SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA ENGINEERING AND TECHNOLOGY

# CSE 240 Data Science with R

## STUDENT WORK BOOK

| | | |
|---|---|---|
| **Name** | **:** | M.Sathishkumar |
| **Unique ID** | **:** | E0119052 |
| **Year** | **:** | II |
| **Quarter** | **:** | Q6 |
| **Department** | **:** | B.Tech CSE (AI &ML) |
| **Faculty Name** | **:** | Prof.B.Nirmala |
| **Academic Year** | **:** | 2020-2021 |

**Date:02/11/2020**

**Q.NO 1: Question**

Consider 2 vectors c(9,10,11,12) and c(13,14,15,16).

Create a 4 by 2 matrix from these two vectors

**Program:**

```
#Matrix creation
matrix<-matrix(c(c(9,10,11,12),c(13,14,15,16)),nrow = 4,ncol = 2)
#combining two vertices and converting into 4 by 2 matrix
print(matrix)
```

**Output:**

```
     [,1] [,2]
[1,]    9   13
[2,]   10   14
[3,]   11   15
[4,]   12   16
```

**Explanation:**

- ✶ Matrix is a two-dimensional data structure in R programming.
- ✶ Matrix can be created using the matrix () function
- ✶ Dimension of the matrix can be defined by passing appropriate value guments nrow and ncol

**Date: 02/11/2020**

**Q.NO 2: Question**

**Program:**

Write an R program to take input from the user (userID and Group/Branch) and display the values

```
#UserInput
Id = readline("Enter your UserID:")
#Prompts user for input(id)
Branch = readline(prompt="Enter you Branch/Group:")
#Prompts user for input(branch)
cat("Your UserID is",Id,"and you belong to",Branch,"group")
#Displaying Values
```

**Output:**

```
Enter your UserID:E0119052
Enter you Branch/Group:AI & ML
Your UserID is E0119052 and you belong to AI & ML group
```

**Explanation:**

- readline reads a line from the terminal (in interactive use).
- readline() lets the user enter a one-line string at the terminal.
- The prompt argument is printed in front of the user input. It usually ends on ": ".

**Date: 02/11/2020**

**Q.NO 3: Question**

Create a data frame Write a R program to create a data frame from four given vectors.

a name b. Subject C. Score d. Rank

**Program:**

```
#Dataframe creation
Name <- c("Jhon", "Lee", "Suzan", "Abhinav", "Brain", "Emma", "David")
#Variables
Subject <- c("Data Science", "Machine Learning", "Deep Learning",
             "Data Structures", "Database Managemnt System",
             "Operating Systems", "Python Programming")

Score <- c(56, 76, 86, 96, 73, 87, 47)

Rank <- c(5,8,6,7,9,2,1)

df = data.frame(Name,Subject,Score,Rank)
print(df)
#Dataframe Created Successfully
```

**Output:**

```
     Name                   Subject Score Rank
1    Jhon              Data Science    56    5
2     Lee          Machine Learning    76    8
3   Suzan             Deep Learning    86    6
4 Abhinav           Data Structures    96    7
5   Brain Database Managemnt System    73    9
6    Emma          Operating Systems    87    2
7   David         Python Programming    47    1
```

**Explanation:**

- A **data frame** is used for storing data tables. It is a list of vectors of equal length.
- The top line of the table, called the **header**, contains the column names.
- Each horizontal line afterward denotes a **data row**, which begins with header, and then followed by the actual data.
- Each data member of a row is called a **cell**.

**Date: 02/11/2020**

**Q.NO 4: Question**

Write a R program to get the statistical summary and nature of the data of a given data frame. ( use 3rd Question dataframe

**Program:**

```
#Dataframe creation
Name <- c("Jhon", "Lee", "Suzan", "Abhinav", "Brain", "Emma", "David")
#Variables
Subject <- c("Data Science", "Machine Learning", "Deep Learning",
             "Data Structures", "Database Managemnt System",
             "Operating Systems", "Python Programming")
Score <- c(56, 76, 86, 96, 73, 87, 47)
Rank <- c(5,8,6,7,9,2,1)
df = data.frame(Name,Subject,Score,Rank)
print(df)
#Dataframe Created Successfully
print(summary(df))
#Dataframe Stastical summary
```

**Output:**

```
     Name                         Subject      Score            Rank
 Abhinav:1   Data Science              :1   Min.   :47.00   Min.   :1.000
 Brain  :1   Data Structures           :1   1st Qu.:64.50   1st Qu.:3.500
 David  :1   Database Managemnt System:1   Median :76.00   Median :6.000
 Emma   :1   Deep Learning             :1   Mean   :74.43   Mean   :5.429
 Jhon   :1   Machine Learning          :1   3rd Qu.:86.50   3rd Qu.:7.500
 Lee    :1   Operating Systems         :1   Max.   :96.00   Max.   :9.000
 Suzan  :1   Python Programming        :1
```

**Explanation:**

Summary is a generic function used to produce result summaries of the results of various model fitting functions.

The function invokes particular methods which depend on the class of the first argument

The form of the value returned by summary depends on the class of its argument

**Date: 02/11/2020**

**Q.NO 5: Question**

Write a R program to extract specific column from a data frame using column name

**Program:**

```
#Dataframe creation
Name <- c("Jhon", "Lee", "Suzan", "Abhinav", "Brain", "Emma", "David")
#Variables
Subject <- c("Data Science", "Machine Learning", "Deep Learning",
             "Data Structures", "Database Managemnt System",
             "Operating Systems", "Python Programming")
Score <- c(56, 76, 86, 96, 73, 87, 47)
Rank <- c(5,8,6,7,9,2,1)
df = data.frame(Name,Subject,Score,Rank)
print(df)
#Dataframe Created Successfully
print(df["Subject"])
#Sliced column with column name
```

**Output:**

```
                      Subject
1              Data Science
2          Machine Learning
3             Deep Learning
4           Data Structures
5 Database Managemnt System
6          Operating Systems
7         Python Programming
```

**Explanation:**

- ✶ We retrieve a data frame column slice with the single square bracket "[]" operator.
- ✶ Here We can retrieve the same column slice by its name.
- ✶ And we can pack the row names in an index vector in order to retrieve multiple rows

**Date: 02/11/2020**

**Q.NO 6: Question**

.Write a R program to extract first two rows from a given data frame

**Program:**

```
#Dataframe creation
Name <- c("Jhon", "Lee", "Suzan", "Abhinav", "Brain", "Emma", "David")
#Variables
Subject <- c("Data Science", "Machine Learning", "Deep Learning",
             "Data Structures", "Database Managemnt System",
             "Operating Systems", "Python Programming")
Score <- c(56, 76, 86, 96, 73, 87, 47)
Rank <- c(5,8,6,7,9,2,1)
df = data.frame(Name,Subject,Score,Rank)
#Dataframe Created Successfully
print(df[1:2,])
#splicing Dataframe with respect to index
```

**Output:**

```
  Name           Subject Score Rank
1 Jhon      Data Science    56    5
2  Lee Machine Learning    76    8
```

**Explanation:**

- ✶ We retrieve rows from a data frame with the single square bracket operator, just like what we did with columns.
- ✶ However, in additional to an index vector of row positions, we append an extra comma character.
- ✶ This is important, as the extra comma signals a wildcard match for the second coordinate for column positions.

**Date: 03/11/2020**

**Q.NO 7: Question**

Create an R script that calculates the square root of a given integer vector x of length one, if the value contained in x is negative it should return NA.

**Program:**

```
#Prompting user for input
x = c(as.integer(readline("Enter a number:")))
#Defining a function for finding square root
squareroot <- function(x){
  if (x < 0){
    print(NA)
  }else{
    print(paste("Square root of the given number :",sqrt(x)))
  }
}
squareroot(x)
#Invoking the function
```

**Output:**

```
Enter a number:57
[1] "Square root of the given number : 7.54983443527075"

Enter a number:-78
[1] NA
```

**Explanation:**

- Missing values are represented by the symbol **NA** (not available)
- If the numeric_Expression is a positive value, the sqrt function returns the square root of a given value.
- If the numeric_Expression is a negative value, the sqrt function return ***NaN***.
- Numeric_Expression is not a number (**NaN**), or Negative Infinity, then sqrt in R returns ***NaN***.

**Date: 03/11/2020**

**Q.NO 8: Question**

Demonstrate and examine the output of letter and LETTER

**Program:**

```
a = "letter"
b = "LETTER"
print(a)          #printing letter
print(b)          #printing LETTER
```

**Output:**

```
[1] "letter"
[1] "LETTER"
```

**Explanation:**

- ✶ This is useful because you can always view your data: just print it.
- ✶ You needn't write special printing logic, even for complicated data structures.
- ✶ The print function has a significant limitation, it prints only one object at a time.
- ✶ Trying to print multiple items gives this mind-numbing error message

**Date: 03/11/2020**

**Q.NO 9: Question**

Create an R script that, given a numeric vector x with length 3, will print the elements by order from high to low.

**Program:**

```
desc <- function(x){
    x<-c(5,7,8)
    }
print(sort(desc(x),decreasing = TRUE))
```

**Output:**

```
[1] 8 7 5
```

**Explanation:**

- Sort (or *order*) a vector or factor (partially) into ascending or descending order. For ordering along more than one variable, e.g., for sorting data frame
- To sort a data frame in R, use the **sort( )** function. By default, sorting is ASCENDING.
- Prepend the sorting variable by a minus sign to indicate DESCENDING order

**Date: 03/11/2020**

**Q.NO 10: Question**

Create an R script that returns the amount of values that are larger than the mean of a vector. You are allowed to use mean(). ( Use function)

**Program:**

```
#function
nums<- function(vec)
    {
  #returning values greater then mean value
  return(vec[vec>mean(vec)])
    }
#Assigning values
nums(c(78,12,459,175,451,178,587,24,985))
```

**Output:**

```
459  451  587  985
```

**Explanation:**

- ✶ It is calculated by taking the sum of the values and dividing with the number of values in a data series.
- ✶ The function **mean()** is used to calculate and decision operator “>” is used to compare the giving condition and take decision

**Date: 03/11/2020**

**Q.NO 11: Question**

Write a double for loop which prints 30 numbers (1:10, 2:11, 3:12). Those are three clusters of ten numbers each. The first loop determines the number of clusters (3) via its length; the second loop the numbers to be printed (1 to 10 at the beginning). Each cluster starts one number higher than the previous one

**Program:**

```
#Nested looping
for(i in seq(1,3)){
  #for every single run in outside loop inside loop gets executed 10 times
  for(j in seq(i,i+10L)){
    print(j)
  }
}
```

**Output:**

```
1  2  3  4  5  6  7  8  9  10  11  2  3  4  5  6  7  8  9  10  11  12  3  4  5  6  7  8  9  10  11  12  13
```

**Explanation:**

- In Nested for Loop, it makes use of the control structures to manage the execution of the expression, one such control structure is Nested for Loop a similar to basic 'for' loop executes.
- It can be defined as placing one 'for' loop inside the first 'for' loop is called as nesting or loop of loops in some terms, which takes the responsibility of two loops such that the outer loop controls the number of repetition of the whole inner detailed information until it is false,
- in other words, the inner loop executes n-times of every execution of the outer for loop and also, it's a great tool to work with R Programming Language.

**Date: 03/11/2020**

**Q.NO 12: Question**

a. You have the data.frame 'mydf' with four columns like below

a = c(3,7,NA, 9) b = c(2,NA,9,3) f = c(5,2,5,6) d = c(NA,3,4,NA)

You want to add another column '5': the 5th column contains the value of col 2 if col 1 is NA; the 5th column contains the value of col 4 if col 2 is NA; the 5th column contains the value of col 3 in all other cases.

**Program:**

```
#Creating a dataframe
mydf = data.frame(a=c(3,7,NA, 9),b=c(2,NA,9,3),f=c(5,2,5,6),d=c(NA,3,4,NA))
#Applying conditions as required to create 5th column
mydf[,5] <- ifelse(is.na(mydf[,1]) & !is.na(mydf[,2]),mydf[,2],
            ifelse(is.na(mydf[,2]) & !is.na(mydf[,4]),mydf[,4], mydf[,3]))

print(mydf)
```

**Output:**

| | a | b | f | d | V5 |
|---|---|---|---|---|---|
| 1 | 3 | 2 | 5 | NA | 5 |
| 2 | 7 | NA | 2 | 3 | 3 |
| 3 | NA | 9 | 5 | 4 | 9 |
| 4 | 9 | 3 | 6 | NA | 6 |

**Explanation:**

- ✶ An if statement can be followed by an optional else statement which executes when the Boolean expression is false.
- ✶ An if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement

**Date: 03/11/2020**

**Q.NO 13: Question**

Write a while loop starting with x = 0. The loop prints all numbers up to 35 but it skips number 7. Condition: If x== 7 next

**Program:**

```
#While looping
x<-0
while (x<=35) {
  if(x==7){
    x<-x+1
  #Applying next function to bypass if x equals 7
  next
  }
  cat(x,"")
  x=x+1
}
```

**Output:**

```
0 1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
```

**Explanation:**

- The **next** statement in is useful when we want to skip the current iteration of a loop without terminating it.
- On encountering next, the R parser skips further evaluation and starts next iteration of the loop.in other words, the inner loop executes n-times of every execution of the outer for loop.

