none-     function
sim            1  -none-     character
call           4  -none-     call
stype          1  -none-     character
strata       104  -none-     numeric
weights      104  -none-     numeric
```

Figure 8: sum

The above code chunk provides the summary of the correlation.

```
range(bootstrap_correlation$t)
```

```
## [1] -0.7399873 -0.3064337
```

```
[1] -0.7399873 -0.3064337
```

Figure 9: range

Through the range bootstrap code, we can find the range of the correlation co-efficient which is between **-0.739** to **-0.306**.

```
mean(bootstrap_correlation$t)
```

```
## [1] -0.5330825
```

We can see that the mean of the coorelation coefficient is negative which is **-0.533**.

```
sd(bootstrap_correlation$t)
```

```
## [1] 0.06130988
```

Figure 10: mean

Figure 11: sd

We can also know the standard deviation which is **0.0613**.

```
boot.ci(boot.out=bootstrap_correlation,type=c('norm','basic','perc','bca'))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bootstrap_correlation, type = c("norm", "basic",
##     "perc", "bca"))
##
## Intervals :
## Level      Normal              Basic
## 95%   (-0.6549, -0.4146 )   (-0.6580, -0.4205 )
##
## Level     Percentile            BCa
## 95%   (-0.6473, -0.4098 )   (-0.6433, -0.4034 )
## Calculations and Intervals on Original Scale
```

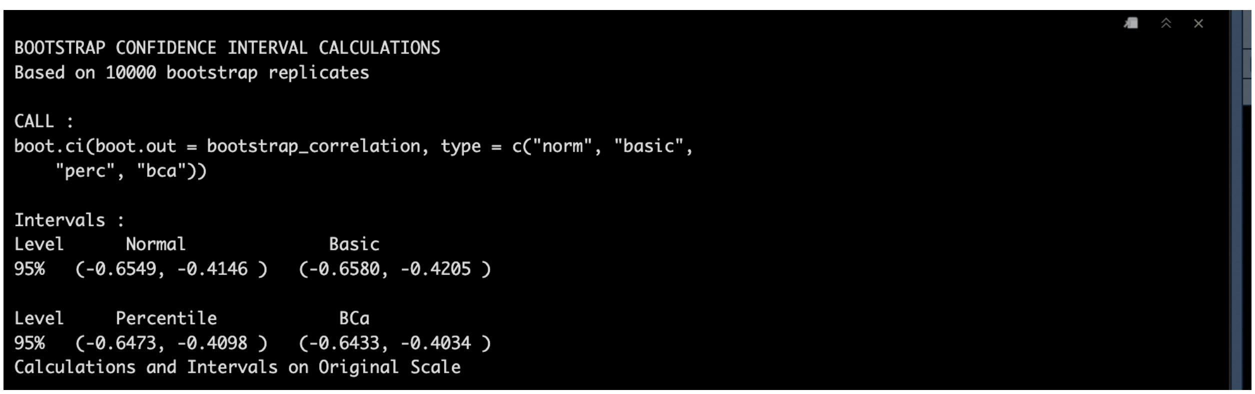


Figure 12: ci

The above function shows the confidence interval of the normal, basic, percentile and bca distribution.

Simulation

```
head(avocado)
```

```
##        Date AveragePrice TotalVolume
## 1 12/27/15         1.33    64236.62
## 2 12/20/15         1.35    54876.98
## 3 12/13/15         0.93   118220.22
## 4  12/6/15         1.08    78992.15
## 5 11/29/15         1.28    51039.60
```

```
## 6 11/22/15          1.26     55979.78
```

```
mean(avocado$TotalVolume)
```

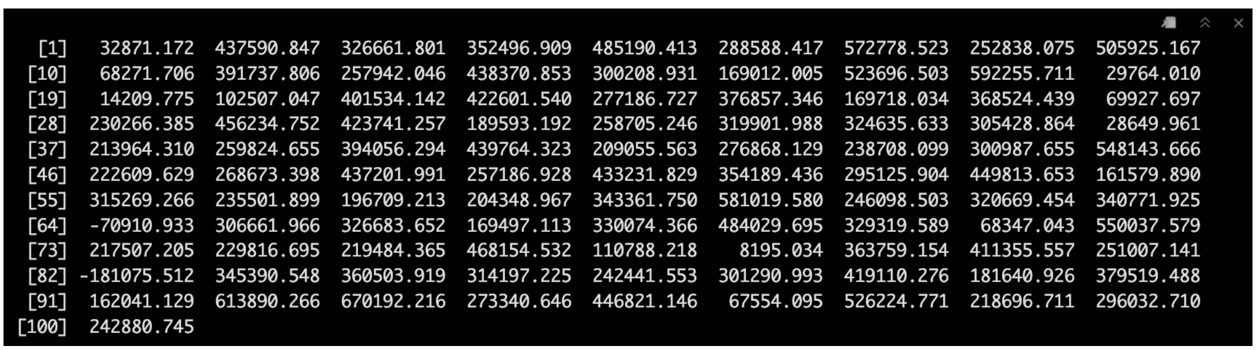
```
## [1] 258277.6
```

