**R and its Applications**

**A PROJECT REPORT**

*Submitted in fulfillment of*

*the requirement for the course of*

**Data Science and Analytics**

**IN**

**TERM - II**

*By*

**Adarsh Keshari**

**Roll No- 200201006**

*Submitted to-*

**Prof. (Dr.) Sridhar Vaithianathan,**

**Associate Professor (Analytics).**

**IMT - Hyderabad.**

**18th** **December 2020**

**Table of Contents-**

1. Basic Introduction of R
2. Assigning objects and Values
3. Doing Basic Arithmetic Operations
4. Data Types
5. Date and Naming Vectors
6. Vector, Matrices and Arrays
7. Data Frames
8. List in R
9. Basic Statistics
10. Standard Deviation
11. Variance
12. Correlation and Covariance
13. Creating Heat Maps
14. Plotting Graphs
15. Simple Linear Regression
16. Multiple Linear Regression

R-Codes Basic Intro and Hypothesis Testing

Adarsh Keshari

15/12/2020

Basic Operations and Shortcut in R

# Getting Working Directory  
getwd()

## [1] "E:/term 2/Data Science and analytics/R studio/7-DSA"

help()

## starting httpd help server ... done

# Ctrl+ enter- to run the command  
# Ctrl+(1,2,3,4)- to directly get into the command,console,environment and the files tabe in R Environment.

# Assigning objects and values

x=5  
y=10

# Doing basic arithemetic operations in R

x\*y # Basic Multiplication

## [1] 50

x/y # Basic Division

## [1] 0.5

x^y

## [1] 9765625

# Gives the value of x raised to the power y  
  
sqrt(x) # Gives the square root of x

## [1] 2.236068

exp(x) # Provides the exponential of x

## [1] 148.4132

log10(x)

## [1] 0.69897

x\*\*y # Gives the value of x raised to the power y

## [1] 9765625

# Assigning a different object  
a=x\*y  
a

## [1] 50

z<-x-y  
z

## [1] -5

class(a)

## [1] "numeric"

# The class function tells us about the Datatype.  
a= "Hello" # Assigning Characters to the variables  
a

## [1] "Hello"

class(a)

## [1] "character"

# Baisc Functions in R

# For Division we can also use the divider function  
divider = function(x,y){  
 result= x/y   
 print(result)   
}  
divider(50,25)

## [1] 2

# For multiplication also we can use the below function  
multiply= function(x,y){  
 result=x\*y  
 print(result)  
}  
multiply(4,3)

## [1] 12

multiply(10,30)

## [1] 300

# Various Data Types. (Nominal, Ordinal, Interval and Ratio)

# Self and System  
  
# Data Types  
x=5  
class(x)

## [1] "numeric"

# Numeric- It gives whether the assigned variables is either Integer(Whole Number) or Decimal(Float-Decimal)  
i= 10L # L- Here the L symbol denotes the Integer  
class(i)

## [1] "integer"

# Is function is used to know whether the asked command is True or False for the assigned variable.  
is.integer(i)

## [1] TRUE

is.numeric(x)

## [1] TRUE

is.numeric(i)

## [1] TRUE

# Character - Assigned the object as a Categorical Value   
s= "RStudio"  
class(s)

## [1] "character"

# Logical- TRUE has a value of (1) and FALSE has a value of (0) in R.  
  
TRUE \* 5 # same as 1 \*5

## [1] 5

FALSE \* 5 # same as 0\*5

## [1] 0

K= TRUE  
class(K)

## [1] "logical"

is.logical(K)

## [1] TRUE

# Date -

# In R the Starting Date is 1st Jan 1970  
# As function can be used to assigned value/operation to an object  
  
date1= as.Date("2020-11-20")  
date1

## [1] "2020-11-20"

class(date1)

## [1] "Date"

as.numeric(date1) # Gives the numeric vaue of the date asked for as each date is assigned with a numeric value.

## [1] 18586

#POSIXct - Gives Date and time together.  
date2 = as.POSIXct("2020-11-20 10:10")  
date2

## [1] "2020-11-20 10:10:00 IST"

as.numeric(date2)

## [1] 1605847200

class(date2)

## [1] "POSIXct" "POSIXt"

#Vector, Array and Matrices

#Vector. R is called as Vectorised Language.   
# A collection of elements, and all are of same types.  
# It cannot be of mixed type.  
# Arrays- n-Dimension COllection of Similar elements in terms of R  
#Matrices- Subset of Arrays. 2-d array is called matrix.  
# Will usually contain "numeric" value.  
  
v= c(1,2,3,4,5) # the c function is used for the combination of same type of elements together.  
  
#we can also do basic arithmetic operations in vectors.  
s=v\*2 # Each element is multiplied by same number.  
s

## [1] 2 4 6 8 10

s1= v/10 # Each element gets divided by 10  
s1

## [1] 0.1 0.2 0.3 0.4 0.5

s2= v^3 # each element gets cube of its number.  
s2

## [1] 1 8 27 64 125

# we can also do in the following way  
  
sqrt(v)

## [1] 1.000000 1.414214 1.732051 2.000000 2.236068

log10(v)

## [1] 0.0000000 0.3010300 0.4771213 0.6020600 0.6989700

#colon (:) used for operation-sequencing  
# Creates sequence of numbers in either direction.  
1:10 # shows number from 1-10

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

10:1

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

-2:7

## [1] -2 -1 0 1 2 3 4 5 6 7

# Two Vectors  
l=1:10  
m=-5:4  
  
# we can also do arithemetic operations in two vectors   
l+m

## [1] -4 -2 0 2 4 6 8 10 12 14

l-m

## [1] 6 6 6 6 6 6 6 6 6 6

# Baisc operations for getting LENGTH of vector

# We can also check the lenght of each vector by using the LENGTH function  
length(l)

## [1] 10

length(m)

## [1] 10

# Unequal length vectors  
l+c(1,2)

## [1] 2 4 4 6 6 8 8 10 10 12

# when the other vector is of unequal length then the number start getting repeated.  
l+c(1,2,3)

## Warning in l + c(1, 2, 3): longer object length is not a multiple of shorter  
## object length

## [1] 2 4 6 5 7 9 8 10 12 11

# We can also do comparison on vectors  
l<=5

## [1] TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE

l<y

## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE

# Vector comparison -"any" and "all"  
l1= 10:1  
m1= -4:5  
  
any(l1<m1) # Any function help us in comparing whether any of the element of the vector is satisfying the condition or not.

## [1] TRUE

any(l1>m1)

## [1] TRUE

anyDuplicated(l1)

## [1] 0

all(l1<m1) # All function help us in comparing whether all the element of the vector is satisfying the condition or not.

## [1] FALSE

# nchar function

# The "nchar" Gives the number of character in a wach word of the elements.  
q=c("Hockey","Football","Baseball","Rugby","Tennis")  
nchar(q)

## [1] 6 8 8 5 6

# Subscripting function

# Subscripting: Assesing "Individual element" in vector  
# Subscripting denoted by Square bracket or Square Bracket is used to target the selected element in the vector.  
  
x[1]

## [1] 5

x[7]

## [1] NA

q[3]

## [1] "Baseball"

q[1:3]

## [1] "Hockey" "Football" "Baseball"

# Provides the value of of 1st,2nd,and 3rd element.  
q[c(1,4)] # Provides the value of 1st and 4th element in the Q data sets.

## [1] "Hockey" "Rugby"

# Naming Vectors

# We can also give name to a vector.  
c(One="a", Two="b", Last="r")

## One Two Last   
## "a" "b" "r"

w=1:3  
names(w)=c("a","b","c")  
w

## a b c   
## 1 2 3

# Factor Vectors- OrdinalData  
  
q

## [1] "Hockey" "Football" "Baseball" "Rugby" "Tennis"

class(q)

## [1] "character"

# Converting elements to factor data type

# converting "q" to factors  
q\_F= as.factor(q) # assigning Factor to q  
q\_F

## [1] Hockey Football Baseball Rugby Tennis   
## Levels: Baseball Football Hockey Rugby Tennis

as.numeric(q\_F) # Assigning Numeric function to q\_F

## [1] 3 2 1 4 5

as.factor(q)

## [1] Hockey Football Baseball Rugby Tennis   
## Levels: Baseball Football Hockey Rugby Tennis

as.numeric(q)

## Warning: NAs introduced by coercion

## [1] NA NA NA NA NA

class(q)

## [1] "character"

class(q\_F)

## [1] "factor"

# R has two types of missing data- NA and NULL  
# NA= Actual Missing Value  
# NULL= Absence of anything.  
z=c(1,2,NA,8,3,NA,3)# R treats NA as an empty element and hence, NA is shown in the final output.  
z

## [1] 1 2 NA 8 3 NA 3

z=c(1,2,8,3,3) # R is a case sensitive platform and hence, na is treated differently as NA.  
z

## [1] 1 2 8 3 3

is.na(z)

## [1] FALSE FALSE FALSE FALSE FALSE

list(z)

## [[1]]  
## [1] 1 2 8 3 3

z\_char=c("Hockey",NA,"Cricket")  
z\_char

## [1] "Hockey" NA "Cricket"

is.na(z\_char) # here na function checks whether there is any NA element in the vector.

## [1] FALSE TRUE FALSE

# NULL and NA

# NULL   
z1=c(1,NULL,3)  
z1

## [1] 1 3

# R treats NULL as an empty cell and hence it doesn't consider it in the final output.  
x1=c(1,NA,3)  
x1

## [1] 1 NA 3

length(z)

## [1] 5

length(z1)

## [1] 2

length(x1)

## [1] 3

# Assigning NULL and checking  
  
d=NULL# Assigning D as a NULL element  
is.null(d)

## [1] TRUE

is.null(z1)

## [1] FALSE

is.null(x1)

## [1] FALSE

is.na(z1) # here the NULL element is not counted.

## [1] FALSE FALSE

is.na(x1)

## [1] FALSE TRUE FALSE

# Matrices

# Creating Matrices in R.  
  
A= matrix(1:10, nrow = 5) # Assigning A as a matrix with element ranging from 1-10 having 5 rows and 2 column.  
  
B= matrix(21:30, nrow = 5)# Assigning B as a matrix with element ranging from 21-30 having 5 rows and 2 column.  
  
C= matrix(21:40, nrow = 2)# Assigning C as a matrix with element ranging from 21-40 having 2 rows and 10 column.  
A

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

B

## [,1] [,2]  
## [1,] 21 26  
## [2,] 22 27  
## [3,] 23 28  
## [4,] 24 29  
## [5,] 25 30

C

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]  
## [1,] 21 23 25 27 29 31 33 35 37 39  
## [2,] 22 24 26 28 30 32 34 36 38 40

# We can do Arithmetic operations on matrices  
A+B

## [,1] [,2]  
## [1,] 22 32  
## [2,] 24 34  
## [3,] 26 36  
## [4,] 28 38  
## [5,] 30 40

A\*B

## [,1] [,2]  
## [1,] 21 156  
## [2,] 44 189  
## [3,] 69 224  
## [4,] 96 261  
## [5,] 125 300

A==B # Checking whether any of the element in Matrix A is equal to any element in Matrix B.