**Date: 03/11/2020**

**Q.NO 14: Question**

Examine the difference between typeof and class () method using R program

**Program:**

```
#Variable
num<-458
class(num)    #It gives numeric
typeof(num)   #It gives double
```

**Output:**

```
'numeric'

'double'
```

**Explanation:**

- ✶ 'class' is a property assigned to an object that determines how generic functions operate with it. It is not a mutually exclusive classification. If an object has no specific class assigned to it, such as a simple numeric vector, its class is usually the same as its mode, by convention.
- ✶ Typeof determines the (R internal) type or storage mode of any object

**Date: 03/11/2020**

**Q.NO 15: Question**

Create a function and demonstrate their features like required, keyword, default.

**Program:**

```
#Keyword Argument
keyword<-function(a,b){
  return(a+b)
}
keyword(b=1,a=3)

#default Argument
def <- function(a=0) {
  return(a+1)
}
def(5)

#required Argument
norma<-function(x){
  return(x+1)
}
norma(1)
```

**Output:**

```
4
6
2
```

**Explanation:**

- ✶ *Arguments are always named when you define a function*. When you call a function, you do not have to specify the name of the argument.
- ✶ Arguments are optional; you do not have to specify a value for them. They can have a default value, which is used if you do not specify a value for that argument yourself.
- ✶ You can use as many arguments as you like, there is no limit to the number of arguments. An argument list comprises of comma-separated values that contain the various formal arguments.

**Date: 03/11/2020**

## Q.NO 16: Question

Create a dataframe and delete the row and column. ( Use the own data values to create frame)

## Program:

```
#Dataframe creation
Name <- c("Jhon", "Lee", "Suzan", "Abhinav", "Brain", "Emma", "David")
#Variables
Subject <- c("Data Science", "Machine Learning", "Deep Learning", "Data Structures",
             "Database Managemnt System", "Operating Systems", "Python Programming")

Score <- c(56, 76, 86, 96, 73, 87, 47)

Rank <- c(5,8,6,7,9,2,1)

df = data.frame(Name,Subject,Score,Rank)
del_row_df<-df[-c(2),] # row deletion
del_col_df<-within(df, rm("Subject")) #Colmn deletion
print("After deleting Row 2")
del_row_df
print("After Deleting Subject Column")
del_col_df
```

## Output:

[1] "After deleting Row 2"

| | Name | Subject | Score | Rank |
|---|---|---|---|---|
| 1 | Jhon | Data Science | 56 | 5 |
| 3 | Suzan | Deep Learning | 86 | 6 |
| 4 | Abhinav | Data Structures | 96 | 7 |
| 5 | Brain | Database Managemnt System | 73 | 9 |
| 6 | Emma | Operating Systems | 87 | 2 |
| 7 | David | Python Programming | 47 | 1 |

[1] "After Deleting Subject Column"

| Name | Score | Rank |
|---|---|---|
| Jhon | 56 | 5 |
| Lee | 76 | 8 |
| Suzan | 86 | 6 |
| Abhinav | 96 | 7 |
| Brain | 73 | 9 |
| Emma | 87 | 2 |
| David | 47 | 1 |

## Explanation:

- -c(rowNum) is used to return a slice of a dataframe without the mentioned row i.e. row deletion
- The within function returns a subset of a dataframe with the second argument ad a function to apply to all the column rm function removes the “Subject” column out of the data frameTypeof determines the (R internal) type or storage mode of any object.

**Date: 04/11/2020**

**Q.NO 17: Question**

Write a function that turns (e.g.) a vector c("a", "b", "c") into the string "a, b, and c". Think carefully about what it should do if given a vector of length 0, 1, or 2.

**Program:**

```
#collapse function
convert=function(x) {
  #Using collapse for cancatenating into one string
  y=paste(x,collapse=",")
  print(y)
}
#Checking for 3 values
convert(c("sathish","kumar","good"))
convert(c("sathish","kumar"))
#Checking for 1 value
convert(c("sathish"))
```

**Output:**

```
[1] "sathish,kumar,good"
[1] "sathish,kumar"
[1] "sathish"
```

**Explanation:**

- **'collapse'** is a property Collapses a character vector of any length into a length 1 vector.
- Syntax - collapse(x, sep = "", width = Inf, last = "")
- **'paste'** Concatenate vectors after converting to character.

**Date: 04/11/2020**

## Q.NO 18: Question

Consider a data frame
Create a function that, given a data frame and two indexes, exchanges two values of the Code variable with each other.
For example, if the index is 1 and 3, you assign: df[1,'Code']=df[3,'Code'] df[3,'Code']=df[1,'Code']

**Program:**

```
#Values for datframe
Id=c(1:10)
Age=c(14,12,15,10,23,21,41,56,78,12)
Sex=c('F','M','M','F','M','F','M','M','F','M')
Code=letters[1:10]
#Creating a datframe
df=data.frame(Id,Age,Sex,Code)

#function that, given a data frame and two indexes, exchanges two values of the Code with each other.
change_values=function(df,firstindex,secondindex)
{
  first_value=df[firstindex,'Code']
  df[firstindex,'Code']=df[secondindex,'Code']
  df[secondindex,'Code']=first_value
  return(df)
}
#Interchnaging values
df=change_values(df,4,8)
df=change_values(df,9,1)
df=change_values(df,3,10)
df
```

**Output:**

| Id | Age | Sex | Code |
|---|---|---|---|
| 1 | 14 | F | i |
| 2 | 12 | M | b |
| 3 | 15 | M | j |
| 4 | 10 | F | h |
| 5 | 23 | M | e |
| 6 | 21 | F | f |
| 7 | 41 | M | g |
| 8 | 56 | M | d |
| 9 | 78 | F | a |
| 10 | 12 | M | c |

**Explanation:**

- ✶ A data frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column.

**Date: 03/11/2020**

**Q.NO 19: Question**

Create a function that given a numeric vector, sort this in ascending order and duplicate it by two.

**Program:**

```
#Ascending values
sort_asc=function(vec)
{
  #Mutplipying by 2
  vec=sort(vec)*2
  return(vec)
}
#input
vec=c(1,5,9,2,5,1,6,3)
#Invoking function
sort_asc(vec)
```

**Output:**

```
2  2  4  6  10  10  12  18
```

**Explanation:**

- Sort (or *order*) a vector or factor (partially) into ascending or descending order. For ordering along more than one variable
- By default, sorting is ASCENDING. Prepend the sorting variable by a minus sign to indicate DESCENDING order

**Date: 04/11/2020**

**Q.NO 20: Question**

Create a function that given a numeric vector X returns the digits 0 to 9 that are not in X. If X=0 2 4 8 the function return 1 3 5 6 7 9

**Program:**

```
#Fuction to find missing numbers in the limit
nums=function(input)
{ #limit to check
  lim=0:9
  #Checking  and saving the values missing
  output=lim[!lim %in% input]
  #%in% gives the present postion
  return(output)
}
#Assigning input
input=c(1,5,3,8)
#Invoking function
nums(input)
```

**Output:**

```
0 2 4 6 7 9
```

**Explanation:**

- %in% operator is used to identify if an element belongs to a vector or Dataframe.
- We can
  - select column of a dataframe in R using %in% operator.
  - create new variable of a column using %in% operator
  - drop column of a dataframe in R using %in% operator.

**Date: 04/11/2020**

**Q.NO 21: Question**

Create a function that given one word, return the position of word's letters on letters vector. For example, if the word is 'abc', the function will return 1 2 3.

**Program:**

```
#function to convert string to corresponding number
convert_str_num<-function(str)
{
  #extracting letters seperately
  extract = stri_extract_all(str, regex=c('\\p{L}'))
  #conerts leters to corresponding numbers
  converted = letters%in%unlist(extract)
  #Returning Value
  return(which(converted))
}
str='abc'
#Invoking function
convert(str)
```

**Output:**

```
1  2  3
```

**Explanation:**

- ✶ Regular expressions are the default pattern engine in stringr. That means when you use a pattern matching function with a bare string, it's equivalent to wrapping it in a call to REGEX()
- ✶ Regular expressions are a concise and flexible tool for describing patterns in strings.

**Date: 04/11/2020**

**Q.NO 22: Question**

Write a code to check the given string is anagram or not

**Program:**

```
#Creating a function to check for anagram
anagram<-function(str1,str2)
{
  #Splicing word into letter with regex
  str1=unlist(stri_extract_all(str1, regex=c('\\p{L}')))
  str2=unlist(stri_extract_all(str2, regex=c('\\p{L}')))
  #First Checking for length
  if (length(str1)==length(str2))
  {
   #Reverse and check and return true
  match=unique(str1%in%str2==str2%in%str1)
  return( ifelse(length(str1)==length(str2) & length(match)==1,ifelse(match==TRUE,TRUE,FALSE),FALSE))
  }
  #If length is not equal diirectly returs false
  if (length(str1)!=length(str2))
    return(FALSE)
}
#Input
str1='rotator'
str2='rotator'
#Invoking function
print(anagram(str1,str2))
```

**Output:**

```
[1] TRUE
```

**Explanation:**

- ✶ Regular expressions are the default pattern engine in stringr. That means when you use a pattern matching function with a bare string, it's equivalent to wrapping it in a call to REGEX()
- ✶ Regular expressions are a concise and flexible tool for describing patterns in strings.
- ✶ Basically, we're splitting the word up into letters. Then using the unlist function to convert this into a vector. Then sorting the vector into alphabetical order.

**Date: 05/11/2020**

**Q.NO 23: Question**

List all example files available with the `readr` library.

**Program:**

```
#Listing all example files available with the readr library.
# install.packages('readr')
print(library(readr, help, pos = 2, lib.loc = NULL))
files=readr_example()
print(files)
```

**Output:**

```
 [1] "readr"     "stats"     "graphics"  "grDevices" "utils"     "datasets"
 [7] "methods"   "base"
 [1] "challenge.csv"     "epa78.txt"         "example.log"
 [4] "fwf-sample.txt"    "massey-rating.txt" "mtcars.csv"
 [7] "mtcars.csv.bz2"    "mtcars.csv.zip"
```

**Explanation:**

- ✶ The goal of readr is to provide a fast and friendly way to read rectangular data (like csv, tsv, and fwf).
- ✶ It is designed to flexibly parse many types of data found in the wild, while still cleanly failing when data unexpectedly changes.
- ✶ If you are new to readr, the best place to start is the data import chapter in R for data science.

**Date: 05/11/2020**

**Q.NO 24: Question**

Read the `mtcars.csv` file.

**Program:**

```
#Reading the mtcars.csv files
print(mtcars)
```

**Output:**

| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mazda RX4 | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108.0 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258.0 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360.0 | 175 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 | 2 |
| Valiant | 18.1 | 6 | 225.0 | 105 | 2.76 | 3.460 | 20.22 | 1 | 0 | 3 | 1 |
| Duster 360 | 14.3 | 8 | 360.0 | 245 | 3.21 | 3.570 | 15.84 | 0 | 0 | 3 | 4 |
| Merc 240D | 24.4 | 4 | 146.7 | 62 | 3.69 | 3.190 | 20.00 | 1 | 0 | 4 | 2 |
| Merc 230 | 22.8 | 4 | 140.8 | 95 | 3.92 | 3.150 | 22.90 | 1 | 0 | 4 | 2 |
| Merc 280 | 19.2 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.30 | 1 | 0 | 4 | 4 |
| Merc 280C | 17.8 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.90 | 1 | 0 | 4 | 4 |
| Merc 450SE | 16.4 | 8 | 275.8 | 180 | 3.07 | 4.070 | 17.40 | 0 | 0 | 3 | 3 |
| Merc 450SL | 17.3 | 8 | 275.8 | 180 | 3.07 | 3.730 | 17.60 | 0 | 0 | 3 | 3 |
| Merc 450SLC | 15.2 | 8 | 275.8 | 180 | 3.07 | 3.780 | 18.00 | 0 | 0 | 3 | 3 |
| Cadillac Fleetwood | 10.4 | 8 | 472.0 | 205 | 2.93 | 5.250 | 17.98 | 0 | 0 | 3 | 4 |
| Lincoln Continental | 10.4 | 8 | 460.0 | 215 | 3.00 | 5.424 | 17.82 | 0 | 0 | 3 | 4 |

**Explanation:**

- We used the **mtcars** data set that is built-in to the **R** distribution .
- **mtcars** data comes from the 1974 Motor Trend magazine.
- The data includes fuel consumption data, and ten aspects of car design for then-current car models

**Date: 05/11/2020**

**Q.NO 25: Question**

Read the first 10 lines from the `mtcars.csv` file.