```
sd(avocado$TotalVolume)
```

```
## [1] 191412.6
```

```
sim <- rnorm(100,mean=260000,sd=190000)
sim
```

```
##   [1]  300073.739 -117533.239  359486.210  636483.561  366150.450  332035.502
##   [7]  288322.516  140142.830  414121.411  478832.089  393659.365  312020.457
##  [13]  125330.229  448671.358  467832.586  309116.609  460177.465  684553.597
##  [19]  339802.761  305526.830   60325.528  559802.620  331421.450   -9971.017
##  [25]  250361.283  258487.013  318668.845  268889.756  362926.451  299040.400
##  [31]    4998.016  187487.372  414295.953  223242.968  391287.841  396250.767
##  [37]  303734.041  208439.740  -99193.449  299366.365    5997.148  349920.806
##  [43]  107021.973   12290.704  412076.375  475049.328   72626.970  119532.867
##  [49]  523227.437 -137231.332  280291.271  310772.213  318615.341  305056.570
##  [55]  348553.886   27806.126  184568.283  281036.366   82725.640  326601.656
##  [61]  441647.431  326349.326 -194247.076   24742.820  454052.570  215527.428
##  [67]  391891.588  148390.121  160066.706  421785.745  135300.474  441933.439
##  [73]  365026.407  269967.716  359753.262   52887.767  708212.207   73533.334
##  [79]  165288.122  479777.260  497078.127  481756.356  221401.744  265571.669
##  [85]  137300.824  372805.747  378124.694  530466.963  156306.474  113335.810
##  [91]  221758.481  189992.911  370887.695  318949.529   75932.076  150594.041
##  [97]  341366.823   19072.613   84007.892  -14686.745
```



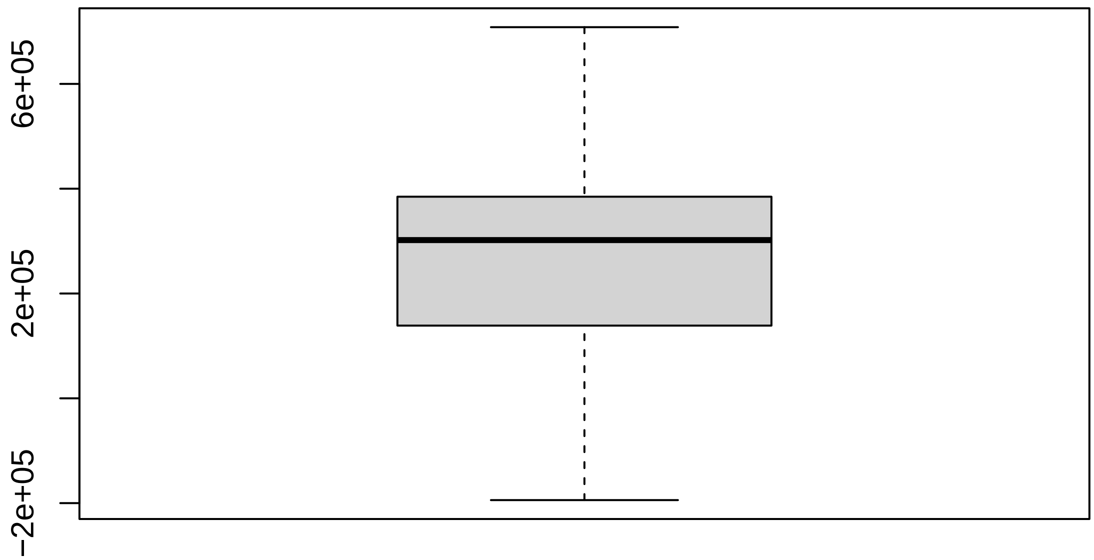
```
  [1]   32871.172  437590.847  326661.801  352496.909  485190.413  288588.417  572778.523  252838.075  505925.167
 [10]   68271.706  391737.806  257942.046  438370.853  300208.931  169012.005  523696.503  592255.711   29764.010
 [19]   14209.775  102507.047  401534.142  422601.540  277186.727  376857.346  169718.034  368524.439   69927.697
 [28]  230266.385  456234.752  423741.257  189593.192  258705.246  319901.988  324635.633  305428.864   28649.961
 [37]  213964.310  259824.655  394056.294  439764.323  209055.563  276868.129  238708.099  300987.655  548143.666
 [46]  222609.629  268673.398  437201.991  257186.928  433231.829  354189.436  295125.904  449813.653  161579.890
 [55]  315269.266  235501.899  196709.213  204348.967  343361.750  581019.580  246098.503  320669.454  340771.925
 [64]  -70910.933  306661.966  326683.652  169497.113  330074.366  484029.695  329319.589   68347.043  550037.579
 [73]  217507.205  229816.695  219484.365  468154.532  110788.218    8195.034  363759.154  411355.557  251007.141
 [82] -181075.512  345390.548  360503.919  314197.225  242441.553  301290.993  419110.276  181640.926  379519.488
 [91]  162041.129  613890.266  670192.216  273340.646  446821.146   67554.095  526224.771  218696.711  296032.710
[100]  242880.745
```

Figure 13: sim