## [,1] [,2]  
## [1,] FALSE FALSE  
## [2,] FALSE FALSE  
## [3,] FALSE FALSE  
## [4,] FALSE FALSE  
## [5,] FALSE FALSE

A %\*% t(B) # the %\*% symbol is used for matrix multiplication in R and the t(B) means the transpose of the matrix B.

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 177 184 191 198 205  
## [2,] 224 233 242 251 260  
## [3,] 271 282 293 304 315  
## [4,] 318 331 344 357 370  
## [5,] 365 380 395 410 425

#Arrays

# Arrays are a multidimensional vector having all the elements of the same type.  
# Creating an Array   
theArray = array(1:20, dim=c(4,3,3))# The first 4in c represent the number of row in an array, the second number (3) represents the number of column in an array and the last number i.e., 3 denotes the number of outer dimensions that will be created.  
theArray

## , , 1  
##   
## [,1] [,2] [,3]  
## [1,] 1 5 9  
## [2,] 2 6 10  
## [3,] 3 7 11  
## [4,] 4 8 12  
##   
## , , 2  
##   
## [,1] [,2] [,3]  
## [1,] 13 17 1  
## [2,] 14 18 2  
## [3,] 15 19 3  
## [4,] 16 20 4  
##   
## , , 3  
##   
## [,1] [,2] [,3]  
## [1,] 5 9 13  
## [2,] 6 10 14  
## [3,] 7 11 15  
## [4,] 8 12 16

# If there is nothing written in the bracket in the Array function then it means that we are assessing the whole part of that array.  
theArray [1, ,]# Here we are accessing all the elements from Row 1, all the columns and all outer dimensions because we have only specifies which row to be considered and the other two components are not specified.

## [,1] [,2] [,3]  
## [1,] 1 13 5  
## [2,] 5 17 9  
## [3,] 9 1 13

theArray[1, ,1] # Here we are accessing all the elements from Row 1, all columns, first outer dimension

## [1] 1 5 9

theArray[, ,1] #Here we are accessing all rows, all columns but only the first outer dimension

## [,1] [,2] [,3]  
## [1,] 1 5 9  
## [2,] 2 6 10  
## [3,] 3 7 11  
## [4,] 4 8 12

# Data Frames

# Creating a Data frame   
  
l1=10:1  
m1=-4:5  
q= c("Hockey","Football","Baseball","Tennis","Curling","Badminton","Rugby","Soccer","Carom","Ludo")  
  
# We use the data.frame function to create a data frame or table combining various data sets.  
theDF= data.frame(l1,m1,q)  
theDF

## l1 m1 q  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Tennis  
## 5 6 0 Curling  
## 6 5 1 Badminton  
## 7 4 2 Rugby  
## 8 3 3 Soccer  
## 9 2 4 Carom  
## 10 1 5 Ludo

q= as.factor(q)  
q

## [1] Hockey Football Baseball Tennis Curling Badminton Rugby   
## [8] Soccer Carom Ludo   
## 10 Levels: Badminton Baseball Carom Curling Football Hockey Ludo ... Tennis

theDF=data.frame(First=l1, Second=m1, Sport=q) # Assigning names to the columns in the data frame.  
theDF

## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Tennis  
## 5 6 0 Curling  
## 6 5 1 Badminton  
## 7 4 2 Rugby  
## 8 3 3 Soccer  
## 9 2 4 Carom  
## 10 1 5 Ludo

# Str-Structure- Gives the whole structure of the data frame or table by explaining the total number of outcomes and also telling about the each columns seperately.  
  
str(theDF)

## 'data.frame': 10 obs. of 3 variables:  
## $ First : int 10 9 8 7 6 5 4 3 2 1  
## $ Second: int -4 -3 -2 -1 0 1 2 3 4 5  
## $ Sport : Factor w/ 10 levels "Badminton","Baseball",..: 6 5 2 10 4 1 8 9 3 7

# Checking the dimensions  
nrow(theDF) # Provides the number of Row in the Data Frame

## [1] 10

ncol(theDF) # Provides the number of columns in the Data Frame

## [1] 3

dim(theDF) # Provides the number of both rows and columns together in the Data Frame.

## [1] 10 3

# Gives the Columns name starting from the First Column  
names(theDF)

## [1] "First" "Second" "Sport"

# Gives only the Colun Heading of the Third Column as square bracked is always used to select a particular element.  
names(theDF)[3]

## [1] "Sport"

# HEAD and TAIL function

# Square bracket is always used for accessing a particular data   
  
# Head and Tail  
head(theDF) # Gives the First 6 rows with all variables

## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Tennis  
## 5 6 0 Curling  
## 6 5 1 Badminton

tail(theDF) # Gives the Last 6 rows with all variables

## First Second Sport  
## 5 6 0 Curling  
## 6 5 1 Badminton  
## 7 4 2 Rugby  
## 8 3 3 Soccer  
## 9 2 4 Carom  
## 10 1 5 Ludo

tail(theDF, n=2) # It gives only the last 2 rows with all variables of the data set.

## First Second Sport  
## 9 2 4 Carom  
## 10 1 5 Ludo

class(theDF)

## [1] "data.frame"

# Accessing individual column   
# For accessing individual column we use the ($) Dollar sign.  
theDF$Sport # Here we only want the data variables of only the Sport column so we used the $ sign. Also, it provides the Levels and this level is organized in alphabetical order with only unique variables.

## [1] Hockey Football Baseball Tennis Curling Badminton Rugby   
## [8] Soccer Carom Ludo   
## 10 Levels: Badminton Baseball Carom Curling Football Hockey Ludo ... Tennis

# LIST Function

# Lists - This function is used to store any number of items of any type and can contain either numeric or characters.  
  
# By using the "list" function we make each argument in "list" to become the element of the list.  
  
list(1,2,3)# Here we are creating a three element list

## [[1]]  
## [1] 1  
##   
## [[2]]  
## [1] 2  
##   
## [[3]]  
## [1] 3

list1=(c(1,2,3))# Here we are creating a single vector element which contains Three elements.  
list1

## [1] 1 2 3

list2 = list(c(1,2,3), 3:7) # Creating a two element list with one element having 3 vector elements   
list2

## [[1]]  
## [1] 1 2 3  
##   
## [[2]]  
## [1] 3 4 5 6 7

# We can also create a two element list combining a date frame and a vector.  
list(theDF, 1:10) # Here theDF is a Data.frame that we have created above and the next is a vector with 10 elements.

## [[1]]  
## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Tennis  
## 5 6 0 Curling  
## 6 5 1 Badminton  
## 7 4 2 Rugby  
## 8 3 3 Soccer  
## 9 2 4 Carom  
## 10 1 5 Ludo  
##   
## [[2]]  
## [1] 1 2 3 4 5 6 7 8 9 10

# We can also create a list of elements by combining anotherr list in the data.  
list3 = list(theDF, 1:10, list1)  
list3

## [[1]]  
## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Tennis  
## 5 6 0 Curling  
## 6 5 1 Badminton  
## 7 4 2 Rugby  
## 8 3 3 Soccer  
## 9 2 4 Carom  
## 10 1 5 Ludo  
##   
## [[2]]  
## [1] 1 2 3 4 5 6 7 8 9 10  
##   
## [[3]]  
## [1] 1 2 3

#We can also name the list in R  
names(list3)= c("data.frame", "vector","list")  
names(list3)

## [1] "data.frame" "vector" "list"

list3

## $data.frame  
## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Tennis  
## 5 6 0 Curling  
## 6 5 1 Badminton  
## 7 4 2 Rugby  
## 8 3 3 Soccer  
## 9 2 4 Carom  
## 10 1 5 Ludo  
##   
## $vector  
## [1] 1 2 3 4 5 6 7 8 9 10  
##   
## $list  
## [1] 1 2 3

# Accessing individual element of a list   
# We use Double Square Brackets[] to asses the individual elements in any list by specifying the name or element type inside the square bracket.  
  
list3[[1]] # Here we are assessing the first element i.e., the dataframe of the list 3 by specifying the position of the element.

## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Tennis  
## 5 6 0 Curling  
## 6 5 1 Badminton  
## 7 4 2 Rugby  
## 8 3 3 Soccer  
## 9 2 4 Carom  
## 10 1 5 Ludo

list3[["data.frame"]] # We can also assess the element of a list by specifying the element name in the list.

## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Tennis  
## 5 6 0 Curling  
## 6 5 1 Badminton  
## 7 4 2 Rugby  
## 8 3 3 Soccer  
## 9 2 4 Carom  
## 10 1 5 Ludo

# Also we can assess any particular field of any specific field in a list by using dollar sign inside a square brackets.   
list3[[1]]$Sport

## [1] Hockey Football Baseball Tennis Curling Badminton Rugby   
## [8] Soccer Carom Ludo   
## 10 Levels: Badminton Baseball Carom Curling Football Hockey Ludo ... Tennis

# We can find the Length of the list and the heading of the different elements in the list by using the length and names function respectively in R.  
length(list3)

## [1] 3

names(list3)

## [1] "data.frame" "vector" "list"

# Reading data into R

# Reading Data into R  
  
# we can directly read data from any website into R( The Common Source File or CSV File)  
  
theUrl = "http://www.jaredlander.com/data/Tomato%20First.csv"  
tomato = read.table(file=theUrl, header=TRUE, sep =",")  
head(tomato)

## Round Tomato Price Source Sweet Acid Color Texture Overall  
## 1 1 Simpson SM 3.99 Whole Foods 2.8 2.8 3.7 3.4 3.4  
## 2 1 Tuttorosso (blue) 2.99 Pioneer 3.3 2.8 3.4 3.0 2.9  
## 3 1 Tuttorosso (green) 0.99 Pioneer 2.8 2.6 3.3 2.8 2.9  
## 4 1 La Fede SM DOP 3.99 Shop Rite 2.6 2.8 3.0 2.3 2.8  
## 5 2 Cento SM DOP 5.49 D Agostino 3.3 3.1 2.9 2.8 3.1  
## 6 2 Cento Organic 4.99 D Agostino 3.2 2.9 2.9 3.1 2.9  
## Avg.of.Totals Total.of.Avg  
## 1 16.1 16.1  
## 2 15.3 15.3  
## 3 14.3 14.3  
## 4 13.4 13.4  
## 5 14.4 15.2  
## 6 15.5 15.1

# It is good to use.csv file.  
# We can't directly use excel files in R so, we have to first convert the excel file into a .csv file to bring that excel file into R.