**Program:**

```
#Reading a the first 10 lines from the mtcars.csv file.
head(mtcars,10)
```

**Output:**

| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mazda RX4 | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108.0 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258.0 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360.0 | 175 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 | 2 |
| Valiant | 18.1 | 6 | 225.0 | 105 | 2.76 | 3.460 | 20.22 | 1 | 0 | 3 | 1 |
| Duster 360 | 14.3 | 8 | 360.0 | 245 | 3.21 | 3.570 | 15.84 | 0 | 0 | 3 | 4 |
| Merc 240D | 24.4 | 4 | 146.7 | 62 | 3.69 | 3.190 | 20.00 | 1 | 0 | 4 | 2 |
| Merc 230 | 22.8 | 4 | 140.8 | 95 | 3.92 | 3.150 | 22.90 | 1 | 0 | 4 | 2 |
| Merc 280 | 19.2 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.30 | 1 | 0 | 4 | 4 |

**Explanation:**

- We used the **mtcars** data set that is built-in to the **R** distribution .
- **mtcars** data comes from the 1974 Motor Trend magazine.
- The data includes fuel consumption data, and ten aspects of car design for then-current car models.
- Head function with specified parameter gives top rows of specified datax

**Date: 06/11/2020**

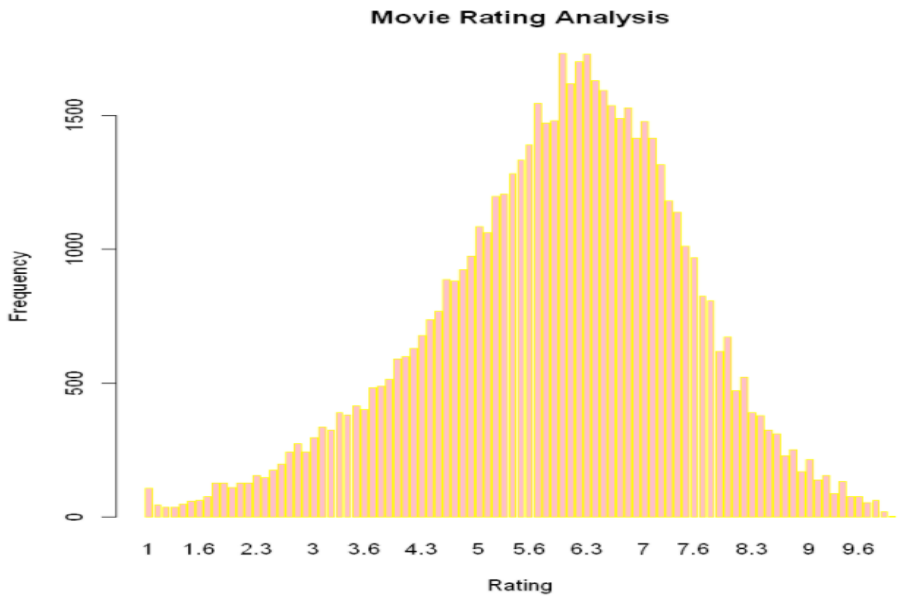
**Q.NO 26: Question**

**Program:**

Bar plot – Movies Dataset

```
#Bar Chart
#Importing Required Packages
library(ggplot2movies)
#taking rating column values from movies dataset
rating=movies$rating
#changing raw data to frequency values
table=table(rating)
#barplot(table name,horiz = true)
#deafult horiz is false
barplot(table,                                  #tablename
        main ="Movie Rating Analysis",          #header_name
        xlab ="Rating",                         #xlabel
        ylab ="Frequency",                      #ylabel
        border="yellow",                        #bordercolour
        col = "pink"                            #colours
)
```

**Output:**




**Explanation:**

- ✶ A bar chart represents data in rectangular bars with length of the bar proportional to the value of the variable. R uses the function **barplot()** to create bar charts.

**Date: 06/11/2020**

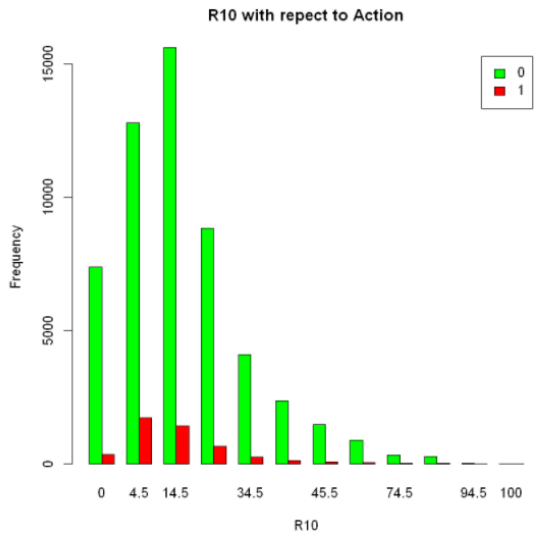
**Q.NO 27: Question**

**Program:**

Group BarGraph – Movies Dataset

```
#Bar Chart
#Importing Required Packages
library(ggplot2movies)
data=table(head(movies$r4,100),head(movies$r8,100))
# GROUPED BARGRAPH
barplot(data,
        ylab="Frequency",
        main="Double Bar Plot",
        xlab = "Rating",beside=TRUE,
        legend=rownames(data),
        col=c("green","red"))
```

**Output:**




**Explanation:**

- ✶ A bar chart represents data in rectangular bars with length of the bar proportional to the value of the variable. R uses the function **barplot()** to create bar charts.

**Date: 06/11/2020**

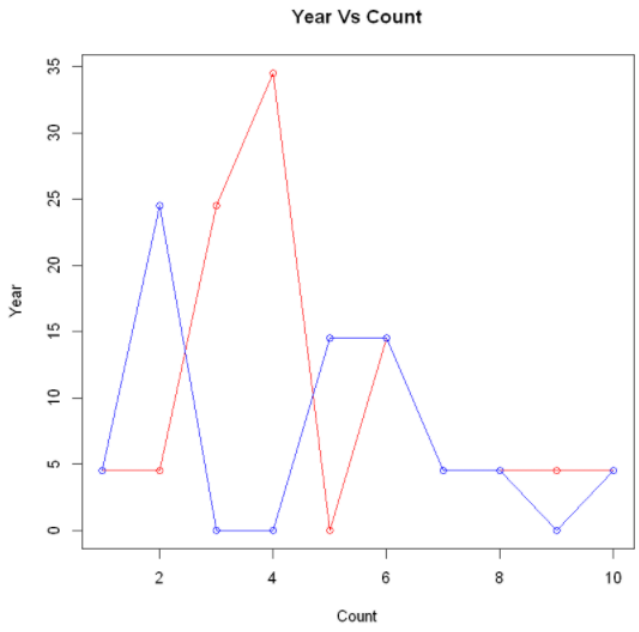
**Q.NO 28: Question**

**Program:**

Line Chart – Movies Dataset

```
#Line plot
#Importing Required Packages
library(ggplot2movies)
#Line chart only accepts numbers(no string)
plot(head(movies$r9,10),
     type = "o",
     col = "red",
     xlab = "Count",
     ylab = "Year",
     main = "Year Vs Count")
#Plotting multiple lines for comparison
lines(head(movies$r4,10), type = "o", col = "blue")
```

**Output:**




**Explanation**

- ✶ A line chart is a graph that connects a series of points by drawing line segments between them. These points are ordered in one of their coordinate (usually the x-coordinate) ".

**Date: 06/11/2020**

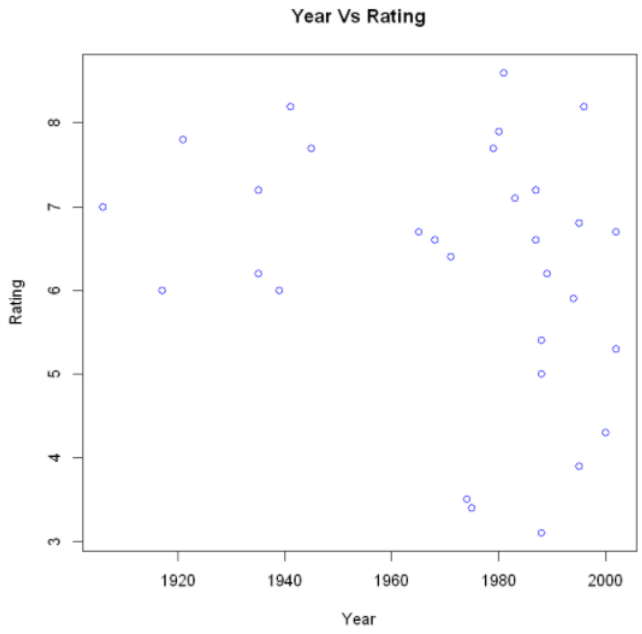
**Q.NO 29: Question**

**Program:**

Scatter plot – Movies Dataset

```
#Scatter plot
#Importing Required Packages
library(ggplot2movies)
#plot(x,y)
plot(head(movies$year,30),head(movies$rating,30),
     xlab = "Year",
     ylab = "Rating",
     main = "Year Vs Ratin",
     col = "Blue"
     )
```

**Output:**




**Explanation:**

- ✶ Scatterplots show many points plotted in the Cartesian plane. Each point represents the values of two variables. One variable is chosen in the horizontal axis and another in the vertical axis.

**Date: 06/11/2020**

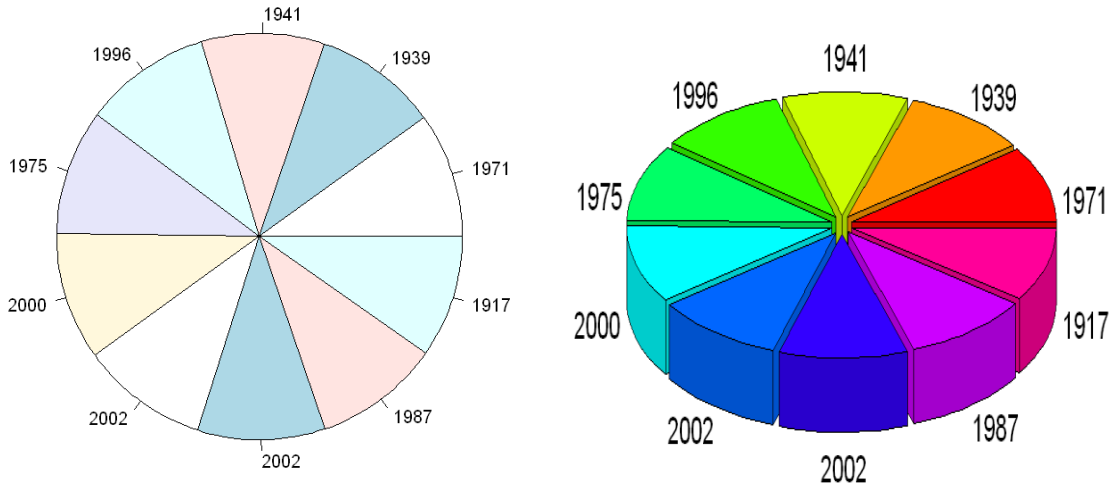
**Q.NO 29: Question**

**Program:**

Pie Chart – Movies Dataset

```
#Scatter plot
#Importing Required Packages
library(ggplot2movies)
library(plotrix)
#pie chart
pie(head(movies$year,10),labels = head(movies$year,10))
#3d pie chart
pie3D(head(movies$year,10),explode = 0.05,labels = head(movies$year,10))
```

**Output:**




**Explanation:**

- ✶ pie-chart is a representation of values as slices of a circle with different colors. The slices are labeled and the numbers corresponding to each slice is also represented in the chart.

**Date: 06/11/2020**

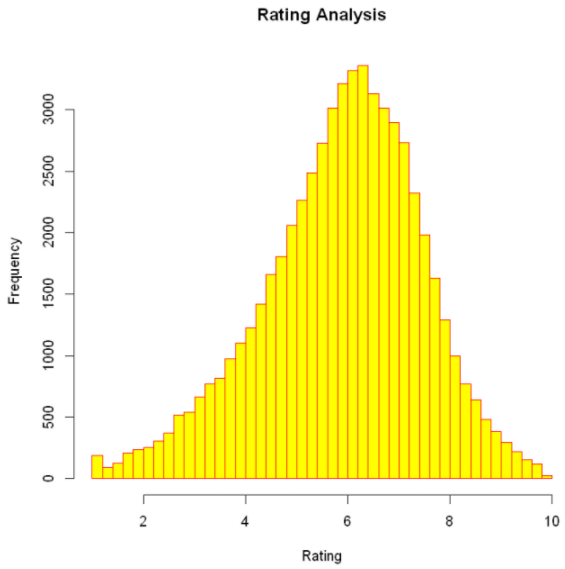
**Q.NO 30: Question**

**Program:**

Hist plot – Movies Dataset

```
#Hist plot
#Importing Required Packages
library(ggplot2movies)
#accesing ratings from movies dataset
rating=(movies$rating)
hist(rating,
     xlab = "Rating",
     col = "yellow",
     border = "red",
     breaks = 40,
     main="Rating Analysis")
```

**Output:**




**Explanation:**

- ✶ A histogram represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to bar chat but the difference is it groups the values into continuous ranges.

