Experiment no:1

Aim: Installation of R and R studio

Theory:

R:

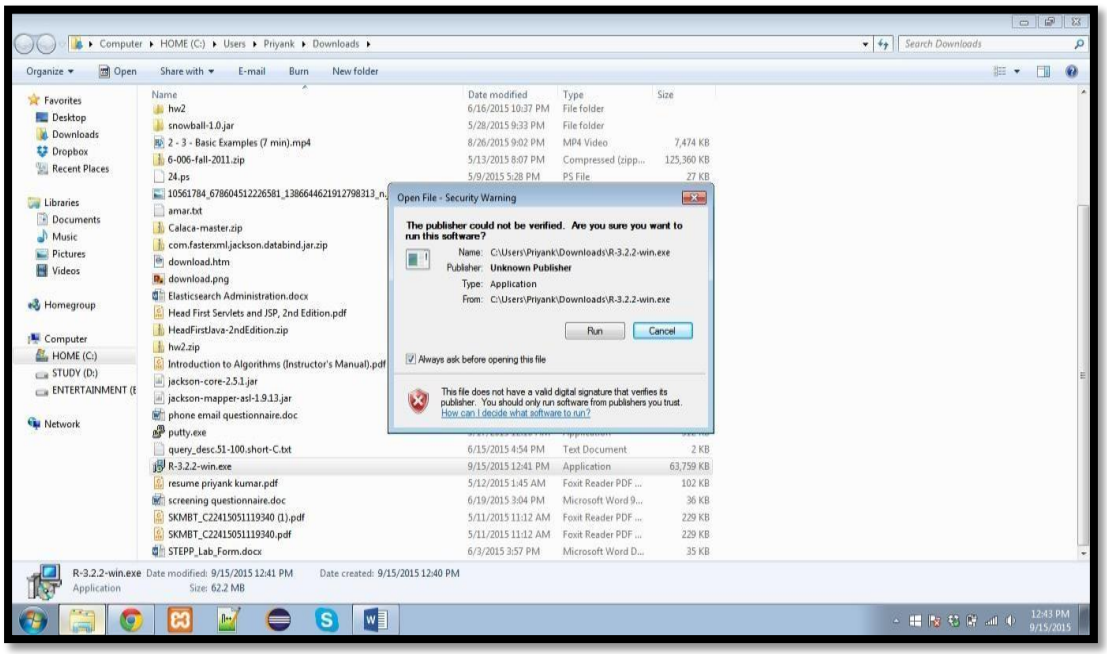
R is a free, open source software program for statistical analysis, based on the S language.

R-STUDIO:

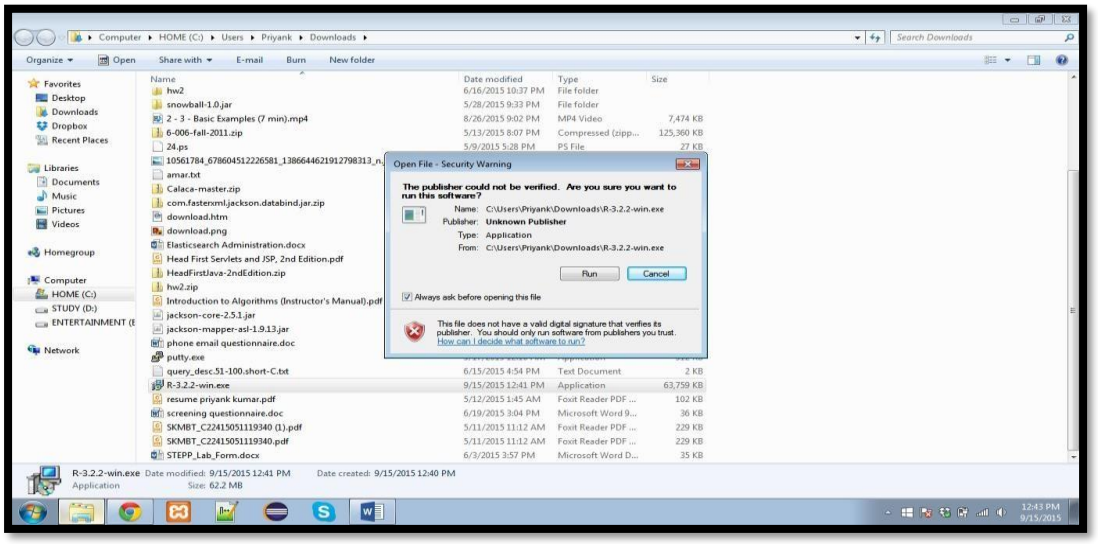
RStudio is a free, open source IDE (integrated development environment) for R. Its interface is organized so that the user can clearly view graphs, data tables, R code, and output all at the same time. It also offers an Import-Wizard-like feature that allows users to import CSV, Excel, SAS (*. sas7bdat), SPSS (*.sav), and Stata (*.dta) files into R without having to write the code to do so.