# we can also asess the buit in Data sets in R

data() # Provides the list of all the Data sets installed in the R in my system.

data(mtcars) # Assessing the mtcars Data set from the R directory.

head(mtcars)

## mpg cyl disp hp drat wt qsec vs am gear carb  
## Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4  
## Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4  
## Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1  
## Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1  
## Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2  
## Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

tail(mtcars)

## mpg cyl disp hp drat wt qsec vs am gear carb  
## Porsche 914-2 26.0 4 120.3 91 4.43 2.140 16.7 0 1 5 2  
## Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.9 1 1 5 2  
## Ford Pantera L 15.8 8 351.0 264 4.22 3.170 14.5 0 1 5 4  
## Ferrari Dino 19.7 6 145.0 175 3.62 2.770 15.5 0 1 5 6  
## Maserati Bora 15.0 8 301.0 335 3.54 3.570 14.6 0 1 5 8  
## Volvo 142E 21.4 4 121.0 109 4.11 2.780 18.6 1 1 4 2

#Statistics Basic

# The basic statistics consists of Mean, Variances,Correlations and T-tests  
  
# Generating a random sample of 100 numbers between 1 and 50  
x2 = sample(x =1:50, size = 20, replace = FALSE)  
x2 # Here the xs will generate an output consisting of 20 elements which are randomly selected from a data set of 50 numbers.

## [1] 37 21 7 45 28 11 23 1 43 17 46 49 3 6 35 31 15 50 40 36

# Simple Arithmetic Mean  
mean(x2)

## [1] 27.2

# Calculating means when there is a missing NA data in the data sets  
y2= x2 # copy x to y  
y2[sample(x=1:50, size = 20, replace = FALSE )] = NA # Null Values  
y2 # The output will show NA in place of missing value.

## [1] NA NA NA 45 NA 11 23 NA NA NA 46 NA 3 6 35 31 15 NA 40 36 NA NA NA NA NA  
## [26] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA

mean(y2) # In finding the mean of a missing value we will get NA as output because we are unable to get the mean because of missing value in Data.

## [1] NA

# We can remove the missing value from the above data to get the mean.  
  
mean(y2,na.rm=TRUE) # We will use the rm function to remove the missing value. Hence, now there will be an output.

## [1] 26.45455

# We can also do a Weighted Mean in R  
Grades = c(95,72,87,66)  
Weights = c(1/2, 1/4, 1/8, 1/8)  
mean(Grades)# Simple Arithmetic mean

## [1] 80

weighted.mean(x=Grades,w=Weights)# The Weighted Mean of Grades and Weights.

## [1] 84.625

#Variance

# we can also find out the variance of any data in R  
var(x2)

## [1] 264.6947

# Standard Deviation

# we can find the standard deviation by using either sqrt the variance method or by directly using the standard deviation formula in R.  
sqrt(var(x2))

## [1] 16.26944

sd(x2) # The standard deviation of x2

## [1] 16.26944

sd(y2)# The standard deviation of y2. It is showing NA because there is a missing value in the y2 data.

## [1] NA

sd(y2, na.rm=TRUE) # We can remove the missing value from y2 to get an actual dtandard deviation in output.

## [1] 15.63562

# Some of the other Commonly Used Functions in R  
min(x2) # Used for generating the minimum value of the element in the data set.

## [1] 1

max(x2)# Used for generating the maximum value of the element in the data set.

## [1] 50

median(x2) # To find the median of the data set.

## [1] 29.5

min(y2) # Here we are getting output as NA because there is a missing value.

## [1] NA

min(y2, na.rm=TRUE) # We remove the missing value to get the final output.

## [1] 3

# The Summary Statistics Function is used for getting the overall summary of the Data sets. This function gives the min, max, median, and the quantiles in the Data set.  
summary(x2)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.00 14.00 29.50 27.20 40.75 50.00

summary(y2)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 3.00 13.00 31.00 26.45 38.00 46.00 39

# Quantile is a function used for generating the quantile in R  
quantile(x2, probs = c(0.25, 0.45)) # Calculating 25th and 45th Quantile in the Data set.

## 25% 45%   
## 14.00 25.75

quantile(x2, probs = c(0.1,0.25,0.5, 0.75,0.99)) # We can also find more than 2 quantiles together.

## 10% 25% 50% 75% 99%   
## 5.70 14.00 29.50 40.75 49.81

quantile(x2, probs = c(0.25, 0.75)) # Calculate 25th and 45th Quantile. Here we are getting NA because of the missing value.

## 25% 75%   
## 14.00 40.75

quantile(y2, probs = c(0.25, 0.75), na.rm = TRUE) # Hence to get the actual output we remove the missing values by using the rm function.

## 25% 75%   
## 13 38

# Correlation and Covariance

install.packages(“ggplot2”)

library(ggplot2)# require(ggplot2)

head(economics)# Generating the first 6 element of the "ggplot2 database that we installed above.

## # A tibble: 6 x 6  
## date pce pop psavert uempmed unemploy  
## <date> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1967-07-01 507. 198712 12.6 4.5 2944  
## 2 1967-08-01 510. 198911 12.6 4.7 2945  
## 3 1967-09-01 516. 199113 11.9 4.6 2958  
## 4 1967-10-01 512. 199311 12.9 4.9 3143  
## 5 1967-11-01 517. 199498 12.8 4.7 3066  
## 6 1967-12-01 525. 199657 11.8 4.8 3018

# To do correlation we use the cor function.  
# HEre we are doing a correlation between the PCE and PSAVERT which are obtained from the"ggplot2" package.  
cor(economics$pce, economics$psavert) #pce-Personal Consumption Expenditure;psavert -Personal Savings Rate

## [1] -0.7928546

# To compare correlation for Multiple variables

# We can also do a correlation for multiple variables by specifying the column numbers by using the combination function.  
cor(economics[, c(2,4:6)])

## pce psavert uempmed unemploy  
## pce 1.0000000 -0.7928546 0.7269616 0.6145176  
## psavert -0.7928546 1.0000000 -0.3251377 -0.3093769  
## uempmed 0.7269616 -0.3251377 1.0000000 0.8693097  
## unemploy 0.6145176 -0.3093769 0.8693097 1.0000000

pce psavert uempmed unemploy

pce 1.0000000 -0.7928546 0.7269616 0.6145176 psavert -0.7928546 1.0000000 -0.3251377 -0.3093769 uempmed 0.7269616 -0.3251377 1.0000000 0.8693097 unemploy 0.6145176 -0.3093769 0.8693097 1.0000000

# Display Correlation in Different Format!  
  
# Lets install the required package and load them onto this R environment for executing!!!  
  
# Load the "reshape" package  
require(reshape2)

## Loading required package: reshape2

# Also load the Scales package for some extra plotting features  
# I havealready instally scales package in my system   
library(scales)  
  
econCor = cor(economics [ , c(2,4:6)])

econMelt = melt(econCor, varnames = c("x3" ,"y3"), value.name = "Correlation")  
# Order it according to correlation

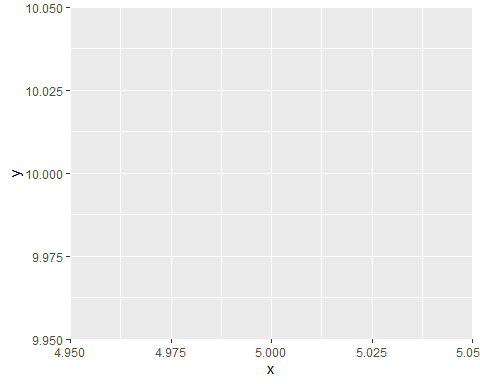
econMelt = econMelt[order(econMelt$Correlation),]  
# Display the melted data  
econMelt

## x3 y3 Correlation  
## 2 psavert pce -0.7928546  
## 5 pce psavert -0.7928546  
## 7 uempmed psavert -0.3251377  
## 10 psavert uempmed -0.3251377  
## 8 unemploy psavert -0.3093769  
## 14 psavert unemploy -0.3093769  
## 4 unemploy pce 0.6145176  
## 13 pce unemploy 0.6145176  
## 3 uempmed pce 0.7269616  
## 9 pce uempmed 0.7269616  
## 12 unemploy uempmed 0.8693097  
## 15 uempmed unemploy 0.8693097  
## 1 pce pce 1.0000000  
## 6 psavert psavert 1.0000000  
## 11 uempmed uempmed 1.0000000  
## 16 unemploy unemploy 1.0000000

x y Correlation

2 psavert pce -0.7928546 5 pce psavert -0.7928546 7 uempmed psavert -0.3251377 10 psavert uempmed -0.3251377 8 unemploy psavert -0.3093769 14 psavert unemploy -0.3093769 4 unemploy pce 0.6145176 13 pce unemploy 0.6145176 3 uempmed pce 0.7269616 9 pce uempmed 0.7269616 12 unemploy uempmed 0.8693097 15 uempmed unemploy 0.8693097 1 pce pce 1.0000000 6 psavert psavert 1.0000000 11 uempmed uempmed 1.0000000 16 unemploy unemploy 1.0000000

# Let's Visualize Correlation  
## Plot it with ggplot  
# Initialize the plot with x and y on the respective axes  
ggplot(econMelt,aes (x=x, y=y),geom\_tile(aes(fill = Correlation)),scale\_fill\_gradient2(low = muted("red"), mid = "white", high = "steelblue",guide = guide\_colorbar(ticks=FALSE, barheight=10), limit=c(-1,1), theme\_minimal(), labs(x= NULL, y=NULL)))



# Correlation

# Data Creation- The first step in correlation  
mydata <- mtcars[, c(1,3,4,5,6,7)] # Getting data from the mtcars files package of R  
head(mydata)

## mpg disp hp drat wt qsec  
## Mazda RX4 21.0 160 110 3.90 2.620 16.46  
## Mazda RX4 Wag 21.0 160 110 3.90 2.875 17.02  
## Datsun 710 22.8 108 93 3.85 2.320 18.61  
## Hornet 4 Drive 21.4 258 110 3.08 3.215 19.44  
## Hornet Sportabout 18.7 360 175 3.15 3.440 17.02  
## Valiant 18.1 225 105 2.76 3.460 20.22

# Finding the Correlation- The Second Step.  
cormat <- round(cor(mydata),2)  
head(cormat)

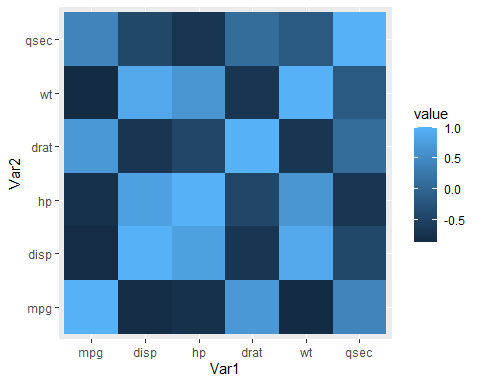
## mpg disp hp drat wt qsec  
## mpg 1.00 -0.85 -0.78 0.68 -0.87 0.42  
## disp -0.85 1.00 0.79 -0.71 0.89 -0.43  
## hp -0.78 0.79 1.00 -0.45 0.66 -0.71  
## drat 0.68 -0.71 -0.45 1.00 -0.71 0.09  
## wt -0.87 0.89 0.66 -0.71 1.00 -0.17  
## qsec 0.42 -0.43 -0.71 0.09 -0.17 1.00

# Creating the correlation heatmap using the ggplot2  
install.packages(“reshape2”)

library(reshape2)  
melted\_cormat <- melt(cormat)  
head(melted\_cormat)