```
summary(sim)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -194247  139432  301904  268351  381416  708212
```

```
boxplot(sim)
```



Regression

```
sale <- read.csv("sales.csv")
sale
```

```
##           City  price NumberofSales
## 1        Bronx 120000          3200
## 2      Broklyn 125000          3000
## 3    Manhattan 200000          2500
## 4       Queens  90000          5000
## 5 Statenisland  85000          4500
```

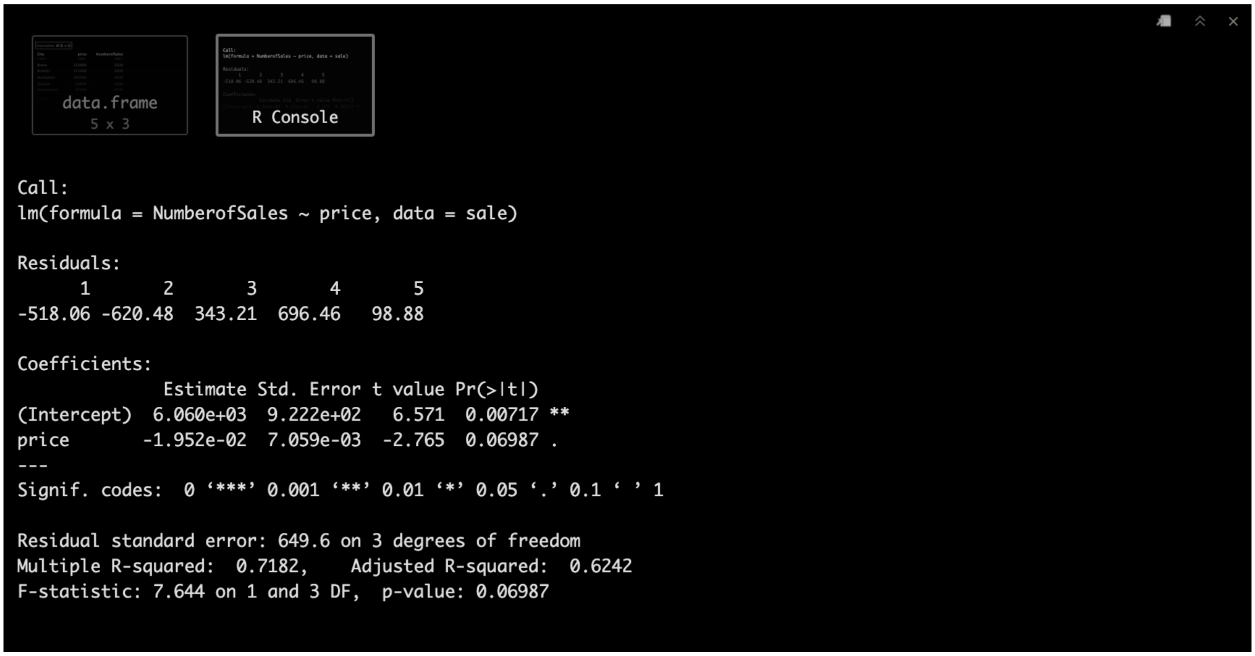
```
relation <- lm(NumberofSales~price, data = sale)

summary(relation)
```

```
##
## Call:
## lm(formula = NumberofSales ~ price, data = sale)
##
## Residuals:
##       1       2       3       4       5
## -518.06 -620.48  343.21  696.46   98.88
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.060e+03  9.222e+02   6.571  0.00717 **
## price       -1.952e-02  7.059e-03  -2.765  0.06987 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 649.6 on 3 degrees of freedom
## Multiple R-squared:  0.7182, Adjusted R-squared:  0.6242
## F-statistic: 7.644 on 1 and 3 DF,  p-value: 0.06987
```