INSTALLATION OF R AND R-STUDIO:

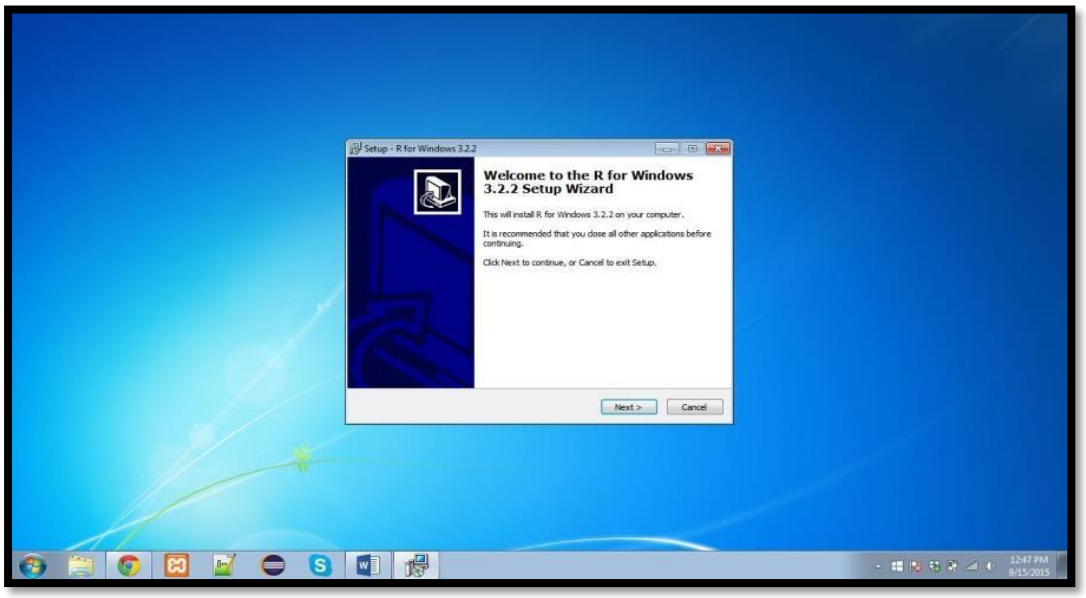
1. Download the installable file from the following link: https://cran.r-project.org/bin/windows/base/
2. Click on the R 3.2.2.exe file. The 3.2.2 is the version number of the file. The versions can be updated as per the latest releases.



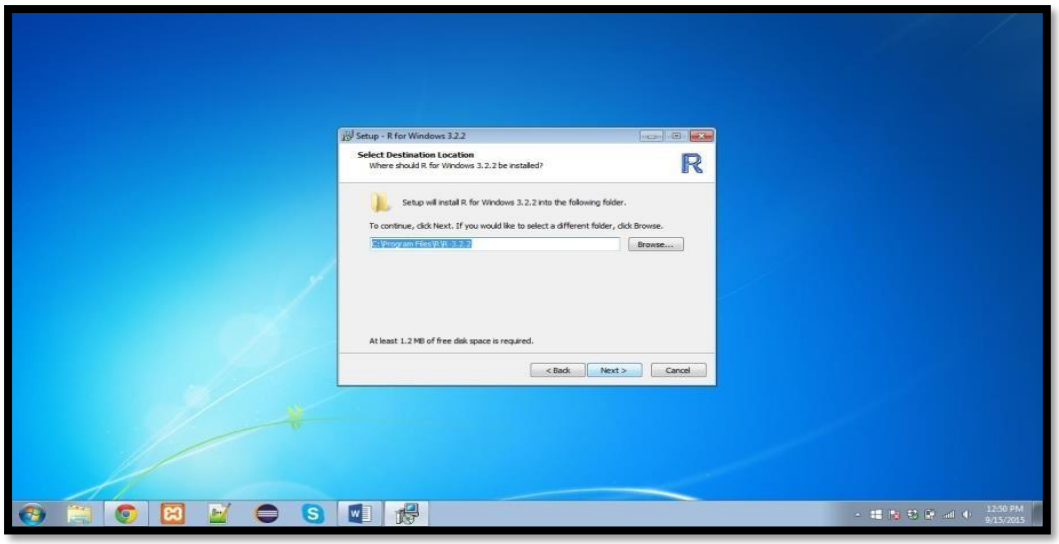
3. The Setup will request permission to be installed on the system click yes to proceed.