## Var1 Var2 value  
## 1 mpg mpg 1.00  
## 2 disp mpg -0.85  
## 3 hp mpg -0.78  
## 4 drat mpg 0.68  
## 5 wt mpg -0.87  
## 6 qsec mpg 0.42

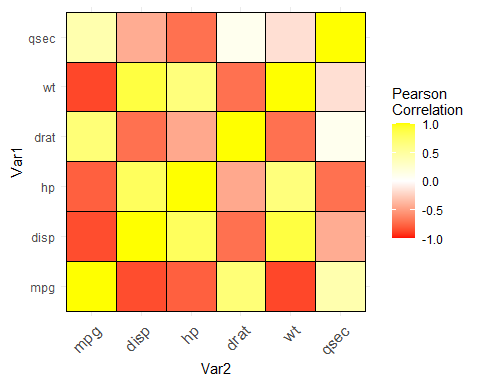
# Creating the graph for the selected data  
#The function geom\_tile()[ggplot2 package] is used for correlation matrix.  
library(ggplot2)  
ggplot(data = melted\_cormat, aes(x=Var1, y=Var2, fill=value)) +   
 geom\_tile()



# The above graph has many redundant information. So we will use NA function to set some of it to show NA.  
  
# Getting the lower triangle of the correlation matrix  
get\_lower\_tri<-function(cormat){  
 cormat[upper.tri(cormat)] <- NA  
 return(cormat)  
}  
  
# Getting the upper triangle of the correlation matrix  
get\_upper\_tri <- function(cormat){  
 cormat[lower.tri(cormat)]<- NA  
 return(cormat)  
}

upper\_tri <- get\_upper\_tri(cormat) upper\_tri

# The last task of finishing the correlation matrix heatmap.  
# We will remove the rows with the NA value and would show only those rows which contain some value.  
# Melt the correlation matrix  
library(reshape2)  
  
# Now we will create the Heatmap for the above data.  
library(ggplot2)  
ggplot(data = melted\_cormat, aes(Var2, Var1, fill = value))+  
 geom\_tile(color = "black")+  
 scale\_fill\_gradient2(low = "red", high = "yellow", mid = "white",   
 midpoint = 0, limit = c(-1,1), space = "Lab",   
 name="Pearson\nCorrelation") +  
 theme\_minimal()+   
 theme(axis.text.x = element\_text(angle = 45, vjust = 1,   
 size = 12, hjust = 1))+  
 coord\_fixed()

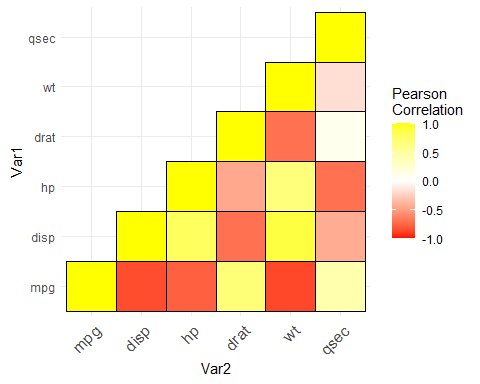


# In the output graph we can see that the negative correlation is shown by the red colour and the positive one are in yellow colour.

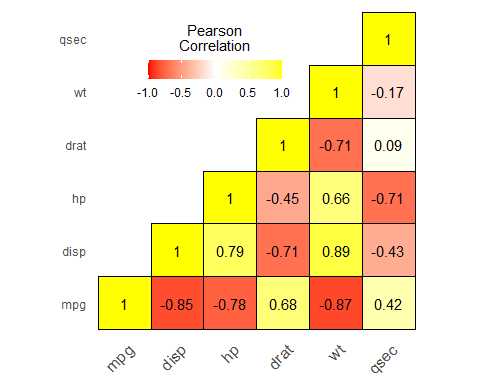
#We use the coord\_fixed() function to make one unit of x-axis equal to one unit of y axis  
# Reordering the correlation matrix  
# We use the reorder\_courmat function to reorder the correlation Matrix in R.  
reorder\_cormat <- function(cormat){  
 dd <- as.dist((1-cormat)/2)  
 hc <- hclust(dd)  
 cormat <-cormat[hc$order, hc$order]  
}  
   
upper\_tri <- get\_upper\_tri(cormat)  
# Melt the correlation matrix  
melted\_cormat <- melt(upper\_tri, na.rm = TRUE)

# Creating a ggheatmap

# We use the ggheatmap function in R to create a heat map. We select the package data and along with that we can also assign which which varaiables of the data we want to select and we can also specify the colours of the correlation in the graph.  
ggheatmap <- ggplot(melted\_cormat, aes(Var2, Var1, fill = value))+  
 geom\_tile(color = "black")+  
 scale\_fill\_gradient2(low = "red", high = "yellow", mid = "white",   
 midpoint = 0, limit = c(-1,1), space = "Lab",   
 name="Pearson\nCorrelation") +  
 theme\_minimal()+ # minimal theme  
 theme(axis.text.x = element\_text(angle = 45, vjust = 1,   
 size = 12, hjust = 1))+  
 coord\_fixed()  
  
# Print the heatmap  
print(ggheatmap)



#Add correlation coefficients on the heatmap  
# We use geom\_text() funtion to write coefficient on the heatmap.  
  
ggheatmap +   
 geom\_text(aes(Var2, Var1, label = value), color = "black", size = 4) +  
 theme(  
 axis.title.x = element\_blank(),  
 axis.title.y = element\_blank(),  
 panel.grid.major = element\_blank(),  
 panel.border = element\_blank(),  
 panel.background = element\_blank(),  
 axis.ticks = element\_blank(),  
 legend.justification = c(1, 0),  
 legend.position = c(0.6, 0.7),  
 legend.direction = "horizontal")+  
 guides(fill = guide\_colorbar(barwidth = 7, barheight = 1,  
 title.position = "top", title.hjust = 0.5))



# Here we are creating a Heatmap showing a correlation between multiple variables and alsong with that we are also highlighting the each block of the Data on the graph.

# T-Test

# We are selecting the tips dta of a restaurant from the reshape2 package of the R. We have installed this package before using the install.package function.  
data(tips, package = "reshape2") # We are making a T-test to analyse about the tips offered in the restaurant.  
head(tips) # We are asking for the first ^ data of the reshape2 package

## total\_bill tip sex smoker day time size  
## 1 16.99 1.01 Female No Sun Dinner 2  
## 2 10.34 1.66 Male No Sun Dinner 3  
## 3 21.01 3.50 Male No Sun Dinner 3  
## 4 23.68 3.31 Male No Sun Dinner 2  
## 5 24.59 3.61 Female No Sun Dinner 4  
## 6 25.29 4.71 Male No Sun Dinner 4

str(tips) # To get the structure of the tip that whether the data is numeric or factor etc. anf to know about a basic summary of the whole data.

## 'data.frame': 244 obs. of 7 variables:  
## $ total\_bill: num 17 10.3 21 23.7 24.6 ...  
## $ tip : num 1.01 1.66 3.5 3.31 3.61 4.71 2 3.12 1.96 3.23 ...  
## $ sex : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 2 2 2 2 2 ...  
## $ smoker : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ day : Factor w/ 4 levels "Fri","Sat","Sun",..: 3 3 3 3 3 3 3 3 3 3 ...  
## $ time : Factor w/ 2 levels "Dinner","Lunch": 1 1 1 1 1 1 1 1 1 1 ...  
## $ size : int 2 3 3 2 4 4 2 4 2 2 ...

# Gender # We are only asking for the gender from the data. This gives us the total number of unique varaibles in the data.  
unique(tips$sex)

## [1] Female Male   
## Levels: Female Male

#One Sample t-test - Cosnsidering only a single group and performing a two tail test with a NULL hypothesis that the mean of the sample is equal to 2.5  
t.test(tips$tip, alternative = "two.sided", mu=2.5)

##   
## One Sample t-test  
##   
## data: tips$tip  
## t = 5.6253, df = 243, p-value = 5.08e-08  
## alternative hypothesis: true mean is not equal to 2.5  
## 95 percent confidence interval:  
## 2.823799 3.172758  
## sample estimates:  
## mean of x   
## 2.998279

#One Sample t-test - Upper Tail. Ho:Mean LE 2.5. Considering that we are doing a upper tail test means that the alternate hypotheis will be either greater than than mean.   
t.test(tips$tip, alternative = "greater", mu=2.5)

##   
## One Sample t-test  
##   
## data: tips$tip  
## t = 5.6253, df = 243, p-value = 2.54e-08  
## alternative hypothesis: true mean is greater than 2.5  
## 95 percent confidence interval:  
## 2.852023 Inf  
## sample estimates:  
## mean of x   
## 2.998279

# Here the p value is very less than the

# Two Sample T-test with having Two groups.  
t.test(tip ~ sex, data = tips, var.equal = TRUE)

##   
## Two Sample t-test  
##   
## data: tip by sex  
## t = -1.3879, df = 242, p-value = 0.1665  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -0.6197558 0.1074167  
## sample estimates:  
## mean in group Female mean in group Male   
## 2.833448 3.089618

# HEre in the below data we can find that the mean of both male and female separately and we can find that the p-value is greater than alpha value so we will accept the null hypothesis.

#Paired Two-Sample T-Test   
# here we are considering the 2 Data sets from the same package.  
  
require(UsingR)

## Loading required package: UsingR

## Loading required package: MASS

## Loading required package: HistData

## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

##   
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':  
##   
## format.pval, units

##   
## Attaching package: 'UsingR'

## The following object is masked from 'package:survival':  
##   
## cancer

head(father.son)

## fheight sheight  
## 1 65.04851 59.77827  
## 2 63.25094 63.21404  
## 3 64.95532 63.34242  
## 4 65.75250 62.79238  
## 5 61.13723 64.28113  
## 6 63.02254 64.24221

write.csv(father.son, "E:/term 2/Data Science and analytics/R studio/7-DSA/father\_son.csv", row.names = FALSE) # we use wite.csv function to write the selected data from the package in our drive in .csv file format.

#ANOVA - Comparing Multiple Groups

# We can also do ANNOVA in R by using the aov() function.  
# We are considering the same package and the same dta set from the above.   
# We are finding the tip obtained in the day from different genders and their variance around all days of the week.  
str(tips)

## 'data.frame': 244 obs. of 7 variables:  
## $ total\_bill: num 17 10.3 21 23.7 24.6 ...  
## $ tip : num 1.01 1.66 3.5 3.31 3.61 4.71 2 3.12 1.96 3.23 ...  
## $ sex : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 2 2 2 2 2 ...  
## $ smoker : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ day : Factor w/ 4 levels "Fri","Sat","Sun",..: 3 3 3 3 3 3 3 3 3 3 ...  
## $ time : Factor w/ 2 levels "Dinner","Lunch": 1 1 1 1 1 1 1 1 1 1 ...  
## $ size : int 2 3 3 2 4 4 2 4 2 2 ...

tipAnova = aov(tip ~ day, tips)  
summary(tipAnova)

## Df Sum Sq Mean Sq F value Pr(>F)  
## day 3 9.5 3.175 1.672 0.174  
## Residuals 240 455.7 1.899

# Here, we get the ANNOVA results and this shows us the F value which we can use to determine whether the hypothesis will be accpeted or not.