**Date: 06/11/2020**

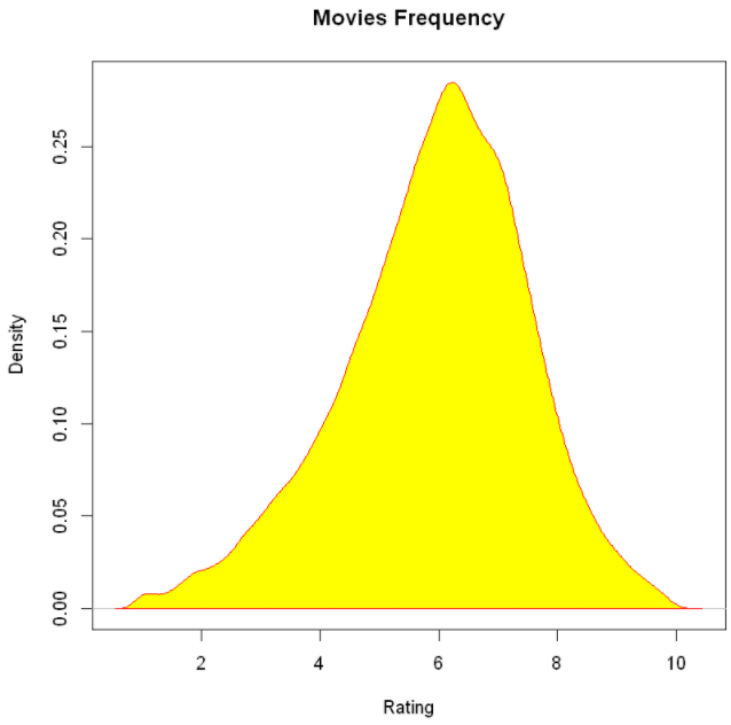
**Q.NO 31: Question**

**Program:**

difference plot – Movies dataset

```
#Hist plot
#Importing Required Packages
library(ggplot2movies)
data=density(movies$rating)
plot(data,main="Movies Frequency", xlab="Rating")
#kde plotting
polygon(data,col='yellow',border="red")
```

**Output:**




**Explanation:**

- ✶ polygon draws the polygons whose vertices are given in x and y.

**Date: 06/11/2020**

**Q.NO 32: Question**

**Program:**

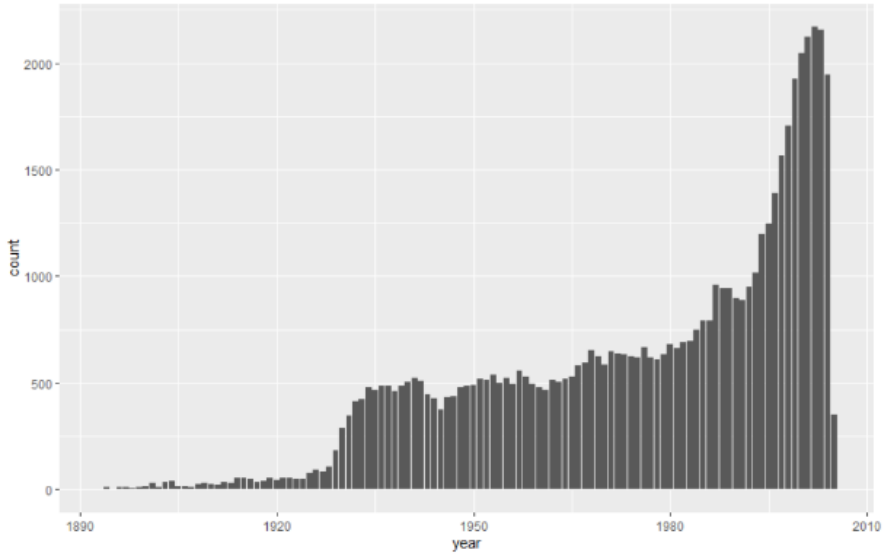
Demonstrate the ggplot2 layer

```
library(ggplot2)
library(ggplot2movies)
dplot <- ggplot(movies, aes(year))

dplot + geom_bar()
dplot + geom_bar(position = "fill")

dplot + geom_bar(position = "dodge")
```

**Output:**




**Explanation:**

- ggplot() initializes a ggplot object. It can be used to declare the input data frame for a graphic and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.

**Date: 06/11/2020**

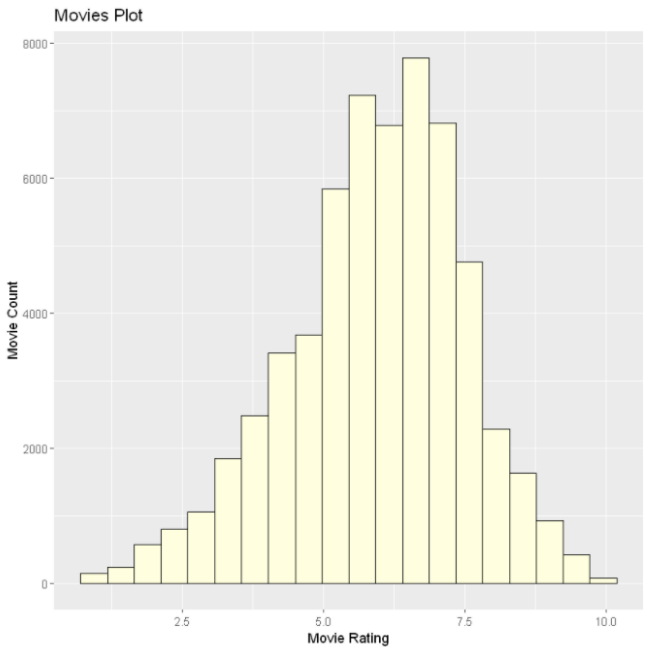
**Q.NO 33: Question**

**Program:**

Ggplot – histogram plotting

```
library(ggplot2)
library(ggplot2movies)
movies<-ggplot(data=movies,aes(x=rating))
movies<-movies+geom_histogram(bins = 20,
                              color = "black",
                              fill="light yellow",
                              alpha=1)+xlab("Movie Rating")+
                              ylab("Movie Count")+
                              ggtitle("Movies Plot")
print(movies)
```

**Output:**




**Explanation:**

✶ ggplot() initializes a ggplot object. It can be used to declare the input data frame for a graphic and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.

**Date: 06/11/2020**

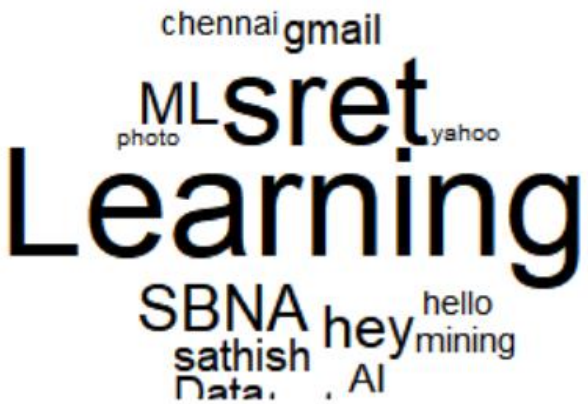
**Q.NO 34: Question**

**Program:**

Word Cloud

```
#Importing Required Packages
library(wordcloud)
data=read.csv("E:/R Programs For Practice/DAY - 5/Class/Data/file.csv",
              header= TRUE)
head(data)
#Creates a word cloud with parameters value and frequency
#worcloud(word = dataframe_name$words,dataframe_name$words,min.freq="Lea
wordcloud(words=data$word,
          freq=data$freq,
          min.freq=2,
          max.words = 50,
          random.order = FALSE)
```

**Output:**




**Explanation:**

- ✶ wordcount() counts words. Currently a "word" is a clustering of characters separated from another clustering of charactersby at least 1 space. That is the law.

**Date: 07/11/2020**

**Q.NO 35: Question**

Perform the following operation:

1. T-Test (score 1 and score 2)

**Program:**

```
#test
Score1 <- c(3,3,3,12,15,16,17,19,23,24,32)
Score2 <- c(20,13,13,20,29,32,23,20,25,15,30)
#t.test()
print(t.test(Score1,Score2))
```

**Output:**

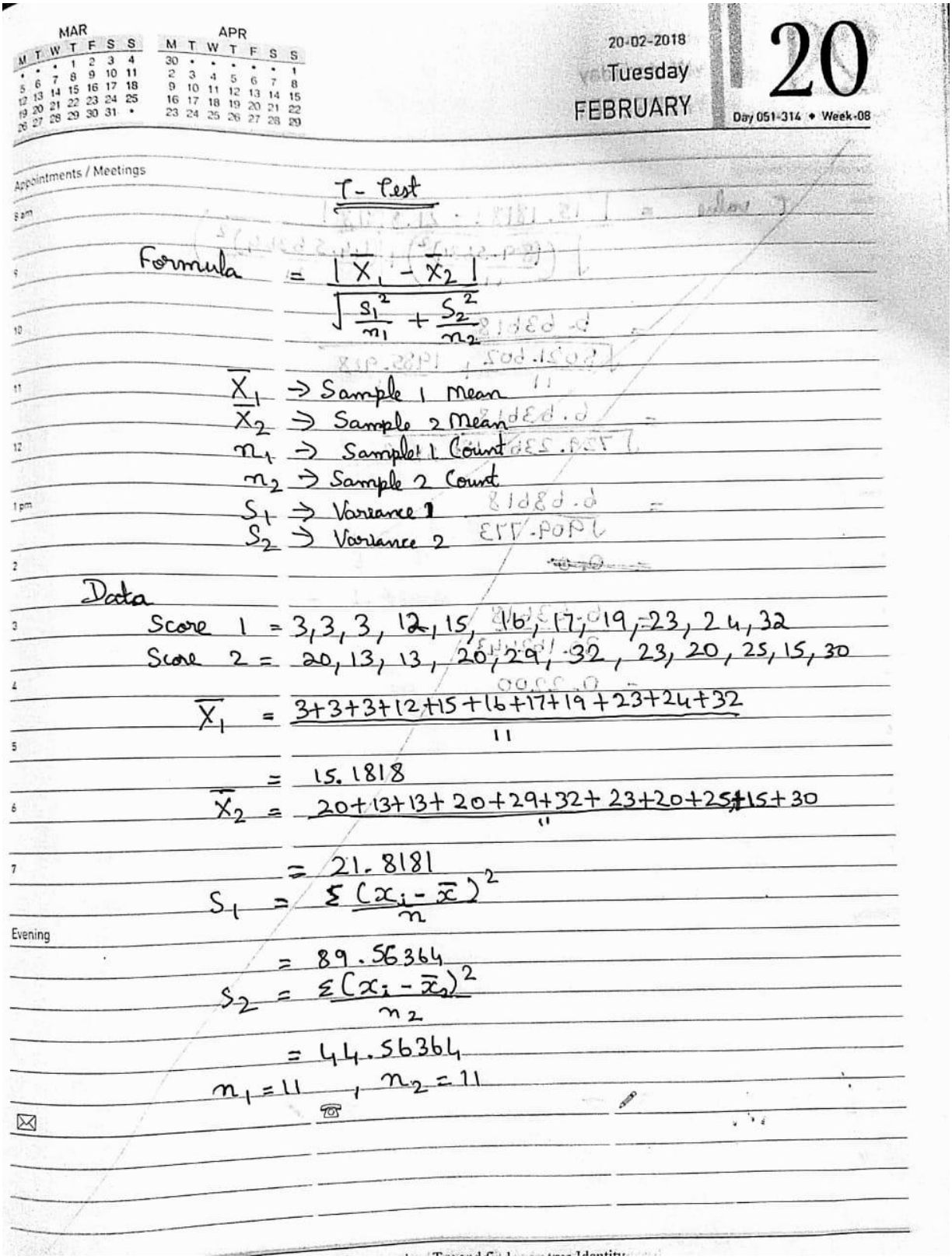
```
        Welch Two Sample t-test

data:  Score1 and Score2
t = -1.9005, df = 17.977, p-value = 0.07353
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -13.9732668   0.7005395
sample estimates:
mean of x mean of y
 15.18182  21.81818
```

**Explanation:**

- ✶ A t-test allows us to compare the average values of the two data sets and determine if they came from the same population.
- ✶ In the above examples, if we were to take a sample of students from class A and another sample of students from class B, we would not expect them to have exactly the same mean and standard deviation.