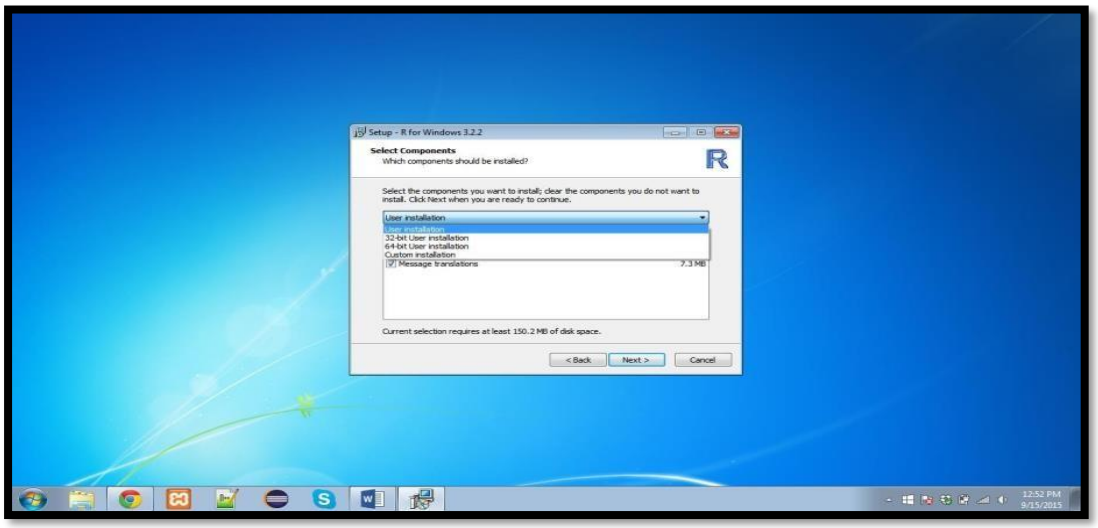
4. Select the Preferred language from the dropdown to begin an installation in that preferred language.
5. Click next to proceed with the installation.



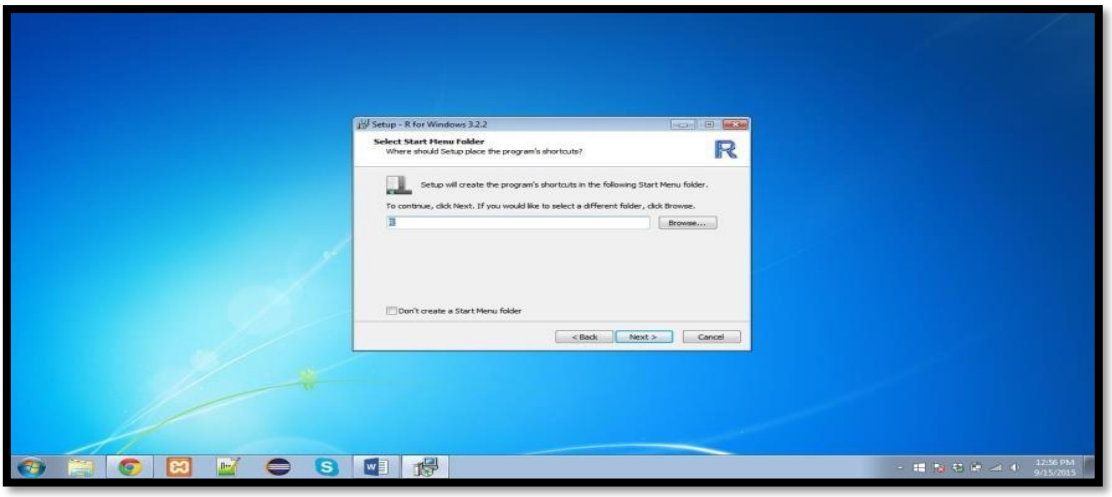
6. Choose the path where you wish to install R by clicking on browse and changing the workspace locations. Click next to proceed with the default installation. The minimum space requirements are mentioned at the bottom of the dialog box. Please check you have required amount of free space in your drive.



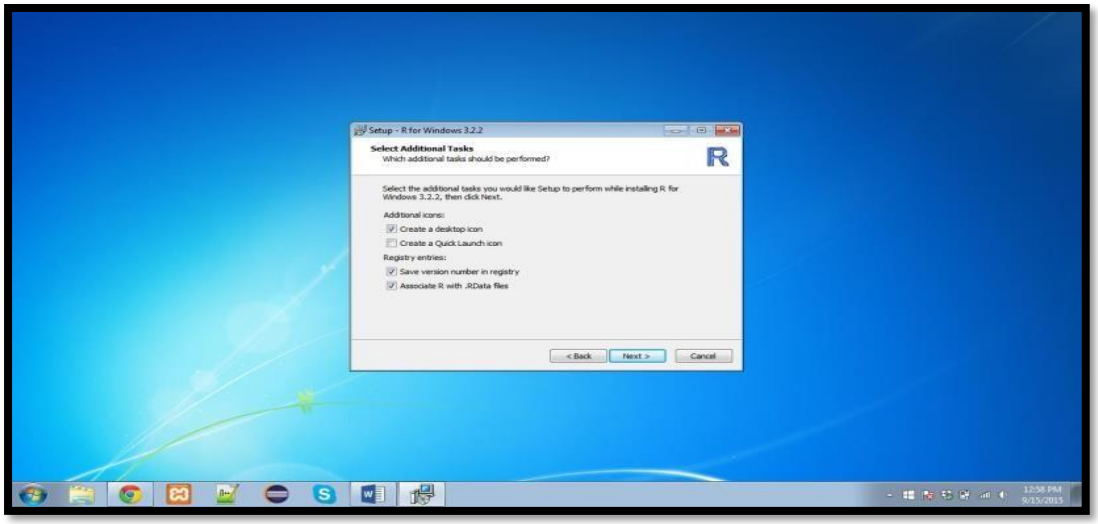
7. Choose the type of installation you require. By default, R installs both the 32- and 64-bit versions on your system. If your system is a 32-bit system you will be requiring a 32-bit installation if the system is a 64-bit system it will be requiring 64-bit installation. Do not uncheck the Core Files and Message Translations. Please make note of the space requirement of the installation.