# Simple Linear Regression

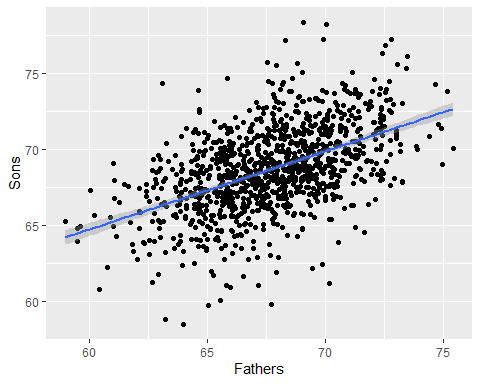
# HEre also we are considering the same data set from above i.e., the father son height relation.  
# We will use the simple linear regression to find our or do a prediction of the son's height using the data of the father's height.  
require(UsingR)  
require(ggplot2) #COnsidering the ggplot2 package  
head(father.son) # Selecting the father.son data

## fheight sheight  
## 1 65.04851 59.77827  
## 2 63.25094 63.21404  
## 3 64.95532 63.34242  
## 4 65.75250 62.79238  
## 5 61.13723 64.28113  
## 6 63.02254 64.24221

# We get the top 6 data of the height of the father and the son.

# Now we will plot the data of the height of the father and son on a graph to get a basic understanding about the relation of the father and son height.  
ggplot(father.son, aes(x=fheight, y=sheight))+geom\_point()+  
 geom\_smooth(method="lm")+labs(x="Fathers", y="Sons")

## `geom\_smooth()` using formula 'y ~ x'



# Now we will do a regression of the height data of the father and son to know about the coefficient of the intercept and the father's height.  
heightsLM = lm(sheight ~ fheight, data = father.son)  
heightsLM

##   
## Call:  
## lm(formula = sheight ~ fheight, data = father.son)  
##   
## Coefficients:  
## (Intercept) fheight   
## 33.8866 0.5141

summary(heightsLM)

##   
## Call:  
## lm(formula = sheight ~ fheight, data = father.son)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -8.8772 -1.5144 -0.0079 1.6285 8.9685   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 33.88660 1.83235 18.49 <2e-16 \*\*\*  
## fheight 0.51409 0.02705 19.01 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.437 on 1076 degrees of freedom  
## Multiple R-squared: 0.2513, Adjusted R-squared: 0.2506   
## F-statistic: 361.2 on 1 and 1076 DF, p-value: < 2.2e-16

# Now we can see that there is a big difference in the height of the father and the son and hence, we will not be able to correctly predict the height of the son using the data of father's height.

# Multiple Linear Regression

# We will download the housing data by providing the URL of the data.  
housing = read.table("http://www.jaredlander.com/data/housing.csv", sep =",", header = TRUE, stringsAsFactors=FALSE)  
  
write.table(housing, "Shousing.csv", col.names = TRUE,row.names = FALSE, quote = FALSE, sep =",")

str(housing)

## 'data.frame': 2626 obs. of 13 variables:  
## $ Neighborhood : chr "FINANCIAL" "FINANCIAL" "FINANCIAL" "FINANCIAL" ...  
## $ Building.Classification: chr "R9-CONDOMINIUM" "R4-CONDOMINIUM" "RR-CONDOMINIUM" "R4-CONDOMINIUM" ...  
## $ Total.Units : int 42 78 500 282 239 133 109 107 247 121 ...  
## $ Year.Built : int 1920 1985 NA 1930 1985 1986 1985 1986 1987 1985 ...  
## $ Gross.SqFt : int 36500 126420 554174 249076 219495 139719 105000 87479 255845 106129 ...  
## $ Estimated.Gross.Income : int 1332615 6633257 17310000 11776313 10004582 5127687 4365900 3637377 11246946 4115683 ...  
## $ Gross.Income.per.SqFt : num 36.5 52.5 31.2 47.3 45.6 ...  
## $ Estimated.Expense : int 342005 1762295 3543000 2784670 2783197 1497788 1273650 1061120 2440761 1231096 ...  
## $ Expense.per.SqFt : num 9.37 13.94 6.39 11.18 12.68 ...  
## $ Net.Operating.Income : int 990610 4870962 13767000 8991643 7221385 3629899 3092250 2576257 8806185 2884587 ...  
## $ Full.Market.Value : int 7300000 30690000 90970000 67556006 54320996 26737996 22210281 19449002 66316999 21821999 ...  
## $ Market.Value.per.SqFt : num 200 243 164 271 247 ...  
## $ Boro : chr "Manhattan" "Manhattan" "Manhattan" "Manhattan" ...

# We are only selecting the Borough column in the entire data set and we convert that convert as a factor in   
housing$Boro= as.factor(housing$Boro)  
  
str(housing)

## 'data.frame': 2626 obs. of 13 variables:  
## $ Neighborhood : chr "FINANCIAL" "FINANCIAL" "FINANCIAL" "FINANCIAL" ...  
## $ Building.Classification: chr "R9-CONDOMINIUM" "R4-CONDOMINIUM" "RR-CONDOMINIUM" "R4-CONDOMINIUM" ...  
## $ Total.Units : int 42 78 500 282 239 133 109 107 247 121 ...  
## $ Year.Built : int 1920 1985 NA 1930 1985 1986 1985 1986 1987 1985 ...  
## $ Gross.SqFt : int 36500 126420 554174 249076 219495 139719 105000 87479 255845 106129 ...  
## $ Estimated.Gross.Income : int 1332615 6633257 17310000 11776313 10004582 5127687 4365900 3637377 11246946 4115683 ...  
## $ Gross.Income.per.SqFt : num 36.5 52.5 31.2 47.3 45.6 ...  
## $ Estimated.Expense : int 342005 1762295 3543000 2784670 2783197 1497788 1273650 1061120 2440761 1231096 ...  
## $ Expense.per.SqFt : num 9.37 13.94 6.39 11.18 12.68 ...  
## $ Net.Operating.Income : int 990610 4870962 13767000 8991643 7221385 3629899 3092250 2576257 8806185 2884587 ...  
## $ Full.Market.Value : int 7300000 30690000 90970000 67556006 54320996 26737996 22210281 19449002 66316999 21821999 ...  
## $ Market.Value.per.SqFt : num 200 243 164 271 247 ...  
## $ Boro : Factor w/ 5 levels "Bronx","Brooklyn",..: 3 3 3 3 3 3 3 3 3 3 ...

# Hence, we can now see that now the Boro row is showing as a factor Data type instead of the Character Data type.

# Now we are only interested in the Boro Data set. So we will select only that coloumn or data set using the $ symbol.  
# ALso, we are only interested in the unique value in the Boro Data set so we will use the unique function.  
unique(housing$Boro)

## [1] Manhattan Brooklyn Queens Bronx Staten Island  
## Levels: Bronx Brooklyn Manhattan Queens Staten Island

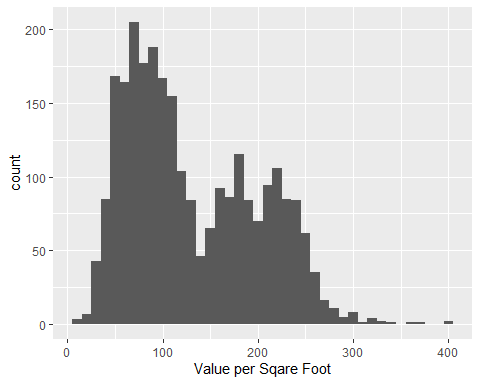
head(housing)

## Neighborhood Building.Classification Total.Units Year.Built Gross.SqFt  
## 1 FINANCIAL R9-CONDOMINIUM 42 1920 36500  
## 2 FINANCIAL R4-CONDOMINIUM 78 1985 126420  
## 3 FINANCIAL RR-CONDOMINIUM 500 NA 554174  
## 4 FINANCIAL R4-CONDOMINIUM 282 1930 249076  
## 5 TRIBECA R4-CONDOMINIUM 239 1985 219495  
## 6 TRIBECA R4-CONDOMINIUM 133 1986 139719  
## Estimated.Gross.Income Gross.Income.per.SqFt Estimated.Expense  
## 1 1332615 36.51 342005  
## 2 6633257 52.47 1762295  
## 3 17310000 31.24 3543000  
## 4 11776313 47.28 2784670  
## 5 10004582 45.58 2783197  
## 6 5127687 36.70 1497788  
## Expense.per.SqFt Net.Operating.Income Full.Market.Value Market.Value.per.SqFt  
## 1 9.37 990610 7300000 200.00  
## 2 13.94 4870962 30690000 242.76  
## 3 6.39 13767000 90970000 164.15  
## 4 11.18 8991643 67556006 271.23  
## 5 12.68 7221385 54320996 247.48  
## 6 10.72 3629899 26737996 191.37  
## Boro  
## 1 Manhattan  
## 2 Manhattan  
## 3 Manhattan  
## 4 Manhattan  
## 5 Manhattan  
## 6 Manhattan

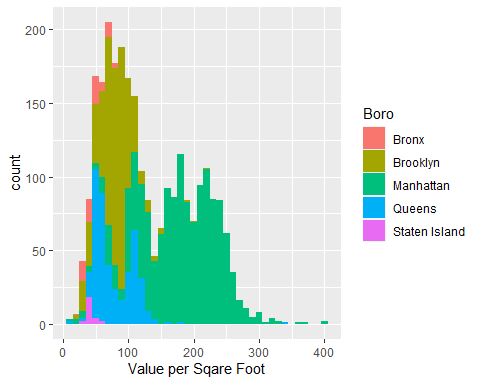
# We will now assign different names to the different columns by writing the names in a sequential order as this will make the first nae that we write to appear on the frst column.  
names(housing)=c("Neighborhood","Class", "Units", "YearBuilt", "SqFt", "Income", "IncomePerSqFt", "Expense", "ExpensePerSqFt", "NetIncome", "Value", "ValuePerSqFt", "Boro")  
  
  
tail(housing)

## Neighborhood Class Units YearBuilt SqFt Income  
## 2621 NEW SPRINGVILLE R4-CONDOMINIUM 37 NA 47880 673193  
## 2622 ROSEBANK R4-CONDOMINIUM 52 NA 62391 831672  
## 2623 ARROCHAR-SHORE ACRES R4-CONDOMINIUM 102 1987 90618 1274089  
## 2624 GRANT CITY R4-CONDOMINIUM 100 1986 78903 1321625  
## 2625 GRANT CITY R4-CONDOMINIUM 159 1961 166712 2343971  
## 2626 GREAT KILLS R4-CONDOMINIUM 67 1965 108864 1298748  
## IncomePerSqFt Expense ExpensePerSqFt NetIncome Value ValuePerSqFt  
## 2621 14.06 336596 7.03 336597 2115260 44.18  
## 2622 13.33 326305 5.23 505367 3354003 53.76  
## 2623 14.06 637045 7.03 637044 5233000 57.75  
## 2624 16.75 673832 8.54 647793 4687000 59.40  
## 2625 14.06 1171985 7.03 1171986 5967531 35.80  
## 2626 11.93 722857 6.64 575891 3673011 33.74  
## Boro  
## 2621 Staten Island  
## 2622 Staten Island  
## 2623 Staten Island  
## 2624 Staten Island  
## 2625 Staten Island  
## 2626 Staten Island

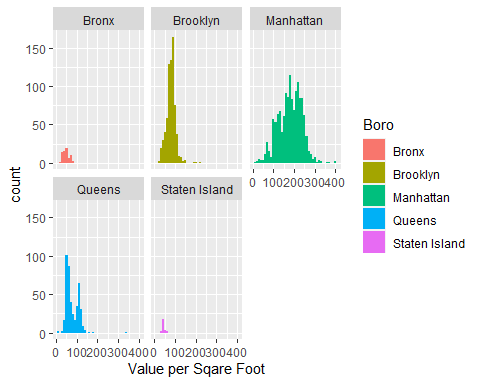
#Now we are considering Valuepersqft as our response variable.  
  