```
# lm()
```

Descriptive Statistics




```
Call:
lm(formula = NumberofSales ~ price, data = sale)

Residuals:
      1       2       3       4       5
-518.06 -620.48  343.21  696.46   98.88

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.060e+03  9.222e+02   6.571  0.00717 **
price       -1.952e-02  7.059e-03  -2.765  0.06987 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 649.6 on 3 degrees of freedom
Multiple R-squared:  0.7182,	Adjusted R-squared:  0.6242
F-statistic: 7.644 on 1 and 3 DF,  p-value: 0.06987
```

Figure 14: ref

Through the descriptive statistics, I have performed min, max, range, standard deviation, quartile, variance, range, and media.

```
foods <- read.csv("Food_Calories.csv")
head(foods)
```

```
##          Food              Serving Calories
## 1   Artichoke 1 artichoke (128 g)       60
## 2     Arugula         1 leaf (2 g)        1
## 3   Asparagus       1 spear (12 g)        2
## 4   Aubergine 1 aubergine (458 g)      115
## 5    Beetroot        1 beet (82 g)       35
## 6 Bell Pepper      1 pepper (73 g)       15
```

Renamed the column names

```
colnames(foods)[2] <- "Serving_per_gram"
head(foods)
```

```
##          Food    Serving_per_gram Calories
## 1   Artichoke 1 artichoke (128 g)       60
## 2     Arugula         1 leaf (2 g)        1
## 3   Asparagus       1 spear (12 g)        2
## 4   Aubergine 1 aubergine (458 g)      115
## 5    Beetroot        1 beet (82 g)       35
## 6 Bell Pepper      1 pepper (73 g)       15
```

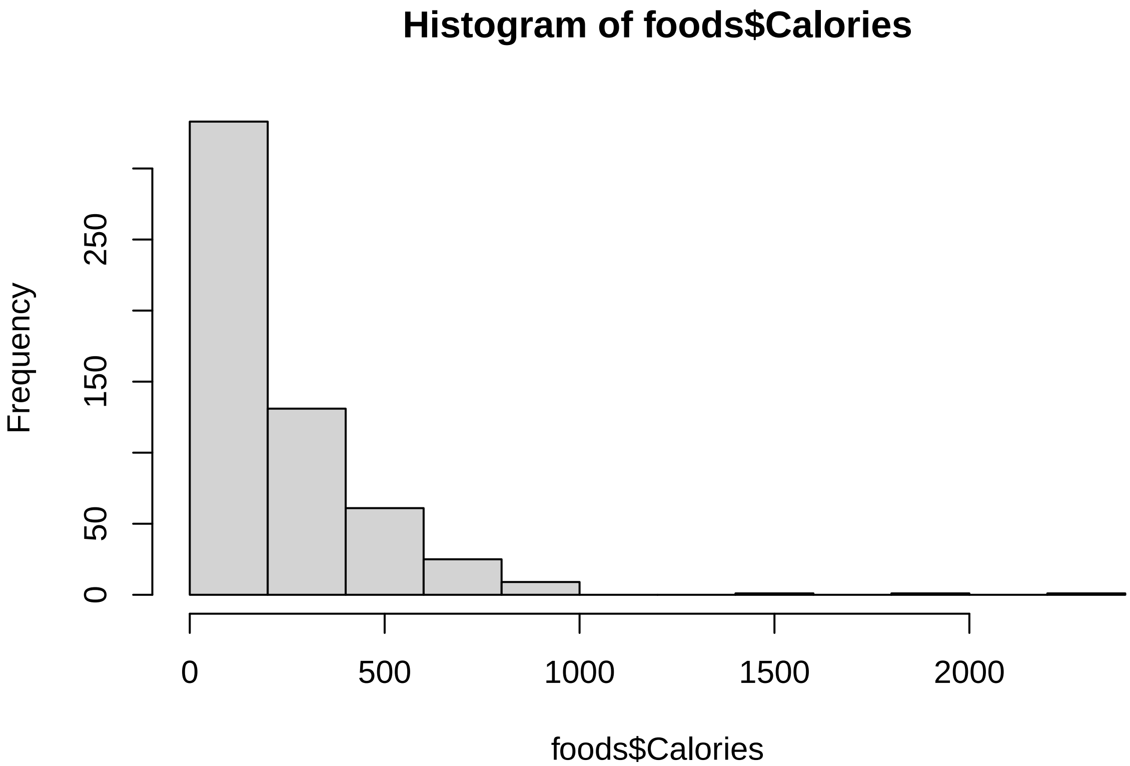
```
library(ggplot2)
```

Taking a random sample of 10%