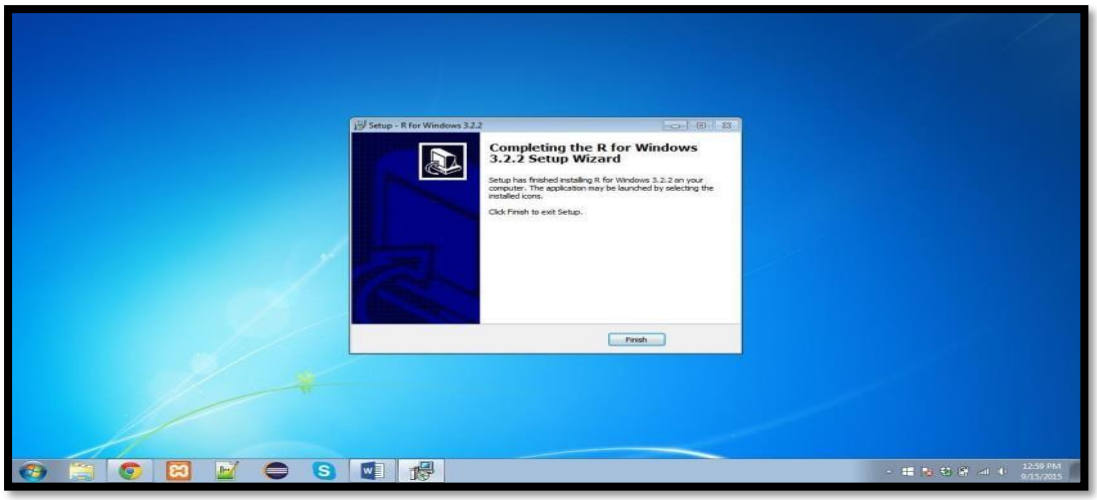
8. To customize the start-up options for R, choose option and customize. To proceed with a vanilla installation, use Next.
9. To generate program shortcuts and naming those as per your requirement specify the necessary customizations. To proceed with the default installation hit next.



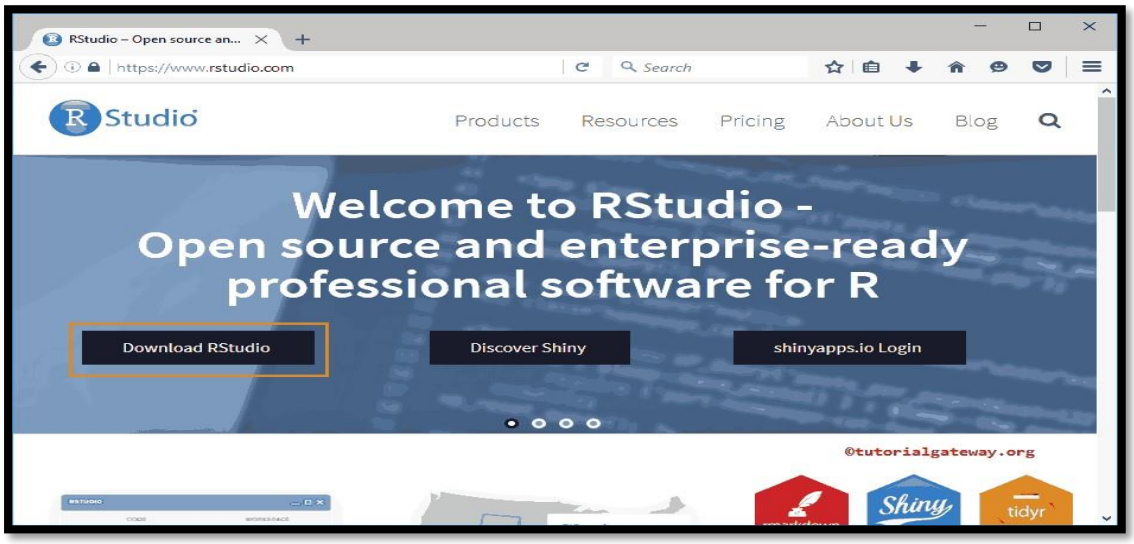
10. Click on the next button to begin your installation.