# Visulaize the Data  
require(ggplot2)  
require(lattice)  
  
# Drawing Histofgram for ValuePerSqFt  
ggplot(housing, aes(x=ValuePerSqFt))+geom\_histogram(binwidth=10)+labs(x="Value per Sqare Foot")



ggplot(housing, aes(x=ValuePerSqFt, fill=Boro))+geom\_histogram(binwidth=10)+labs(x="Value per Sqare Foot")

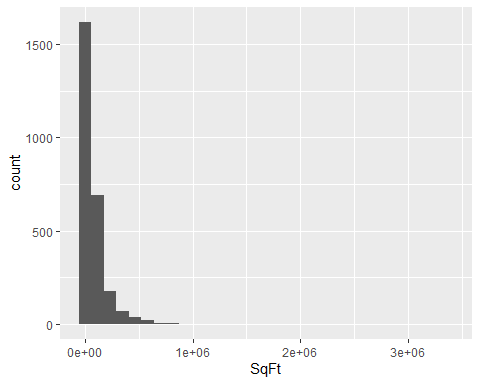


ggplot(housing, aes(x=ValuePerSqFt, fill=Boro))+geom\_histogram(binwidth=10)+labs(x="Value per Sqare Foot")+facet\_wrap("Boro")



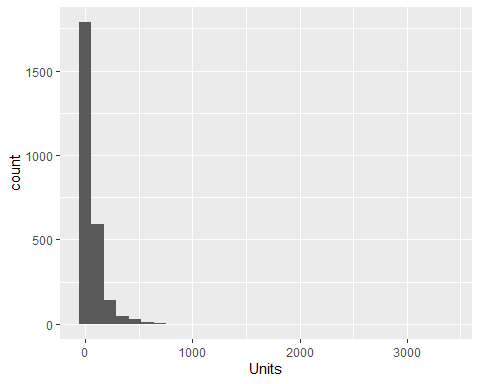
# Creating Histogram  
# For Square Footage and Units  
  
ggplot(housing, aes(x=SqFt))+geom\_histogram()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



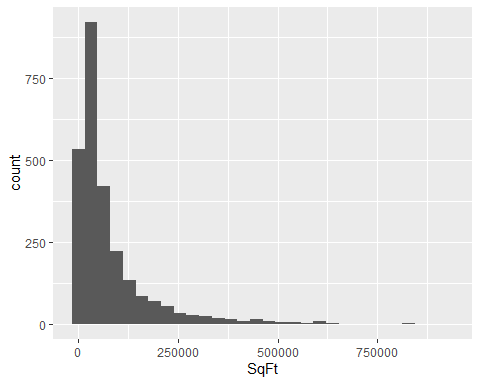
ggplot(housing, aes(x=Units))+geom\_histogram()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



ggplot(housing[housing$Units<1000,], aes(x=SqFt))+geom\_histogram()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

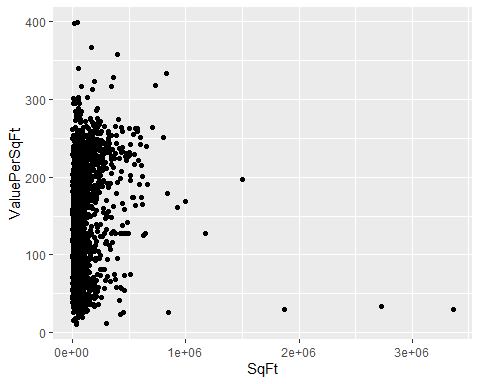


ggplot(housing[housing$Units<1000,], aes(x=Units))+geom\_histogram()

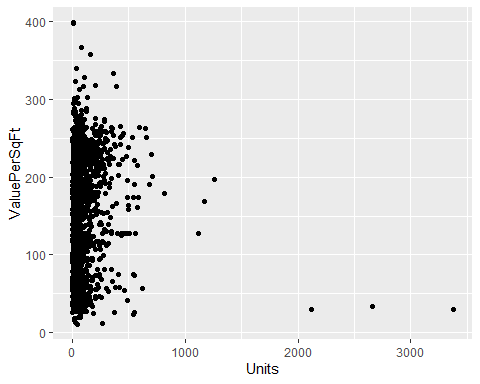
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# We use point after the geom function to get the scatter graph.  
ggplot(housing, aes(x=SqFt, y=ValuePerSqFt))+geom\_point()



ggplot(housing, aes(x=Units, y=ValuePerSqFt))+geom\_point()

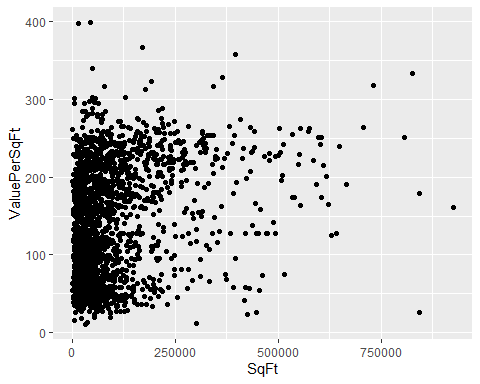


# How many Housing$Units greater than 1000  
sum(housing$Units>=1000)

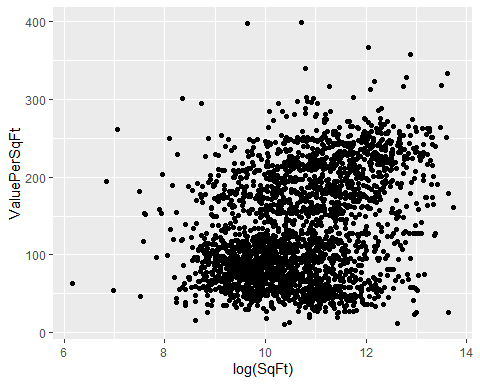
## [1] 6

# Remove Housing$Units greater than 1000  
housing=housing[housing$Units<1000,]

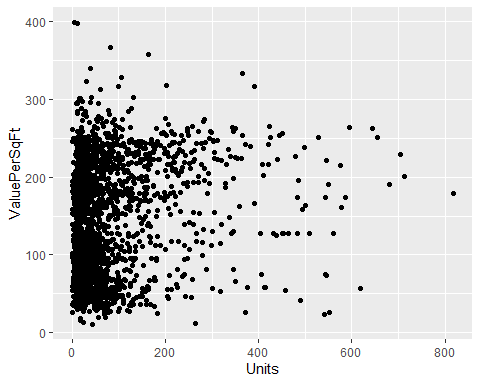
# We would plot the value persqft value against the sqft value using the ggplot function.  
ggplot(housing, aes(x=SqFt, y=ValuePerSqFt))+geom\_point()



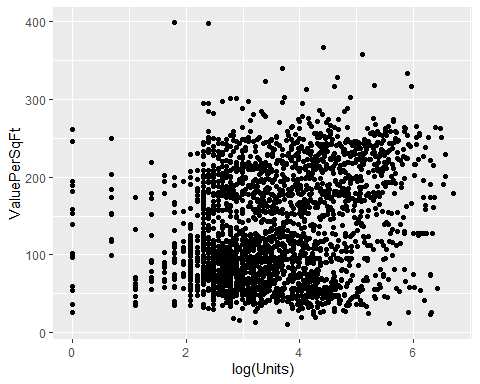
ggplot(housing, aes(x=log(SqFt), y=ValuePerSqFt))+geom\_point()



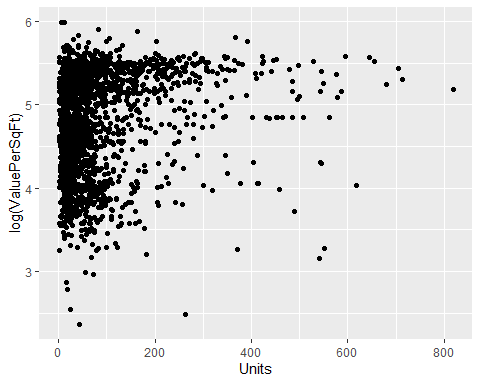
# Plot ValePerSqFt against Units  
ggplot(housing, aes(x=Units, y=ValuePerSqFt))+geom\_point()



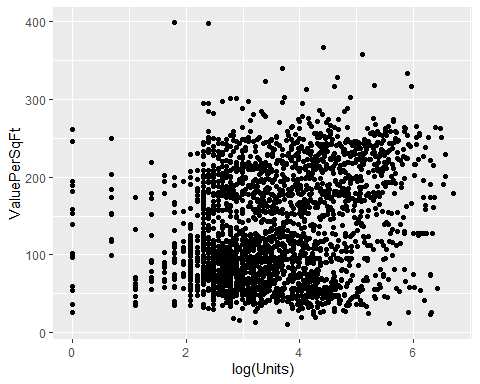
ggplot(housing, aes(x=log(Units), y=ValuePerSqFt))+geom\_point()



ggplot(housing, aes(x=Units, y=log(ValuePerSqFt)))+geom\_point()



ggplot(housing, aes(x=log(Units), y=ValuePerSqFt))+geom\_point()



# Fit the Model  
house1=lm(ValuePerSqFt~Units+SqFt+Boro, data=housing)  
summary(house1)

##   
## Call:  
## lm(formula = ValuePerSqFt ~ Units + SqFt + Boro, data = housing)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -168.458 -22.680 1.493 26.290 261.761   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 4.430e+01 5.342e+00 8.293 < 2e-16 \*\*\*  
## Units -1.532e-01 2.421e-02 -6.330 2.88e-10 \*\*\*  
## SqFt 2.070e-04 2.129e-05 9.723 < 2e-16 \*\*\*  
## BoroBrooklyn 3.258e+01 5.561e+00 5.858 5.28e-09 \*\*\*  
## BoroManhattan 1.274e+02 5.459e+00 23.343 < 2e-16 \*\*\*  
## BoroQueens 3.011e+01 5.711e+00 5.272 1.46e-07 \*\*\*  
## BoroStaten Island -7.114e+00 1.001e+01 -0.711 0.477   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 43.2 on 2613 degrees of freedom  
## Multiple R-squared: 0.6034, Adjusted R-squared: 0.6025   
## F-statistic: 662.6 on 6 and 2613 DF, p-value: < 2.2e-16