Manual calculation for the T-value



20-02-2018
Tuesday
FEBRUARY
20
Day 051-314 • Week-08

Appointments / Meetings

T- Test

Formula $= \frac{|\bar{X}_1 - \bar{X}_2|}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}$

$\bar{X}_1 \Rightarrow$ Sample 1 Mean
$\bar{X}_2 \Rightarrow$ Sample 2 Mean
$n_1 \Rightarrow$ Sample 1 Count
$n_2 \Rightarrow$ Sample 2 Count
$S_1 \Rightarrow$ Variance 1
$S_2 \Rightarrow$ Variance 2

Data

Score 1 = 3, 3, 3, 12, 15, 16, 17, 19, 23, 24, 32
Score 2 = 20, 13, 13, 20, 29, 32, 23, 20, 25, 15, 30

$\bar{X}_1 = \frac{3+3+3+12+15+16+17+19+23+24+32}{11}$

$= 15.1818$

$\bar{X}_2 = \frac{20+13+13+20+29+32+23+20+25+15+30}{11}$

$= 21.8181$

$S_1 = \frac{\Sigma (x_i - \bar{x})^2}{n}$

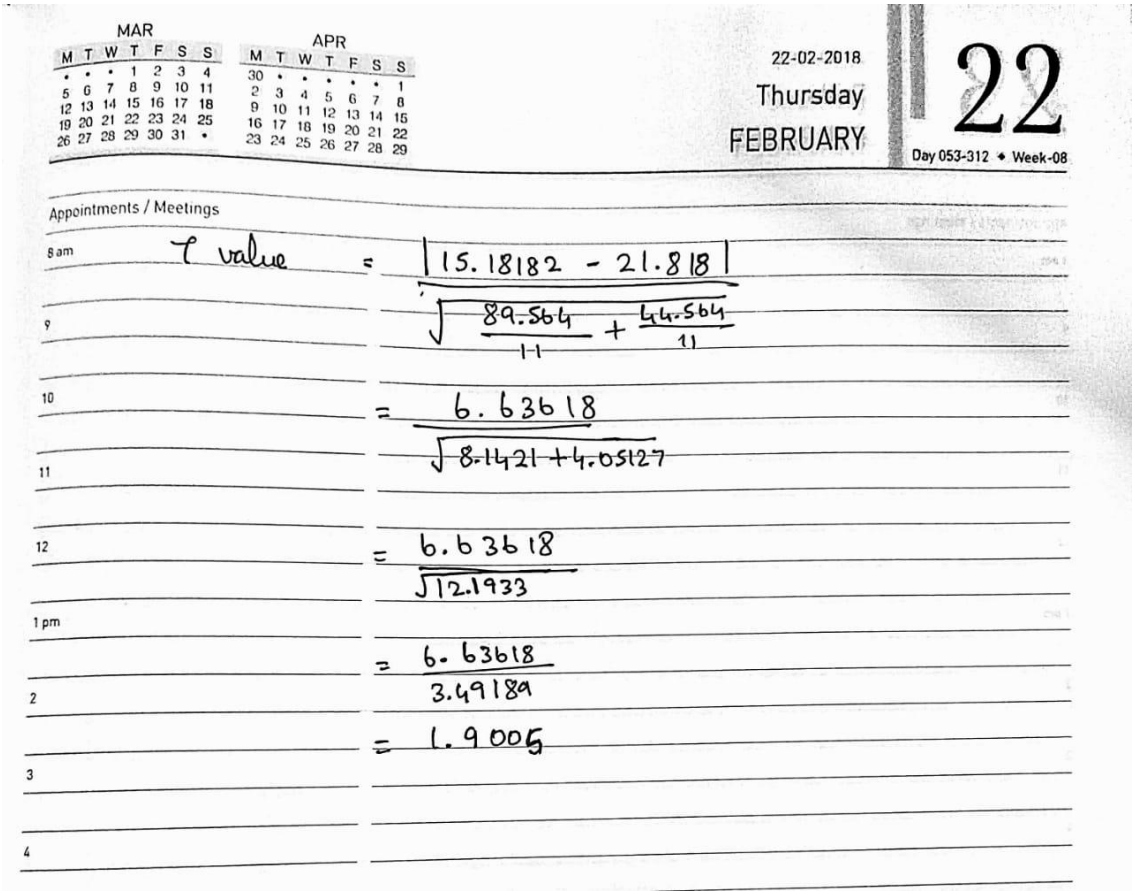
Evening

$= 89.56364$

$S_2 = \frac{\Sigma (x_i - \bar{x}_2)^2}{n_2}$

$= 44.56364$

$n_1 = 11$, $n_2 = 11$




## Excel sheet R- Program

| Subject | Score1 | Score2 |
|---|---|---|
| 1 | 3 | 20 |
| 2 | 3 | 13 |
| 3 | 3 | 13 |
| 4 | 12 | 20 |
| 5 | 15 | 29 |
| 6 | 16 | 32 |
| 7 | 17 | 23 |
| 8 | 19 | 20 |
| 9 | 23 | 25 |
| 10 | 24 | 15 |
| 11 | 32 | 30 |

| | | |
|---|---|---|
| Mean | 15.18182 | 21.8182 |
| Std | 9.463807 | 6.6756 |
| Variance | 89.56364 | 44.5636 |
| Count | 11 | 11 |

| | |
|---|---|
| T-Test (P value) | 0.071879 |
| T Value manually | 0.19004 |
| Degree of freedom | 20 |
| T Value for 20 with 5 percent error (critical value) | 1.725 |

Critical value is higher than calculated T Value.So we can reject NULL HYPOTHESIS

**Date: 07/11/2020**

**Q.NO 35: Question**

**Program:**

ANOVA Excel sheet

| | A | x-mean | (x-mean)^2 | B | x-mean | (x-mean)^2 | C | x-mean | (x-mean)^2 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | -5.71429 | 32.65306122 | 1 | -5.5714286 | 31.04081633 | 1 | -5 | 25 | | |
| | 2 | -3.71429 | 13.79591837 | 2 | -4.5714286 | 20.89795918 | 4 | -2 | 4 | | |
| | 3 | -2.71429 | 7.367346939 | 3 | -3.5714286 | 12.75510204 | 5 | -1 | 1 | | |
| | 5 | -0.71429 | 0.510204082 | 9 | 2.4285714 | 5.897959184 | 5 | -1 | 1 | | |
| | 8 | 2.285714 | 5.224489796 | 10 | 3.4285714 | 11.75510204 | 8 | 2 | 4 | | |
| | 10 | 4.285714 | 18.36734694 | 10 | 3.4285714 | 11.75510204 | 9 | 3 | 9 | | |
| | 12 | 6.285714 | 39.51020408 | 11 | 4.4285714 | 19.6122449 | 10 | 4 | 16 | | |
| Sum | 40 | 0 | 117.4285714 | 46 | 0 | 113.7142857 | 42 | 0 | 60 | | |
| Mean | 5.714286 | | | 6.571429 | | | 6 | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| SSW | 291.1429 | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | Stacked group | x-Mean | (x-mean)^2 | | | | | | |
| | | | 0 | -6.09524 | 37.151927 | | | | Total sum of square | 293.8095 | |
| | | | 2 | -4.09524 | 16.770975 | | | | SSW | 291.1429 | |
| | | | 3 | -3.09524 | 9.5804989 | | | | Sum of square b/w group | 2.666667 | |
| | | | 5 | -1.09524 | 1.1995465 | | | | No.of sample | 2 | |
| | | | 8 | 1.904762 | 3.6281179 | | | | no.of element-num of group | 18 | |
| | | | 10 | 3.904762 | 15.247166 | | | | | | |
| | | | 12 | 5.904762 | 34.866213 | | | | | | |
| | | | 1 | -5.09524 | 25.961451 | | | | Sum of square b/w degree of freedom Num | 1.333333 | |
| | | | 2 | -4.09524 | 16.770975 | | | | Sum of square within degree of freedom denominator | 16.1746 | |
| | | | 3 | -3.09524 | 9.5804989 | | | | | | |
| | | | 9 | 2.904762 | 8.4376417 | | | | F-static value | 0.082434 | |
| | | | 10 | 3.904762 | 15.247166 | | | | | | |
| | | | 10 | 3.904762 | 15.247166 | | | | | | |
| | | | 11 | 4.904762 | 24.056689 | | | | | | |
| | | | 1 | -5.09524 | 25.961451 | | | | | | |
| | | | 4 | -2.09524 | 4.3900227 | | | | | | |
| | | | 5 | -1.09524 | 1.1995465 | | | | | | |
| | | | 5 | -1.09524 | 1.1995465 | | | | | | |
| | | | 8 | 1.904762 | 3.6281179 | | | | | | |
| | | | 9 | 2.904762 | 8.4376417 | | | | | | |
| | | | 10 | 3.904762 | 15.247166 | | | | | | |
| | | | | | | | | | | | |
| | | Mean | 6.095238095 | | | | | | | | |
| | | | | | | | | | | | |
| | | Sum | | | 293.80952 | | | | | | |

**Explanation:**

**Date: 06/11/2020**

## R Program

**Code:**

```
#Anova Test
A<-c(0,2,3,5,8,10,12)
B<-c(1,2,3,9,10,10,11)
C<-c(1,4,5,5,8,9,10)
#Making data as datafrane
combined_groups <-data.frame(cbind(A,B,C))
print(combined_groups)
#stacking into one column
stack_group = stack(combined_groups)
print(stack_group)
#applying aov anova function
anova  = aov(values~ind,data = stack_group)
print(summary(anova))
```

**Output:**

```
             Df Sum Sq Mean Sq F value Pr(>F)
ind           2   2.67   1.333   0.082  0.921
Residuals    18 291.14  16.175
```

**Explanation:**

- An **ANOVA test** is a way to find out if survey or experiment results are significant.
- In other words, they help you to figure out if you need to reject the null hypothesis or accept the alternate hypothesis.
- Basically, you're **testing** groups to see if there's a difference between them.

**Date: 06/11/2020**

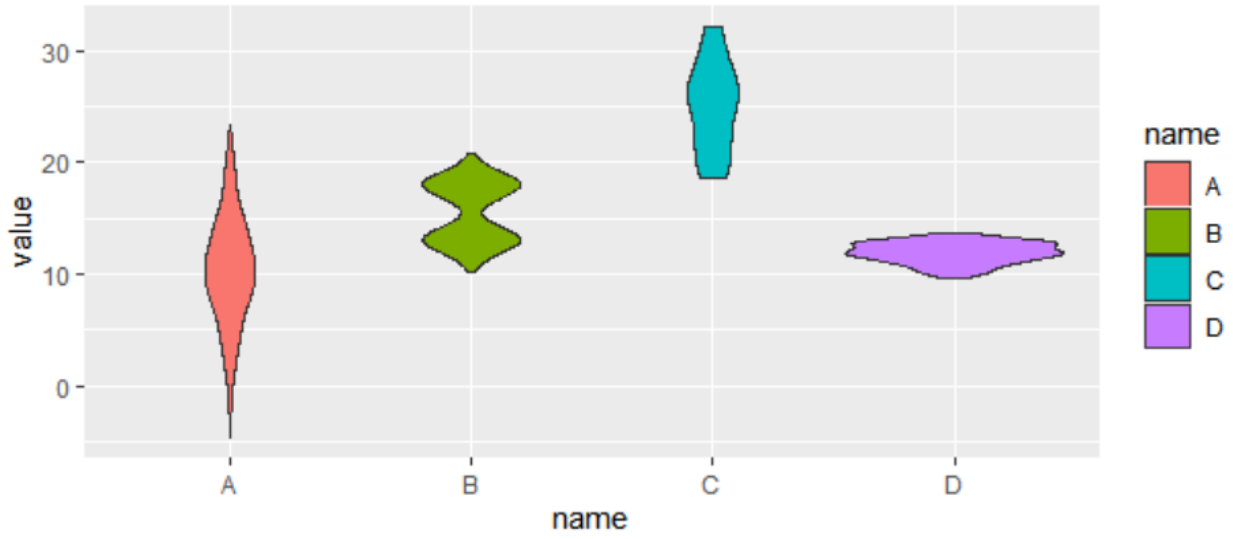
**Q.NO 36: Question**

**Program:**

Draw the Violin plot

```
#Importing Required Packages
library(ggplot2)
data <- data.frame(
  name=c( rep("A",500), rep("B",500),
          rep("B",500), rep("C",20), rep('D', 100)  ),
  value=c( rnorm(500, 10, 5), rnorm(500, 13, 1),
           rnorm(500, 18, 1), rnorm(20, 25, 4), rnorm(100, 12, 1) )
)
#Ggplot()
p <- ggplot(data, aes(x=name, y=value, fill=name)) +
  geom_violin()
print(p)
```

**Output:**




**Explanation:**

- A **violin plot** is a method of **plotting** numeric data. It is similar to a box **plot**, with the addition of a rotated kernel density **plot** on each side.
- While a box **plot** only shows summary statistics such as mean/median and interquartile ranges, the **violin plot** shows the full distribution of the data.