11. After the installation has completed you will see the final screen. Click finish to complete the installation.

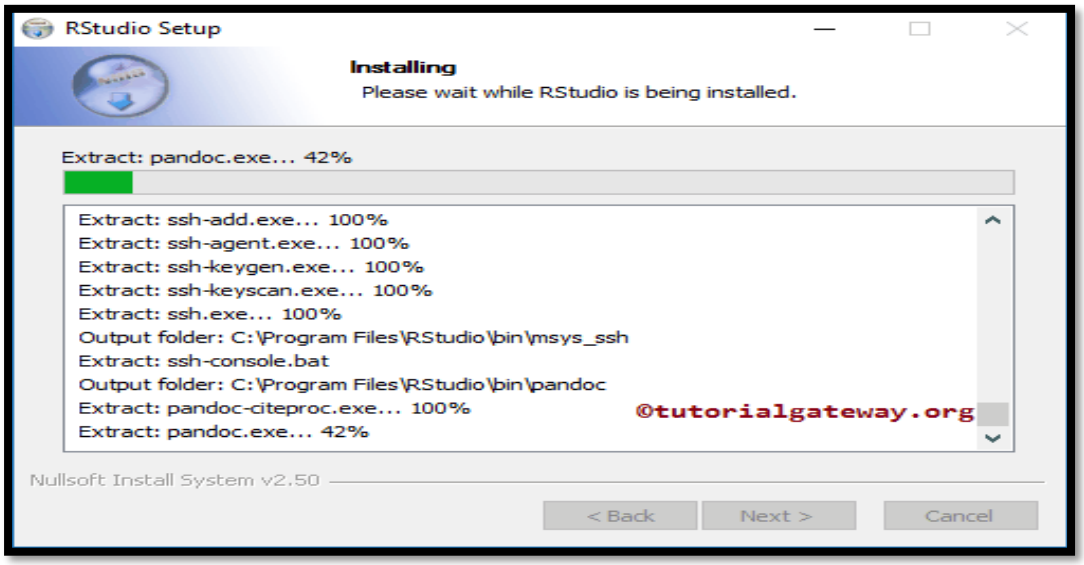


12. Open Start Menu and you will find R in the available set of Programs.
13. Click on the R icon in the menu settings to open R.
14. To download R Studio, first Go to the official R Studio website by clicking this link Download.

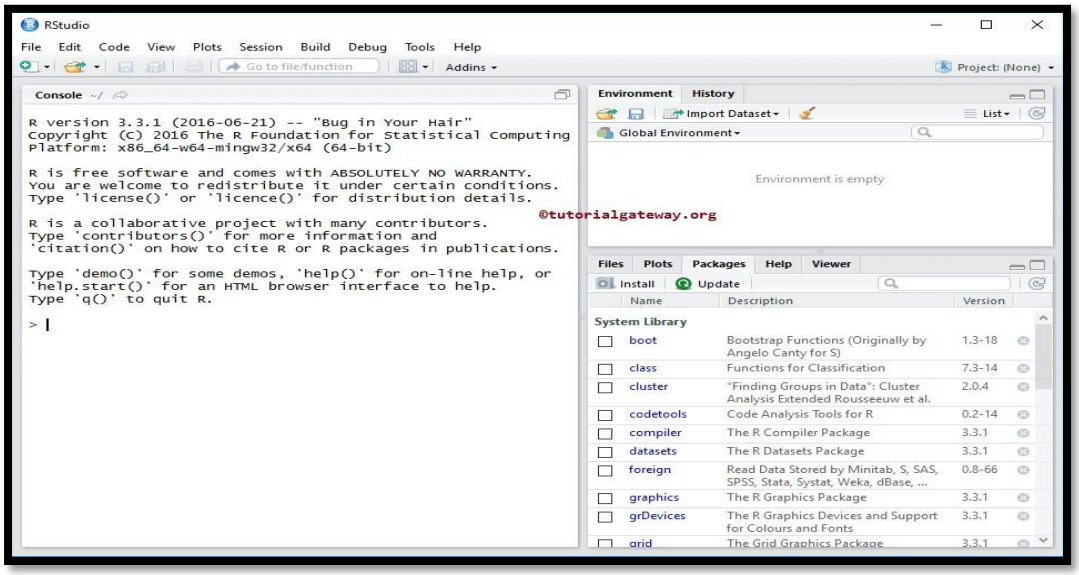


15. Once you click on the specified download RStudio button, the following tab or window opened. Please select the R Studio Desktop (Free License) and click the download button.
16. Or scroll down the page and select the RStudio for your Operating System. From the below screenshot, see that we are selecting the RStudio 0.99.903 – Windows Vista/7/8/10
17. Once you Click on the RStudio hyperlink, a pop-up window opened to save this file. Please select the Save or download R Studio File option.
18. After you click on the install R Studio application, Welcome to the RStudio Setup Wizard window opened. Please select the Next button.

19. Here, you can change the default R Studio installation directory. To do that, we have to click on the Browse… button beside the text box. It opens our file system to navigate the location and select the directory.
20. Here, you can change the R Studio program's shortcuts default directory. For this, we have to select the directory name from the list.
21. Please wait for the extraction and installation of R studio completed. Next, Click the Finish button for completing the Installation of R Studio.



22. Let us open the R Studio to see the R Software environment.



Conclusion: Thus, we have successfully installed R and R studio.