# To get only Coefficients  
house1$coefficients

## (Intercept) Units SqFt BoroBrooklyn   
## 4.430325e+01 -1.532405e-01 2.069727e-04 3.257554e+01   
## BoroManhattan BoroQueens BoroStaten Island   
## 1.274259e+02 3.011000e+01 -7.113688e+00

coef(house1)

## (Intercept) Units SqFt BoroBrooklyn   
## 4.430325e+01 -1.532405e-01 2.069727e-04 3.257554e+01   
## BoroManhattan BoroQueens BoroStaten Island   
## 1.274259e+02 3.011000e+01 -7.113688e+00

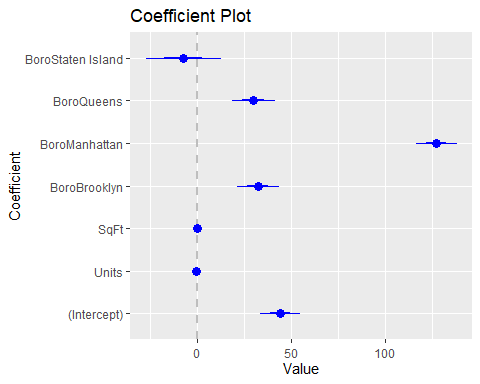
coefficients(house1)

## (Intercept) Units SqFt BoroBrooklyn   
## 4.430325e+01 -1.532405e-01 2.069727e-04 3.257554e+01   
## BoroManhattan BoroQueens BoroStaten Island   
## 1.274259e+02 3.011000e+01 -7.113688e+00

# Draw coefficients Plot  
  
require(coefplot)

## Loading required package: coefplot

coefplot(house1)



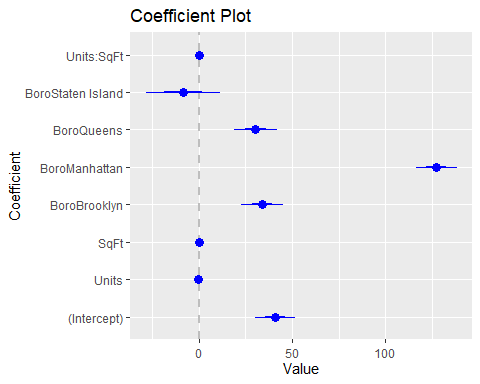
house2=lm(ValuePerSqFt~Units\*SqFt+Boro, data = housing)  
house3=lm(ValuePerSqFt~Units:SqFt+Boro, data = housing)  
house2$coefficients

## (Intercept) Units SqFt BoroBrooklyn   
## 4.093685e+01 -1.024579e-01 2.362293e-04 3.394544e+01   
## BoroManhattan BoroQueens BoroStaten Island Units:SqFt   
## 1.272102e+02 3.040115e+01 -8.419682e+00 -1.809587e-07

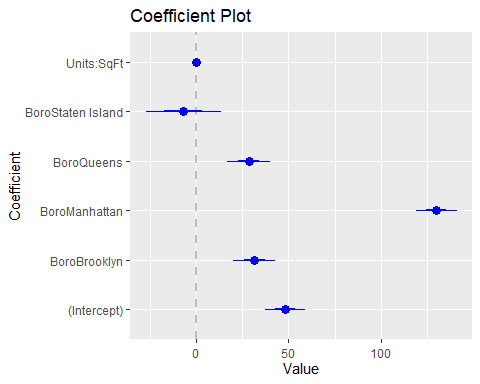
house3$coefficients

## (Intercept) BoroBrooklyn BoroManhattan BoroQueens   
## 4.804972e+01 3.141208e+01 1.302084e+02 2.841669e+01   
## BoroStaten Island Units:SqFt   
## -7.199902e+00 1.088059e-07

# Drawaing COeficcient Plot Graph in R  
# We use the coefplot function to plot the coefficients in the Regression on a Graph.  
# Coefficient plot graph for House 2  
coefplot(house2)



# Coefficient plot graph for House 3  
coefplot(house3)



# We are doing a regression of the value of value per sqft units and income.   
house4=lm(ValuePerSqFt~ SqFt\*Units\*Income, data = housing)  
house4$coefficients

## (Intercept) SqFt Units Income   
## 1.116433e+02 -1.694688e-03 7.142611e-03 7.250830e-05   
## SqFt:Units SqFt:Income Units:Income SqFt:Units:Income   
## 3.158094e-06 -5.129522e-11 -1.279236e-07 9.107312e-14

# Interaction between TWO Categorical variables  
house5=lm(ValuePerSqFt~Units:Class\*Boro, data = housing)  
house5$coefficients

## (Intercept)   
## 5.494287e+01   
## BoroBrooklyn   
## 2.653390e+01   
## BoroManhattan   
## 1.191513e+02   
## BoroQueens   
## 2.671272e+01   
## BoroStaten Island   
## -1.297158e+01   
## Units:ClassR2-CONDOMINIUM   
## -3.041300e-01   
## Units:ClassR4-CONDOMINIUM   
## -1.062423e-01   
## Units:ClassR9-CONDOMINIUM   
## -4.872488e-02   
## Units:ClassRR-CONDOMINIUM   
## 5.268445e-03   
## Units:ClassR2-CONDOMINIUM:BoroBrooklyn   
## -2.656421e-05   
## Units:ClassR4-CONDOMINIUM:BoroBrooklyn   
## 1.255519e-01   
## Units:ClassR9-CONDOMINIUM:BoroBrooklyn   
## -1.039370e-01   
## Units:ClassRR-CONDOMINIUM:BoroBrooklyn   
## -6.328234e-01   
## Units:ClassR2-CONDOMINIUM:BoroManhattan   
## -1.156371e+00   
## Units:ClassR4-CONDOMINIUM:BoroManhattan   
## 2.433943e-01   
## Units:ClassR9-CONDOMINIUM:BoroManhattan   
## 1.276151e-02   
## Units:ClassRR-CONDOMINIUM:BoroManhattan   
## 1.430965e-02   
## Units:ClassR2-CONDOMINIUM:BoroQueens   
## 9.655360e-02   
## Units:ClassR4-CONDOMINIUM:BoroQueens   
## 6.848450e-02   
## Units:ClassR9-CONDOMINIUM:BoroQueens   
## -5.456138e-02   
## Units:ClassRR-CONDOMINIUM:BoroQueens   
## 6.057630e-01   
## Units:ClassR2-CONDOMINIUM:BoroStaten Island   
## NA   
## Units:ClassR4-CONDOMINIUM:BoroStaten Island   
## 1.048513e-01   
## Units:ClassR9-CONDOMINIUM:BoroStaten Island   
## NA   
## Units:ClassRR-CONDOMINIUM:BoroStaten Island   
## NA

# We use I() function for getting the ratios in R  
house6=lm(ValuePerSqFt~ I(SqFt/Units)+Boro, data = housing)  
house6$coefficients

## (Intercept) I(SqFt/Units) BoroBrooklyn BoroManhattan   
## 43.754838763 0.004017039 30.774343209 130.769502685   
## BoroQueens BoroStaten Island   
## 29.767922792 -6.134446417

house7=lm(ValuePerSqFt~ (Units+SqFt)^2,housing)  
house7$coefficients

## (Intercept) Units SqFt Units:SqFt   
## 1.070301e+02 -1.125194e-01 4.964623e-04 -5.159669e-07

house8=lm(ValuePerSqFt~Units\*SqFt, housing)  
house8$coefficients

## (Intercept) Units SqFt Units:SqFt   
## 1.070301e+02 -1.125194e-01 4.964623e-04 -5.159669e-07

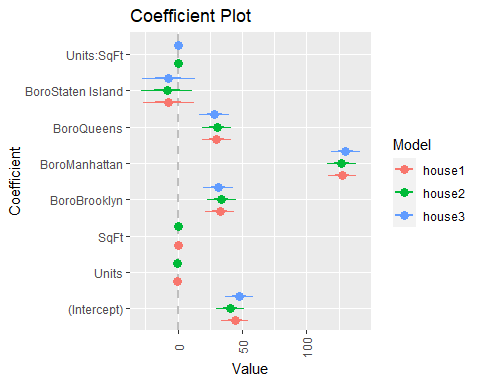
identical(house7$coefficients, house8$coefficients)

## [1] TRUE

house9=lm(ValuePerSqFt~ I(Units+SqFt)^2,housing)  
house9$coefficients

## (Intercept) I(Units + SqFt)   
## 1.147034e+02 2.107231e-04

# We can also plot multiple coefficient of data in R   
# For this we use the multiplot() function.  
multiplot(house1, house2, house3)



# We use the summary function to get the detailed summary about the regression analysis.  
summary(house1)

##   
## Call:  
## lm(formula = ValuePerSqFt ~ Units + SqFt + Boro, data = housing)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -168.458 -22.680 1.493 26.290 261.761   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 4.430e+01 5.342e+00 8.293 < 2e-16 \*\*\*  
## Units -1.532e-01 2.421e-02 -6.330 2.88e-10 \*\*\*  
## SqFt 2.070e-04 2.129e-05 9.723 < 2e-16 \*\*\*  
## BoroBrooklyn 3.258e+01 5.561e+00 5.858 5.28e-09 \*\*\*  
## BoroManhattan 1.274e+02 5.459e+00 23.343 < 2e-16 \*\*\*  
## BoroQueens 3.011e+01 5.711e+00 5.272 1.46e-07 \*\*\*  
## BoroStaten Island -7.114e+00 1.001e+01 -0.711 0.477   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 43.2 on 2613 degrees of freedom  
## Multiple R-squared: 0.6034, Adjusted R-squared: 0.6025   
## F-statistic: 662.6 on 6 and 2613 DF, p-value: < 2.2e-16

summary(house2)

##   
## Call:  
## lm(formula = ValuePerSqFt ~ Units \* SqFt + Boro, data = housing)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -161.888 -22.867 1.802 26.431 261.733   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 4.094e+01 5.393e+00 7.590 4.41e-14 \*\*\*  
## Units -1.025e-01 2.728e-02 -3.755 0.000177 \*\*\*  
## SqFt 2.362e-04 2.245e-05 10.521 < 2e-16 \*\*\*  
## BoroBrooklyn 3.395e+01 5.556e+00 6.110 1.15e-09 \*\*\*  
## BoroManhattan 1.272e+02 5.444e+00 23.369 < 2e-16 \*\*\*  
## BoroQueens 3.040e+01 5.696e+00 5.338 1.02e-07 \*\*\*  
## BoroStaten Island -8.420e+00 9.985e+00 -0.843 0.399160   
## Units:SqFt -1.810e-07 4.530e-08 -3.995 6.65e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 43.08 on 2612 degrees of freedom  
## Multiple R-squared: 0.6058, Adjusted R-squared: 0.6047   
## F-statistic: 573.4 on 7 and 2612 DF, p-value: < 2.2e-16

summary(house3)