**Date: 07/11/2020**

**Q.NO 37: Question**

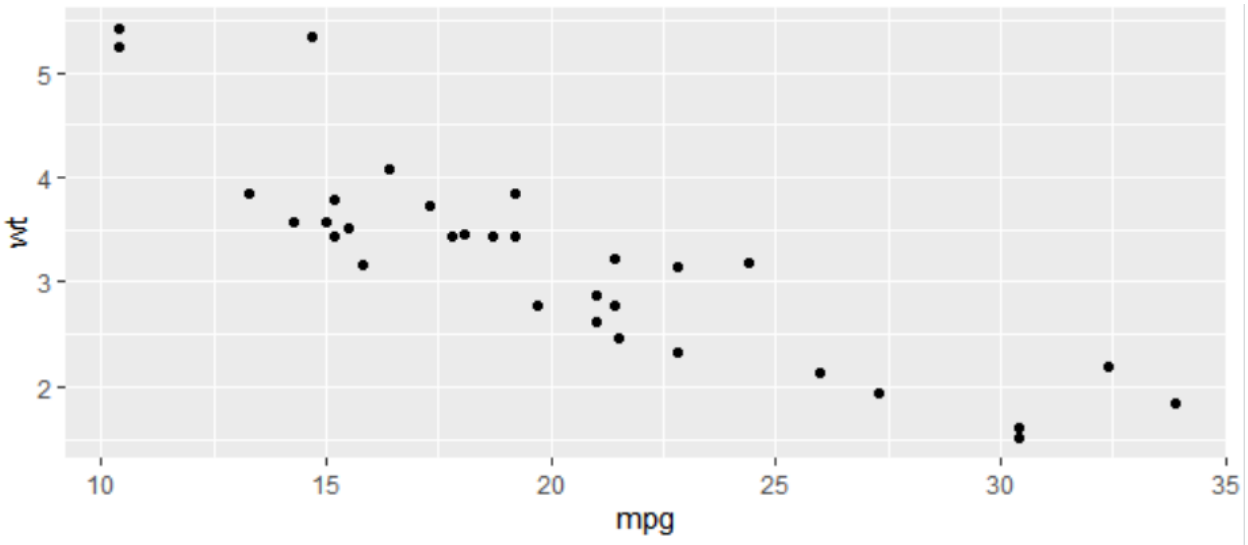
**Program:**

Draw the Qplot for 5 Types

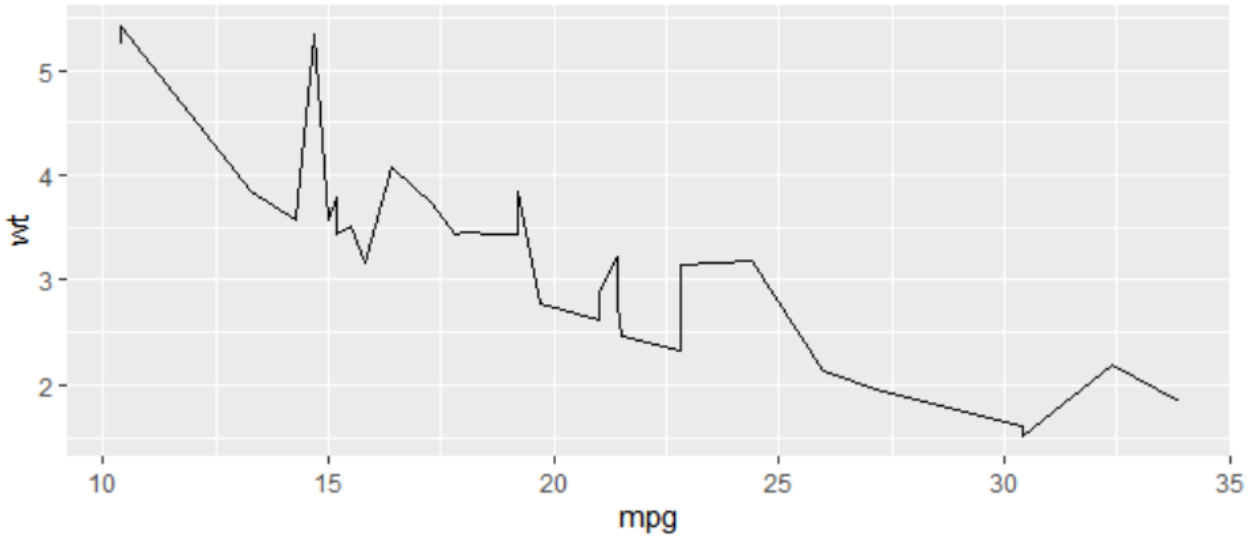
```
#qqplot
#Plotting with 5 geom
#geom = point
qplot(mpg, wt, data = mtcars, geom = "point")
#geom = line
qplot(mpg, wt, data = mtcars, geom = "line")
#geom = histogram
qplot(mpg,data = mtcars, geom = "histogram",bins = 30)
#geom = path
qplot(mpg,wt,data = mtcars, geom = "path")
#geom = smooth
qplot(mpg,wt,data = mtcars, geom = "smooth")
```

**Output:**

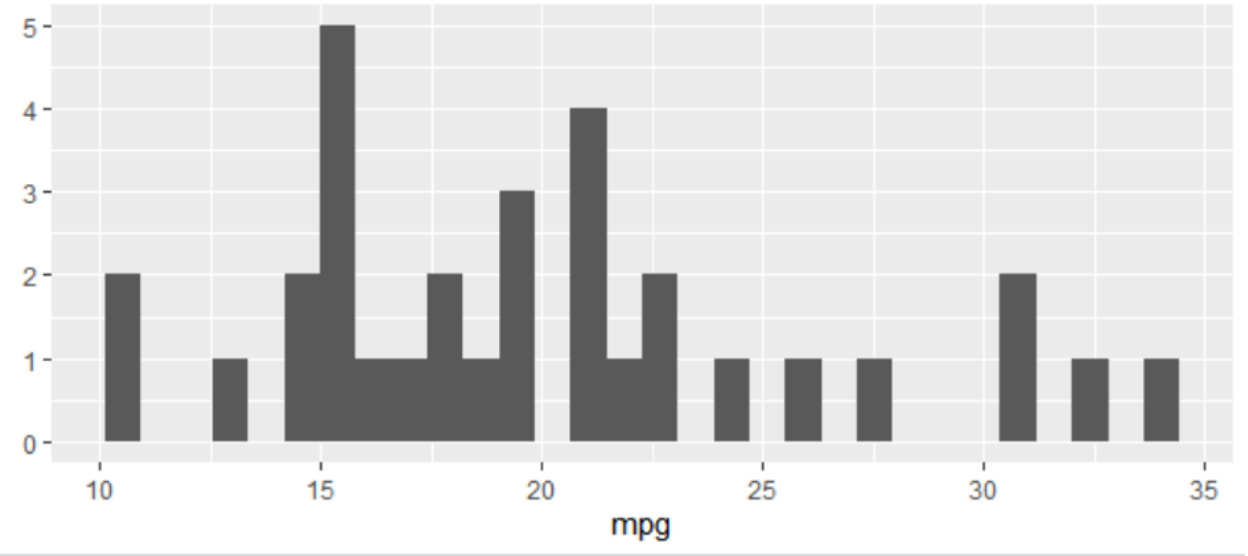
**Point plot**



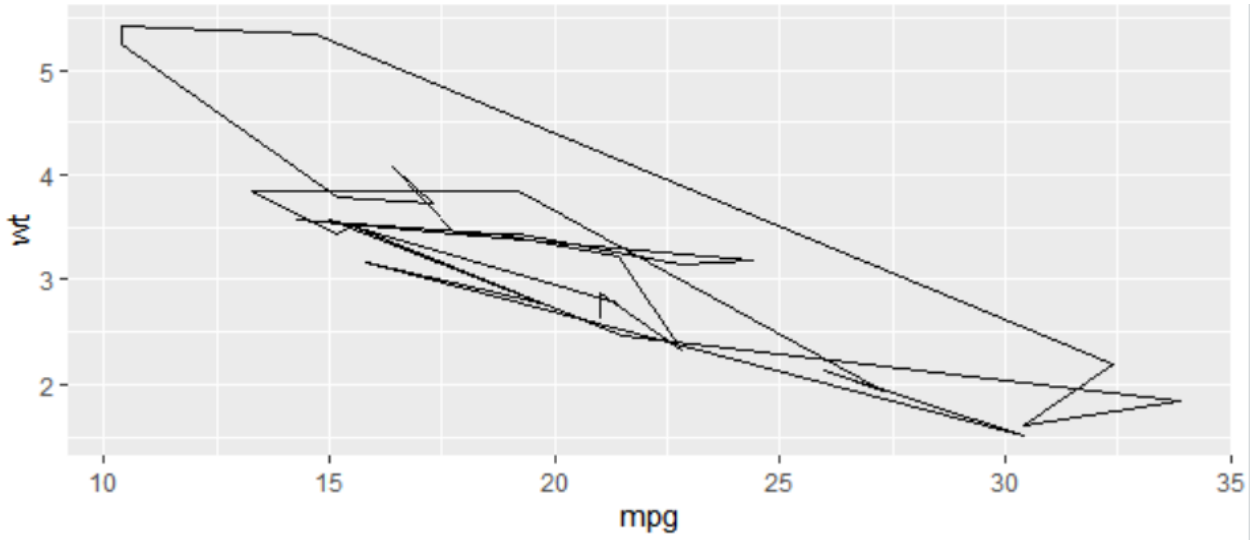
## Line plot



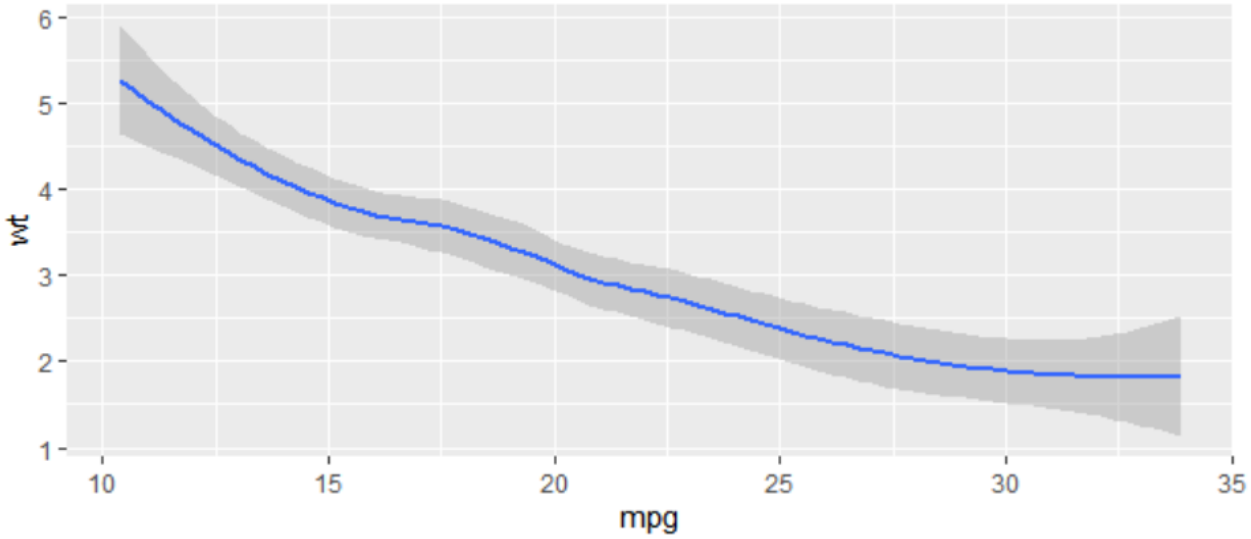

## Histogram



## Path




## Smooth




**Explanation:**

- ✶ **qplot**() is a shortcut designed to be familiar if you're used to base plot() .
- ✶ It's a convenient wrapper for creating a number of different types of plots using a consistent calling scheme.

**Date: 07/11/2020**

**Q.NO 38: Question**

**Program:**

Demonstrate the skewness and kurtosis using built-in dataset or your own dataset to get the data summary.

```
#Importing Required Packages
library(e1071)
#Reading Csv file
df = read.csv("E:/R Programs For Practice/DAY - 6/Data/mark.csv")
#skewness
cat("Skewness",skewness(df$mark),"\n")
plot(density(df$mark))
polygon(density(df$mark), col="red", border="blue")
#kurtosis
cat("kurtosis: ",kurtosis(df$mark))
```

**Output:**

```
Skewness -0.4277135
kurtosis:  -0.7261525
```

**Explanation:**

- **Skewness** is a measure of symmetry, or more precisely, the lack of symmetry.
- **Kurtosis** is a measure of whether the data are heavy-tailed or light-tailed relative to a normal distribution.

**Date: 09/11/2020**

**Q.NO 39: Question**

**Program:**

Demonstrate the skewness and kurtosis using built-in dataset or your own dataset to get the data summary.

```
#Importing Required Packages
library(e1071)
#Reading Csv file
df = read.csv("E:/R Programs For Practice/DAY - 6/Data/mark.csv")
#skewness
cat("Skewness",skewness(df$mark),"\n")
plot(density(df$mark))
polygon(density(df$mark), col="red", border="blue")
#kurtosis
cat("kurtosis: ",kurtosis(df$mark))
```

**Output:**

```
Skewness -0.4277135
kurtosis:  -0.7261525
```

**Explanation:**

- ✶ **Skewness** is a measure of symmetry, or more precisely, the lack of symmetry.
- ✶ **Kurtosis** is a measure of whether the data are heavy-tailed or light-tailed relative to a normal distribution.