```
New_Foods <- sample_frac(foods, 0.1)
head(New_Foods)
```

```
##                  Food Serving_per_gram Calories
## 1           Succotash    1 cup (192 g)      221
## 2                Acai  1 oz. (28.35 g)       20
## 3              Lychee  1 lychee (10 g)        7
## 4 Cream of Onion Soup    1 cup (244 g)      107
## 5  Cellophane Noodles    1 cup (140 g)      491
## 6                Nori  1 sheet (2.6 g)        1
```

```
hist(foods$Calories)
```



Conclusion for Objective 3

I have performed all the course activities and have completed the simulation, bootstrapping, regression and descriptive statistics and I have got a in-depth knowledge of working on it. Hence, I believe I have met the objective 3.

Objective 4: Use source documentation and other resources to troubleshoot and extend R programs.

While doing the projects, I faced lots of error. However, all that error is the reason for which I have gained an in-depth knowledge in R, its function and to know the R package. Troubleshooting those error, researching, going through the course work and the activities again, referring to the stackoverflow helped me alot n understanding R to its fullest.

The one example of my troubleshooting was I had load both the dplyr library and MASS library and both of the library had the select function. So, when loading my select function of the dplyr it showed an error saying the unused argument. Then looking at the stackoverflow, I found out that I need to mention which library should the select function use. Hence, I performed below code and that executed my code.

```
New_Data = read.csv("New_Data.csv")
Trend <- New_Data %>%
  dplyr::select(City, Year, Sale_Price) %>%
  group_by(City, Year) %>%
```

```
  summarise(Average_Sale_Price = mean(Sale_Price))
```

```
## `summarise()` has grouped output by 'City'. You can override using the `.groups` argument.
```

Similarly, in terms of extending the R program, I used a new library called gganimate which I used in my Shiny app that had a layered animation in the line graph.

```
# output$linegraph <-renderImage({
#     Four_Cities <- read.csv(file ='Four_Cities.csv')
#
#     outfile <- tempfile(fileext='.gif')
#     q= ggplot(Four_Cities, aes(x = Year, y = Avg_Sale_Price, colour = City))+
#       geom_line(stat='identity')+ theme_bw() + transition_reveal(Year)
#
#     anim_save("linegraph.gif", animate(q,height=400,width=800,fps=20,duration=20,end_pause=60,res=120
#     list(src = "linegraph.gif",
#          contentType = 'image/gif'
#
#     )}, deleteFile = FALSE)
```

Conclusion of Objective 4

I believe I have performed all the task and I'm able to troubleshoot the error on my own. Also, I can now use new library functions and packages. Hence, I believe I have met the Objective4.

Objective5: Write clear, efficient, and well-documented R programs.

I have a good amount of knowledge in writing R codes and in RMD files. I have completed all my self reflection in RMD file where I have used the heading, bold and italics option. Additionally, I have attached the images in R Markdown. So, I believe I can write clear, efficient, organized and a well-documented R program.

Based on the progress you have made (i.e., see your response in (3)), what final grade would you give yourself for this course? Try to stick to the major grade levels ("A", "B", "C", or "D or below"). Please reach out to me if you have concerns or were unable to finish your final project.

My learning towards this course was always progressive through the class activities and my final projects. Hence, I would like to give myself an **"A"** as per the objectives I have met towards the course. The class activities helped me grasp the basic knowledge of R. Similarly, implementing all the learning into my project and implementing all the objectives enhanced my understanding of R in-depth.

Do you have any other thoughts or reflections about the course that you'd like to share?

The course and resources were really helpful for me. I could get back to the course material when I used to get an error. Also, I could revise it at all times. As of my experience, I really believed the activities really helps the student grasp the basic knowledge and it will help each student to apply it in their project. This course has enhanced my knowledge towards R.