##   
## Call:  
## lm(formula = ValuePerSqFt ~ Units:SqFt + Boro, data = housing)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -165.666 -22.840 0.161 27.351 263.442   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 4.805e+01 5.425e+00 8.857 < 2e-16 \*\*\*  
## BoroBrooklyn 3.141e+01 5.668e+00 5.542 3.30e-08 \*\*\*  
## BoroManhattan 1.302e+02 5.561e+00 23.414 < 2e-16 \*\*\*  
## BoroQueens 2.842e+01 5.822e+00 4.881 1.12e-06 \*\*\*  
## BoroStaten Island -7.200e+00 1.020e+01 -0.706 0.48   
## Units:SqFt 1.088e-07 1.926e-08 5.649 1.79e-08 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 44.07 on 2614 degrees of freedom  
## Multiple R-squared: 0.5873, Adjusted R-squared: 0.5865   
## F-statistic: 743.8 on 5 and 2614 DF, p-value: < 2.2e-16

summary(house4)

##   
## Call:  
## lm(formula = ValuePerSqFt ~ SqFt \* Units \* Income, data = housing)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -226.020 -26.409 -7.011 20.752 274.520   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.116e+02 1.382e+00 80.791 <2e-16 \*\*\*  
## SqFt -1.695e-03 5.324e-05 -31.834 <2e-16 \*\*\*  
## Units 7.143e-03 3.886e-02 0.184 0.854   
## Income 7.251e-05 1.320e-06 54.919 <2e-16 \*\*\*  
## SqFt:Units 3.158e-06 1.328e-07 23.775 <2e-16 \*\*\*  
## SqFt:Income -5.130e-11 1.835e-12 -27.959 <2e-16 \*\*\*  
## Units:Income -1.279e-07 4.785e-09 -26.733 <2e-16 \*\*\*  
## SqFt:Units:Income 9.107e-14 4.519e-15 20.152 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 40.47 on 2612 degrees of freedom  
## Multiple R-squared: 0.6522, Adjusted R-squared: 0.6513   
## F-statistic: 699.8 on 7 and 2612 DF, p-value: < 2.2e-16

summary(house5)

##   
## Call:  
## lm(formula = ValuePerSqFt ~ Units:Class \* Boro, data = housing)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -173.145 -22.686 0.031 26.007 259.975   
##   
## Coefficients: (3 not defined because of singularities)  
## Estimate Std. Error t value  
## (Intercept) 5.494e+01 9.267e+00 5.929  
## BoroBrooklyn 2.653e+01 9.500e+00 2.793  
## BoroManhattan 1.192e+02 9.389e+00 12.691  
## BoroQueens 2.671e+01 9.636e+00 2.772  
## BoroStaten Island -1.297e+01 1.706e+01 -0.760  
## Units:ClassR2-CONDOMINIUM -3.041e-01 4.244e-01 -0.717  
## Units:ClassR4-CONDOMINIUM -1.062e-01 1.430e-01 -0.743  
## Units:ClassR9-CONDOMINIUM -4.872e-02 1.236e-01 -0.394  
## Units:ClassRR-CONDOMINIUM 5.268e-03 3.715e-01 0.014  
## Units:ClassR2-CONDOMINIUM:BoroBrooklyn -2.656e-05 4.524e-01 0.000  
## Units:ClassR4-CONDOMINIUM:BoroBrooklyn 1.256e-01 1.466e-01 0.857  
## Units:ClassR9-CONDOMINIUM:BoroBrooklyn -1.039e-01 1.770e-01 -0.587  
## Units:ClassRR-CONDOMINIUM:BoroBrooklyn -6.328e-01 7.391e-01 -0.856  
## Units:ClassR2-CONDOMINIUM:BoroManhattan -1.156e+00 4.757e-01 -2.431  
## Units:ClassR4-CONDOMINIUM:BoroManhattan 2.434e-01 1.435e-01 1.696  
## Units:ClassR9-CONDOMINIUM:BoroManhattan 1.276e-02 1.250e-01 0.102  
## Units:ClassRR-CONDOMINIUM:BoroManhattan 1.431e-02 3.725e-01 0.038  
## Units:ClassR2-CONDOMINIUM:BoroQueens 9.655e-02 4.309e-01 0.224  
## Units:ClassR4-CONDOMINIUM:BoroQueens 6.848e-02 1.465e-01 0.467  
## Units:ClassR9-CONDOMINIUM:BoroQueens -5.456e-02 1.300e-01 -0.420  
## Units:ClassRR-CONDOMINIUM:BoroQueens 6.058e-01 9.633e-01 0.629  
## Units:ClassR2-CONDOMINIUM:BoroStaten Island NA NA NA  
## Units:ClassR4-CONDOMINIUM:BoroStaten Island 1.049e-01 2.058e-01 0.510  
## Units:ClassR9-CONDOMINIUM:BoroStaten Island NA NA NA  
## Units:ClassRR-CONDOMINIUM:BoroStaten Island NA NA NA  
## Pr(>|t|)   
## (Intercept) 3.45e-09 \*\*\*  
## BoroBrooklyn 0.00526 \*\*   
## BoroManhattan < 2e-16 \*\*\*  
## BoroQueens 0.00561 \*\*   
## BoroStaten Island 0.44711   
## Units:ClassR2-CONDOMINIUM 0.47371   
## Units:ClassR4-CONDOMINIUM 0.45750   
## Units:ClassR9-CONDOMINIUM 0.69346   
## Units:ClassRR-CONDOMINIUM 0.98869   
## Units:ClassR2-CONDOMINIUM:BoroBrooklyn 0.99995   
## Units:ClassR4-CONDOMINIUM:BoroBrooklyn 0.39180   
## Units:ClassR9-CONDOMINIUM:BoroBrooklyn 0.55715   
## Units:ClassRR-CONDOMINIUM:BoroBrooklyn 0.39197   
## Units:ClassR2-CONDOMINIUM:BoroManhattan 0.01513 \*   
## Units:ClassR4-CONDOMINIUM:BoroManhattan 0.08999 .   
## Units:ClassR9-CONDOMINIUM:BoroManhattan 0.91868   
## Units:ClassRR-CONDOMINIUM:BoroManhattan 0.96936   
## Units:ClassR2-CONDOMINIUM:BoroQueens 0.82271   
## Units:ClassR4-CONDOMINIUM:BoroQueens 0.64028   
## Units:ClassR9-CONDOMINIUM:BoroQueens 0.67473   
## Units:ClassRR-CONDOMINIUM:BoroQueens 0.52952   
## Units:ClassR2-CONDOMINIUM:BoroStaten Island NA   
## Units:ClassR4-CONDOMINIUM:BoroStaten Island 0.61041   
## Units:ClassR9-CONDOMINIUM:BoroStaten Island NA   
## Units:ClassRR-CONDOMINIUM:BoroStaten Island NA   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 42.61 on 2598 degrees of freedom  
## Multiple R-squared: 0.6165, Adjusted R-squared: 0.6134   
## F-statistic: 198.8 on 21 and 2598 DF, p-value: < 2.2e-16

summary(house6)

##   
## Call:  
## lm(formula = ValuePerSqFt ~ I(SqFt/Units) + Boro, data = housing)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -167.585 -23.001 0.282 27.558 261.709   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 4.375e+01 5.560e+00 7.869 5.18e-15 \*\*\*  
## I(SqFt/Units) 4.017e-03 9.438e-04 4.256 2.15e-05 \*\*\*  
## BoroBrooklyn 3.077e+01 5.684e+00 5.414 6.72e-08 \*\*\*  
## BoroManhattan 1.308e+02 5.574e+00 23.461 < 2e-16 \*\*\*  
## BoroQueens 2.977e+01 5.842e+00 5.096 3.72e-07 \*\*\*  
## BoroStaten Island -6.134e+00 1.023e+01 -0.600 0.549   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 44.18 on 2614 degrees of freedom  
## Multiple R-squared: 0.5851, Adjusted R-squared: 0.5843   
## F-statistic: 737.2 on 5 and 2614 DF, p-value: < 2.2e-16

summary(house7)

##   
## Call:  
## lm(formula = ValuePerSqFt ~ (Units + SqFt)^2, data = housing)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -296.38 -46.65 -12.21 48.52 284.15   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.070e+02 1.874e+00 57.121 < 2e-16 \*\*\*  
## Units -1.125e-01 3.974e-02 -2.831 0.00467 \*\*   
## SqFt 4.965e-04 3.225e-05 15.393 < 2e-16 \*\*\*  
## Units:SqFt -5.160e-07 6.541e-08 -7.889 4.45e-15 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 63.42 on 2616 degrees of freedom  
## Multiple R-squared: 0.1443, Adjusted R-squared: 0.1433   
## F-statistic: 147.1 on 3 and 2616 DF, p-value: < 2.2e-16

summary(house9)

##   
## Call:  
## lm(formula = ValuePerSqFt ~ I(Units + SqFt)^2, data = housing)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -266.37 -48.80 -14.04 52.70 279.58   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.147e+02 1.575e+00 72.83 <2e-16 \*\*\*  
## I(Units + SqFt) 2.107e-04 1.193e-05 17.67 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 64.79 on 2618 degrees of freedom  
## Multiple R-squared: 0.1065, Adjusted R-squared: 0.1062   
## F-statistic: 312.1 on 1 and 2618 DF, p-value: < 2.2e-16

housingNew=read.table("http://www.jaredlander.com/data/housingNew.csv", sep =",", header = TRUE, stringsAsFactors=FALSE)  
# write.table(housingNew, "Testhousing.csv", col.names = NA,row.names = TRUE, quote = FALSE, sep =",")  
# To predict with 95 % confidence bounds  
housePredict=predict(house1, newdata=housingNew, se.fit=TRUE, interval="prediction", level=0.95)  
head(housePredict$fit)

## fit lwr upr  
## 1 74.00645 -10.813887 158.8268  
## 2 82.04988 -2.728506 166.8283  
## 3 166.65975 81.808078 251.5114  
## 4 169.00970 84.222648 253.7968  
## 5 80.00129 -4.777303 164.7799  
## 6 47.87795 -37.480170 133.2361

View(housePredict$fit)

house4Predict=predict(house4, newdata=housingNew, se.fit=TRUE, interval="prediction", level=0.95)  
head(house4Predict$fit)

## fit lwr upr  
## 1 110.87020 31.488812 190.2516  
## 2 70.55054 -8.841812 149.9429  
## 3 124.94771 45.360975 204.5344  
## 4 150.81847 71.369611 230.2673  
## 5 101.93912 22.563391 181.3149  
## 6 100.59737 21.223953 179.9708

View(house4Predict$fit)

**Key Learnings-**

* R greatest advantage is that it is Open-source Software.
* It also gives flexibility in command line interface.
* By doing the assignment I get to learn more about the basic and complex features of R.
* It helps me in providing a platform where I can do various types of Hypothesis testing with ease.
* It also helped me in providing an another medium of creating graphs.
* I also get to learn about various features and functions.
* Also I got to learn that R gives us an option to directly import data from the internet.
* R also gives us the flexibility of doing various tasks like creating tables, doing arithmetic operations and making graphs in one platform only.