**Date: 09/11/2020**

**Q.NO 40: Question**

**Program:**

Create a variable called my_pattern and implement the required pattern for finding one digit and one uppercase alphanumeric character, in variable text1. This time, combine predefined classes in the regex pattern. Use function grepl to verify if the searched pattern exists on the string.

```
str<- "Data Science 2020"
pattern = grepl('[A-Z]',str) && grepl('[0-9]',str)
pattern
```

**Output:**

```
> pattern
[1] TRUE
```

**Explanation:**

- The implementation of those patterns can be performed through several base-r functions, such as:
  - `Grep`
  - `grepl`

**Date: 09/11/2020**

**Q.NO 41: Question**

**Program:**

Using the sub function, replace the pattern found on the previous exercice by the string " is not " Place the resulting string in text2 variable.

```
str<- "Data Science 2020"
pattern =  "[[:upper:][:digit:]]"

replaced <- sub(pattern," is not ",str)

print(replaced)
```

**Output:**

```
> print(replaced)
[1] " is not ata Science 2020"
```

**Explanation:**

- ✶ The implementation of those patterns can be performed through several base-r functions, such as:
  - `Grep`
  - `grepl`

**Date: 09/11/2020**

**Q.NO 42: Question**

**Program:**

Probability that a normal random variable with mean 22 and variance 25

(i) lies between 16.2 and 27.5 ii) is greater than 29 (iii) is less than 17 (iv) is less than 15 or greater than 25

```
#lies between 16.2 and 27.5
pnorm(27.5,22,sd=5)-pnorm(16.2,22,sd=5)
#is greater than 29
1-pnorm(29,22,sd=5)
#is less than 17
pnorm(17,22,sd=5)
#is less than 15 or greater than 25
pnorm(15,22,sd=5)+1-pnorm(25,22,sd=5)
```

**Output:**

```
> #lies between 16.2 and 27.5
> pnorm(27.5,22,sd=5)-pnorm(16.2,22,sd=5)
[1] 0.7413095
> #is greater than 29
> 1-pnorm(29,22,sd=5)
[1] 0.08075666
> #is less than 17
> pnorm(17,22,sd=5)
[1] 0.1586553
> #is less than 15 or greater than 25
> pnorm(15,22,sd=5)+1-pnorm(25,22,sd=5)
[1] 0.3550098
>
```

**Explanation:**

★ the **p-norm** is a **norm** on suitable real vector spaces given by the **p**th root of the sum (or integral) of the **p**th-powers of the absolute values of the vector components..

**Date: 09/11/2020**

**Q.NO 43: Question**

**Program:**

Probability that in 60 tosses of a fair coin the head comes up (i) 20,25 or 30 times (ii) less than 20 times (iii) between 20 and 30 times

```
#Probability that in 60 tosses of a fair coin the head comes up
#20,25 or 30 times
sum(dbinom(c(20,25,30),60,prob=0.5))
#less than 20 times
pbinom(19,60,prob=0.5)
#between 20 and 30 times
pbinom(30,60,prob=0.5)-pbinom(20,60,prob=0.5)
```

**Output:**

```
> #Probability that in 60 tosses of a fair coin the head comes up
> #20,25 or 30 times
> sum(dbinom(c(20,25,30),60,prob=0.5))
[1] 0.1512435
> #less than 20 times
> pbinom(19,60,prob=0.5)
[1] 0.003108801
> #between 20 and 30 times
> pbinom(30,60,prob=0.5)-pbinom(20,60,prob=0.5)
[1] 0.5445444
```

**Explanation:**

- **pbinom**() function in R Language is used to compute the value of negative binomial cumulative density.

**Date: 10/11/2020**

**Q.NO 44: Question**

**Program:**

The coin is flipped ten times. Find the probability of 7 heads occurring.

```
#The coin is flipped ten times. Find the probability of 7 heads occurring.
#BinomialDistribution
cat("probability of 7 heads occurring in 10 times = ",pbinom(7,size=10,prob=0.5))
```

**Output:**

```
> cat("probability of 7 heads occurring in 10 times = ",pbinom(7,size=10,prob=0.5))
probability of 7 heads occurring in 10 times =  0.9453125
```

**Explanation:**

- ✶ The **binomial distribution** is a discrete probability distribution. It describes the outcome of *n* independent trials in an experiment. Each trial is assumed to have only two outcomes, either success or failure.
- ✶ This function gives the cumulative probability of an event. It is a single value representing the probability.

**Date: 10/11/2020**

**Q.NO 45: Question**

**Program:**

A card is selected three times (and replaced). Find the probability of 2 face cards occurring.

```
# A card is selected three times (and replaced). Find the probability of 2 face cards occurrin
cat("probability of 2 face cards occurring in three times = ",pbinom(2,size=3,prob=1/52))
```

**Output:**

```
> cat("probability of 2 face cards occurring in three times = ",pbinom(2,size=3,prob=1/52))
probability of 2 face cards occurring in three times =  0.9999929
```

**Explanation:**

- ✶ The **binomial distribution** is a discrete probability distribution. It describes the outcome of *n* independent trials in an experiment. Each trial is assumed to have only two outcomes, either success or failure.
- ✶ This function gives the cumulative probability of an event. It is a single value representing the probability.

**Date: 10/11/2020**

**Q.NO 46: Question**

**Program:**

A student decides to guess on a section of his ACT test. The section contains 50 multiple choice questions and each question has 5 possible answers. Find the expected number of correct responses.

```
# A student decides to guess on a section of his ACT test.
# The section contains 50 multiple choice questions and each question has 5 possible answers.
# Find the expected number of correct responses.
cat("probabilty of correct responses = ",pbinom(4,size=50,prob=0.2))
```

**Output:**

```
> cat("probabilty of correct responses = ",pbinom(4,size=50,prob=0.2))
probabilty of correct responses =  0.01849602
>
```

**Explanation:**

- The **binomial distribution** is a discrete probability distribution. It describes the outcome of *n* independent trials in an experiment. Each trial is assumed to have only two outcomes, either success or failure.
- This function gives the cumulative probability of an event. It is a single value representing the probability.

**Date: 10/11/2020**

**Q.NO 47: Question**

**Program:**

A company ships 5000 cell phones. They are expected to last an average of 10,000 hours before needing repair; with a standard deviation of 500 hours.

Assume the survival time of the phones are normally distributed. If a phone is randomly selected to be tracked for repairs find the expected number that needs repair,

a) after 11,000 hours

```
#A company ships 5000 cell phones.
#They are expected to last an average of 10,000 hours before needing repair;
#with a standard deviation of 500 hours.
#Assume the survival time of the phones are normally distributed.
#If a phone is randomly selected to be tracked for repairs find the expected number that ne
#a) after 11,000 hours
cat("Expected NUmber of Phones to be repaired = ",
    pnorm(11000,mean=10000,sd=500,lower.tail=FALSE))
```

**Output:**

```
> cat("probabilty of correct responses = ",pbinom(4,size=50,prob=0.2))
probabilty of correct responses =  0.01849602
```

**Explanation:**

- **pnorm** is the **R** function that calculates the c. d. f. where X is normal. Optional arguments described on the on-line documentation specify the parameters of the particular normal distribution.

**Date: 11/11/2020**

**Q.NO 48: Question**

**Program:**

Load the Boston Housing dataset from the mlbench library and inspect the different types of variables present.

```
library(mlbench)
library(ggplot2)
data("BostonHousing")
housing <- BostonHousing
str(housing)
```

**Output:**

```
> housing <- BostonHousing
> str(housing)
'data.frame':   506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ chas   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ..
 $ rm     : num  6.58 6.42 7.18 7 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : num  1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ b      : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

**Date: 11/11/2020**

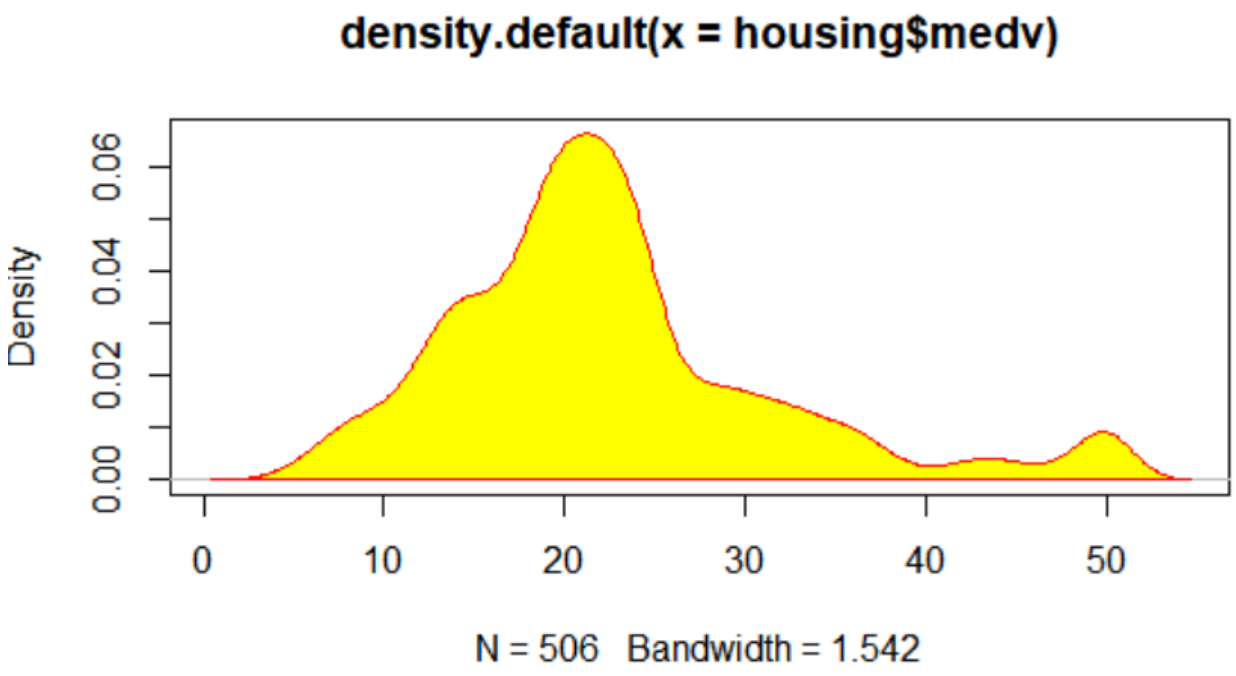
**Q.NO 49: Question**

**Program:**

Explore and visualize the distribution of our target variable.

```
data("BostonHousing")
housing <- BostonHousing
plot(density(housing$medv))
polygon(density(housing$medv),col = "yellow",border = "red")
```

**Output:**



**Date: 11/11/2020**

**Q.NO 50: Question**

**Program:**

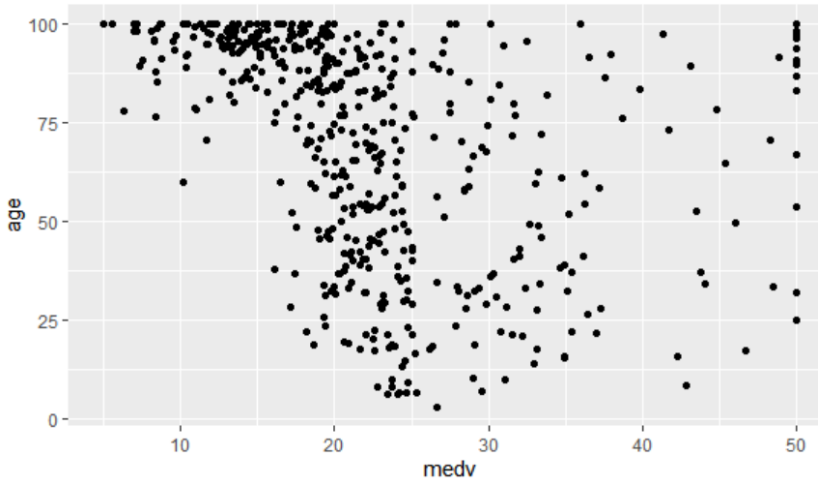
Explore and visualize any potential correlations between medv and the variables crim, rm, age, rad, tax and lstat.

```
#A company ships 5000 cell phones.
#They are expected to last an average of 10,000 hours before needing repair;
#with a standard deviation of 500 hours.
#Assume the survival time of the phones are normally distributed.
#If a phone is randomly selected to be tracked for repairs find the expected number that ne
#a) after 11,000 hours
cat("Expected NUmber of Phones to be repaired = ",
    pnorm(11000,mean=10000,sd=500,lower.tail=FALSE))
```

**Output:**

```
        Pearson's product-moment correlation

data:  x and housing$lstat
t = -24.528, df = 504, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.7749982 -0.6951959
sample estimates:
       cor
-0.7376627
```



**Date: 11/11/2020**

**Q.NO 51: Question**

**Program:**

Set a seed of 123 and split your data into a train and test set using a 75/25 split. You may find the caret library helpful here.

```
#Package for test-test split
library(mlbench)
library(caTools)
set.seed(123)
df <- BostonHousing
#Splitting for test-train 0.75/0.25
train_test=sample.split(df,SplitRatio = 0.75)
train=subset(df,split=TRUE)
test=subset(df,split=FALSE)

df
```

**Output:**

```
 53  5.28 25.0
 54  8.43 23.4
 55 14.80 18.9
 56  4.81 35.4
 57  5.77 24.7
 58  3.95 31.6
 59  6.86 23.3
 60  9.22 19.6
 61 13.15 18.7
 62 14.44 16.0
 63  6.73 22.2
 64  9.50 25.0
 65  8.05 33.0
 66  4.67 23.5
 67 10.24 19.4
 68  8.10 22.0
```

**Date: 11/11/2020**

**Q.NO 52: Question**

**Program:**

We have seen that crim, rm, tax, and lstat could be good predictors of medv. To get the ball rolling, let us fit a linear model for these terms.