Experiment no:2

Aim: Basic functionality in R-variables, basic arithmetic, help command

Theory:

Variables in R:

A variable is a name given to a memory location, which is used to store values in a computer program. Variables in R programming can be used to store numbers (real and complex), words, matrices, and even tables. R is a dynamically programmed language which means that unlike other programming languages, we do not have to declare the data type of a variable before we can use it in our program. For a variable to be valid, it should follow these rules:

1. It should contain letters, numbers, and only dot or underscore characters.
2. It should not start with a number (e.g.: - 2iota)
3. It should not start with a dot followed by a number (e.g.: - .2iota)
4. It should not start with an underscore (e.g.: - _iota)
5. It should not be a reserved keyword

Constants in R:

The entities whose values are fixed are called constants. There are two types of constants in R:

- **Numeric Constants:** All numeric values such as integer, double, or complex fall under this category. Numeric constants followed by 'L' and 'i' are considered as integer and complex respectively. And, numeric constants preceded by 0x/0X are treated as hexadecimal numbers.
- **Character Constants:** These constants are represented by single (') or double (") quotes called delimiters.

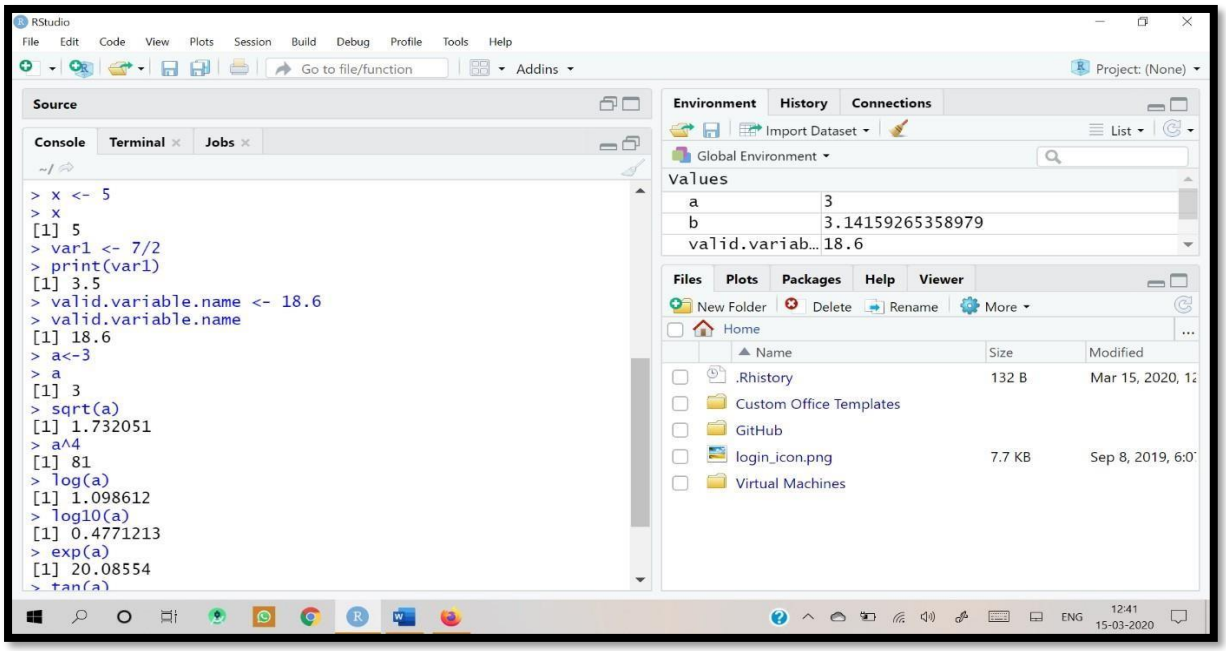
Basic Arithmetic in R:

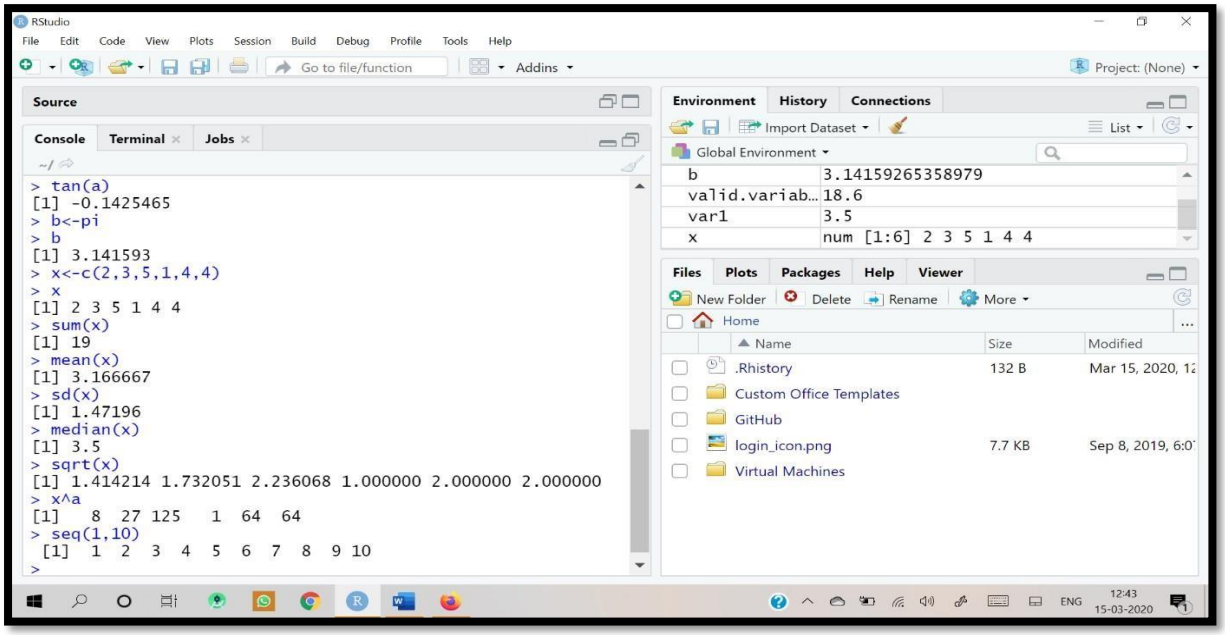
The R Arithmetic operators include operators like Arithmetic Addition, Subtraction, Division, Multiplication, Exponent, Integer Division, and Modulus. All these R arithmetic operators are binary operators, which means they operate on two operands.