```
#Package for test-test split
library(mlbench)
library(caTools)
set.seed(123)
df <- BostonHousing
#Splitting for test-train 0.75/0.25
train_test=sample.split(df,SplitRatio = 0.75)
train=subset(df,split=TRUE)
test=subset(df,split=FALSE)
linear_regression <- lm(medv ~ crim + rm + tax + lstat, data = train)
```

**Output:**

```
Call:
lm(formula = medv ~ crim + rm + tax + lstat, data = train)

Coefficients:
(Intercept)         crim           rm          tax        lstat
  -1.414928    -0.061579     5.248721    -0.005018    -0.534835
```

**Date: 11/11/2020**

**Q.NO 53: Question**

**Program:**

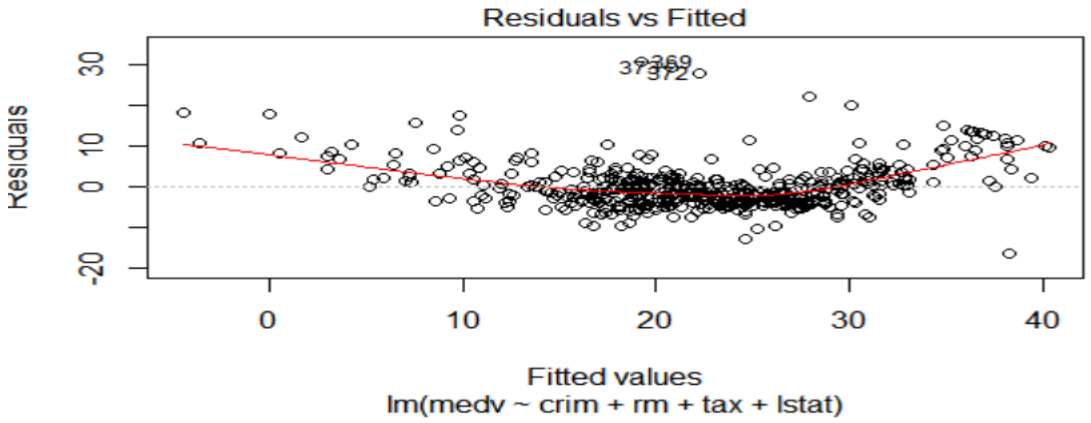
Obtain an r-squared value for your model and examine the diagnostic plots found by plotting your linear model.

```
#Package for test-test split
library(mlbench)
library(caTools)
set.seed(123)
df <- BostonHousing
#Splitting for test-train 0.75/0.25
train_test=sample.split(df,SplitRatio = 0.75)
train=subset(df,split=TRUE)
test=subset(df,split=FALSE)
linear_regression <- lm(medv ~ crim + rm + tax + lstat, data = train)

r_squared <- summary(linear_regression)$r.squared
print(paste("R - Square Value: ",r_squared))
plot(linear_regression)
```

**Output:**

```
> print(paste("R - Square Value: ",r_squared))
[1] "R - Square Value:  0.650605871674497"
```



**Date: 11/11/2020**

**Q.NO 54: Question**

**Program:**

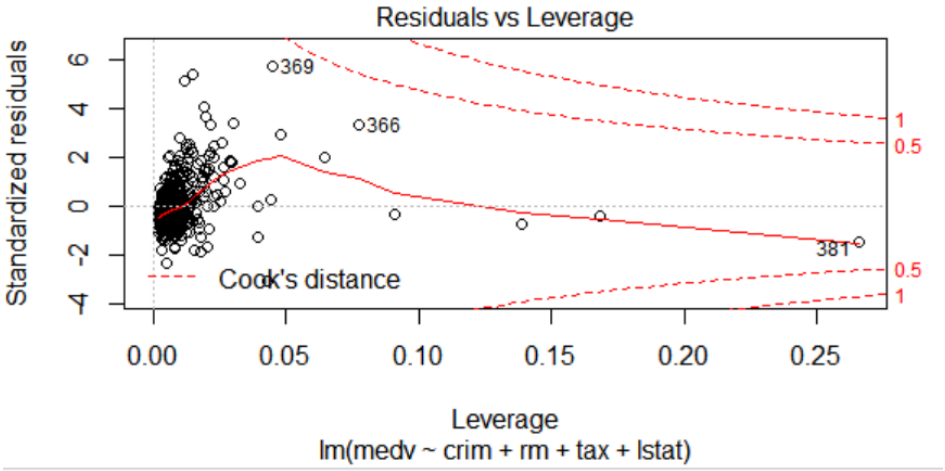
Create a data frame of your predicted values and the original values.

```
#Package for test-test split
library(mlbench)
library(caTools)
set.seed(123)
df <- BostonHousing
#Splitting for test-train 0.75/0.25
train_test=sample.split(df,SplitRatio = 0.75)
train=subset(df,split=TRUE)
test=subset(df,split=FALSE)
linear_regression <- lm(medv ~ crim + rm + tax + lstat, data = train)

r_squared <- summary(linear_regression)$r.squared
print(paste("R - Square Value: ",r_squared))
plot(linear_regression)

predicted <- predict(linear_regression,test)
results <- data.frame(predicted,test$medv)
results
```

**Output:**



**Date: 11/11/2020**

**Q.NO 55: Question**

**Program:**

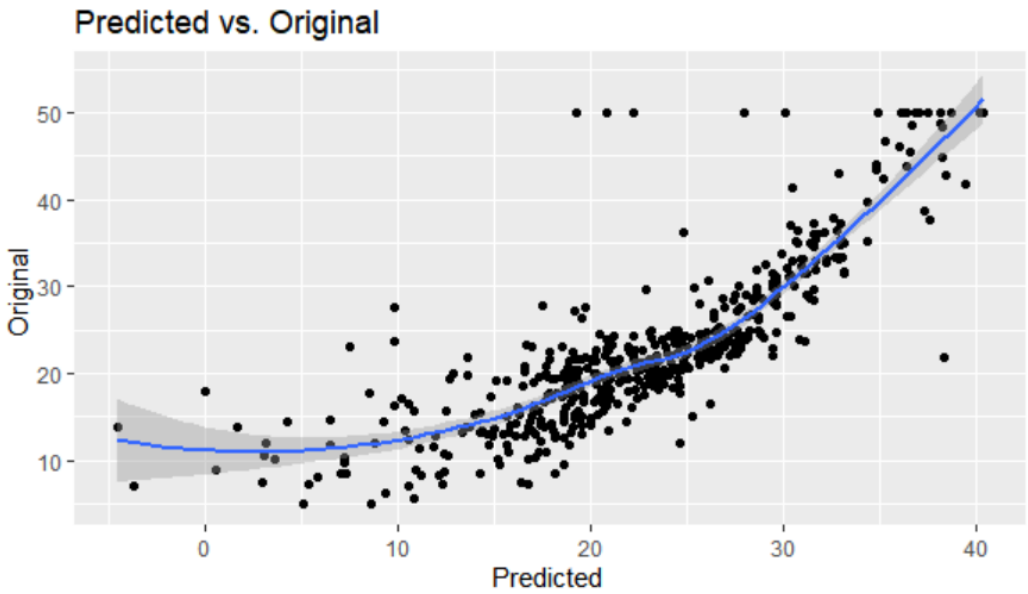
Plot this to visualize the performance of your model.

```
#Package for test-test split
library(mlbench)
library(caTools)
library(ggplot2)
set.seed(123)
df <- BostonHousing
#Splitting for test-train 0.75/0.25
train_test=sample.split(df,SplitRatio = 0.75)
train=subset(df,split=TRUE)
test=subset(df,split=FALSE)
linear_regression <- lm(medv ~ crim + rm + tax + lstat, data = train)

r_squared <- summary(linear_regression)$r.squared
print(paste("R - Square Value: ",r_squared))


predicted <- predict(linear_regression,test)
results <- data.frame(predicted,test$medv)
```

**Output:**



**Date: 12/11/2020**

**Q.NO 56: Question**

**Program:**

Install and load the package googleVis. Create a data frame First of all let's create an experimental data.frame to use for all our plots.

This is an example: dfr=data.frame(name=c("GRE", "ARG", "BRA"), val1=c(20,32,19), val2=c(25,52,12))

```
library(googleVis)
dfr=data.frame(name=c("GRE", "ARG", "BRA"),
               val1=c(20,32,19),
               val2=c(25,52,12))
```

**Output:**

```
> dfr
  name val1 val2
1  GRE   20   25
2  ARG   32   52
3  BRA   19   12
>
```

**Explanation:**

- The **Google Visualization** API provides formatters that can be used to reformat data in a visualization.

**Date: 12/11/2020**

**Q.NO 57: Question**

**Program:**

Create a data frame named “df”. Give as variables the “Pts” (Points) and “Rbs” (Rebounds) of three NBA players. Names and values are up to you.Note:

```
library(googleVis)
df=data.frame(name=c("Sathish", "Karan", "Arun"),
              Pts=c(18,78,56),
              Rbs=c(5,7,9))
df
```

**Output:**

```
> df
     name Pts Rbs
1 Sathish  18   5
2   Karan  78   7
3    Arun  56   9
>
```

**Date: 12/11/2020**

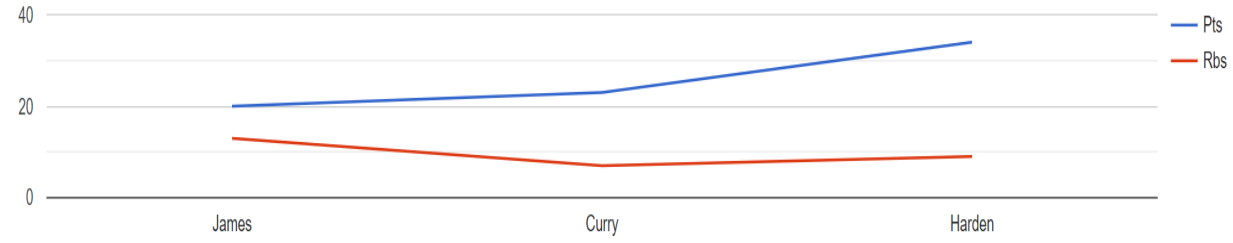
**Q.NO 58: Question**

**Program:**

Line Chart To produce a Line Chart you can use: LineC <- gvisLineChart(df) plot(LineC)

```
library(googleVis)
df=data.frame(name=c("James", "Curry", "Harden"),
              Pts=c(20,23,34),
              Rbs=c(13,7,9))
Line_1<- gvisLineChart(df)
plot(Line_1)
```

**Output:**




**Explanation:**

- **pnorm** is the **R** function that calculates the c. d. f. where X is normal. Optional arguments described on the on-line documentation specify the parameters of the particular normal distribution.

**Date: 12/11/2020**

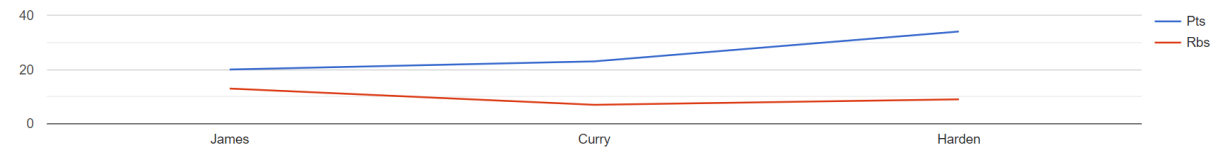
**Q.NO 59: Question**

**Program:**

Create a single axis Line chart that displays only the “Pts” of the “df” data frame.

```
library(googleVis)
df=data.frame(name=c("James", "Curry", "Harden"),
              Pts=c(20,23,34),
              Rbs=c(13,7,9))
Line_2 <- gvisLineChart(df)
plot(Line_2)
```

**Output:**




**x**

**Date: 12/11/2020**

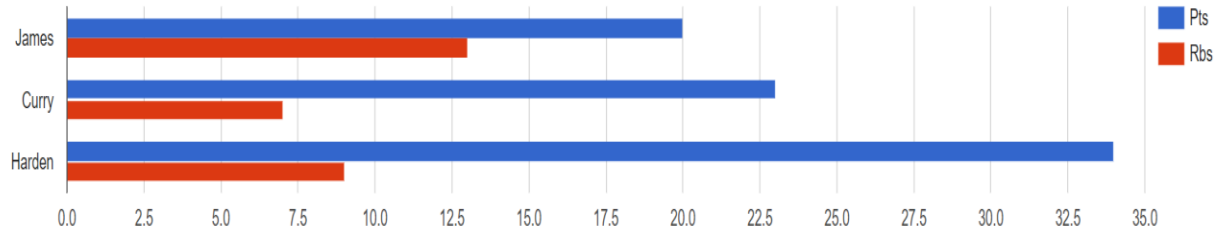
**Q.NO 60: Question**

**Program:**

create a two axis line chart that displays both “Pts” and “Rbs” of the “df” data frame. HINT: Use list().

```
library(googleVis)
df=data.frame(name=c("James", "Curry", "Harden"),
              Pts=c(20,23,34),
              Rbs=c(13,7,9))
BarChart <- gvisBarChart(df)
plot(BarChart)
```

**Output:**




# THANK YOU