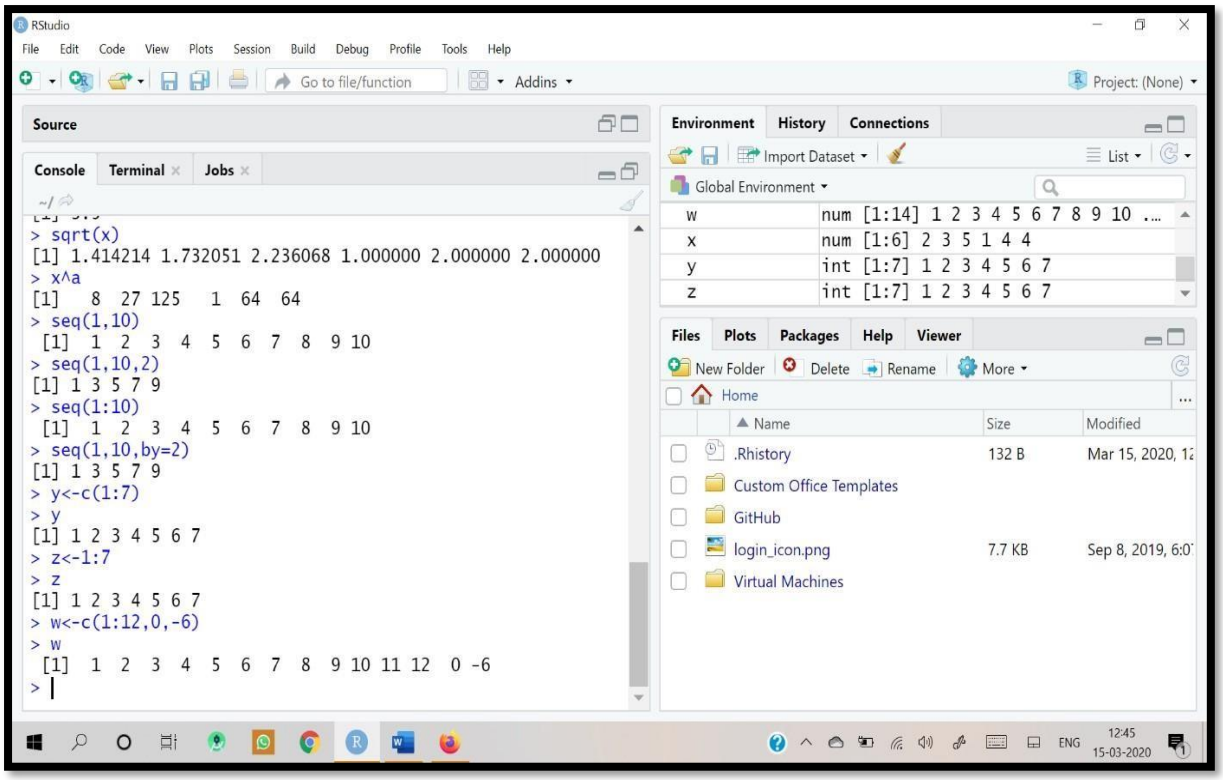
Help command:

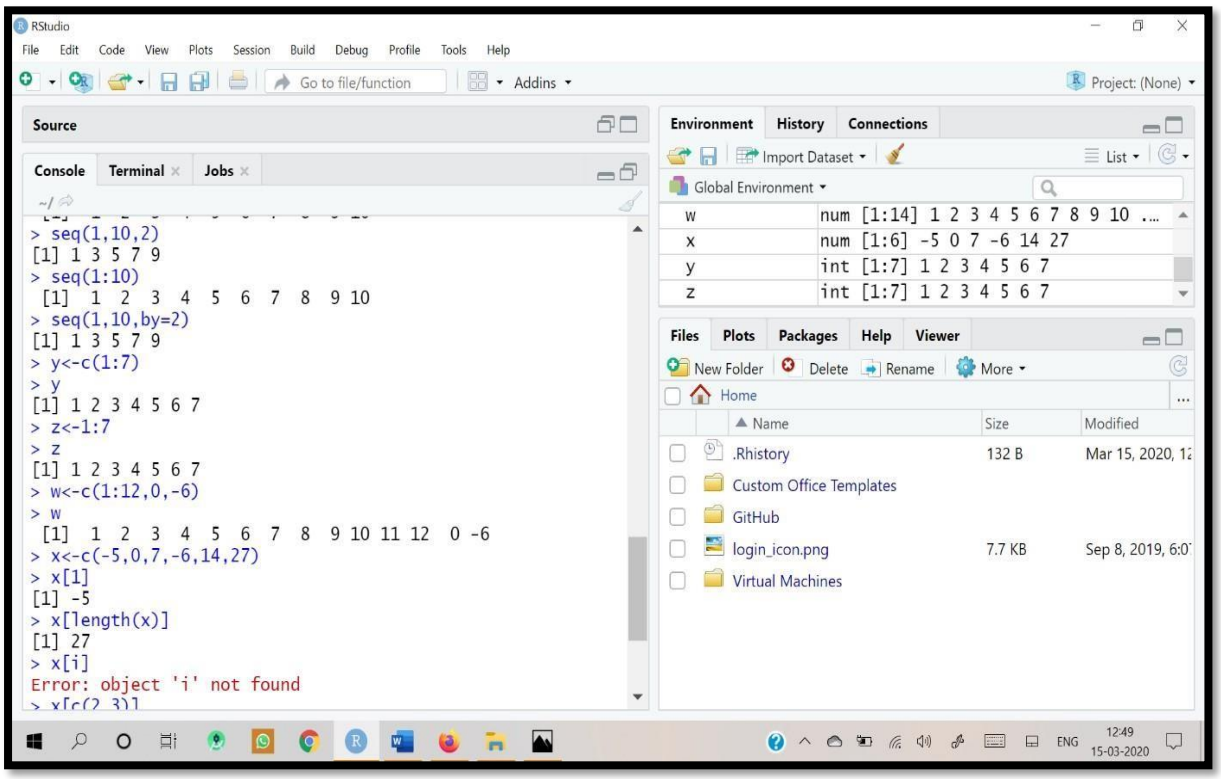
The help() function help in R to provide access to the documentation pages for R functions, data sets, and other objects, both for packages in the standard R distribution and for contributed packages.

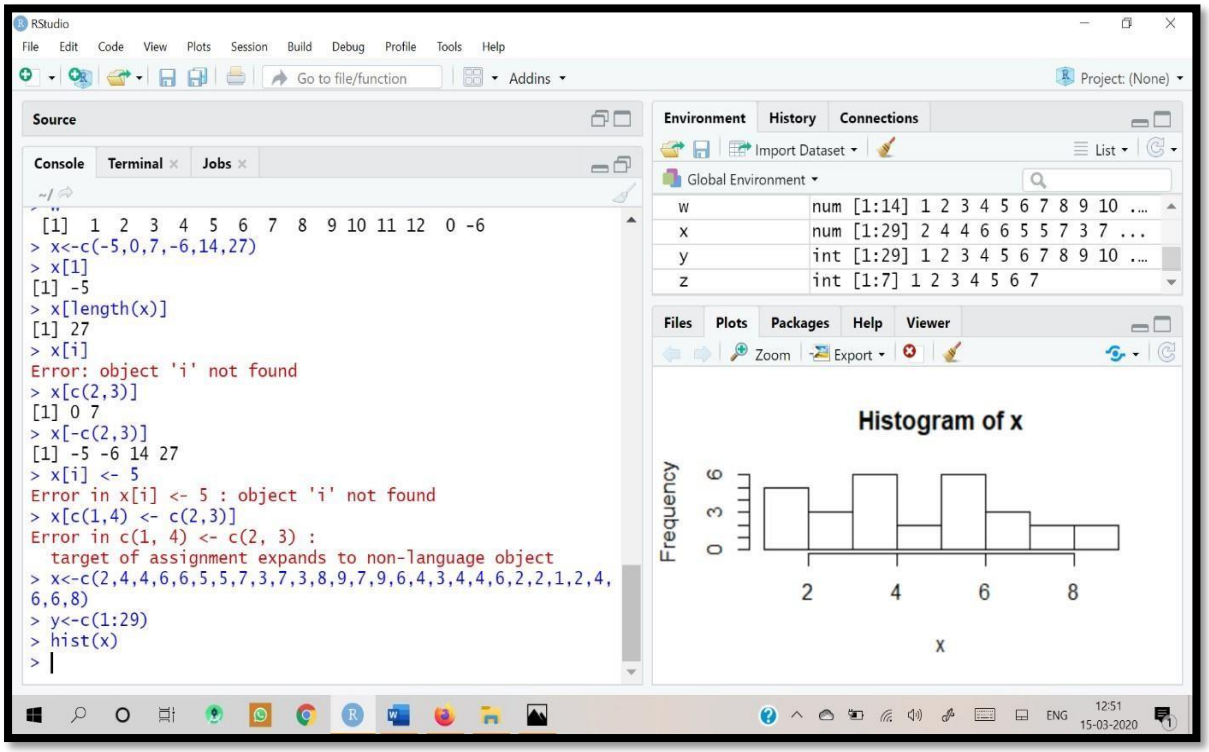
Output:

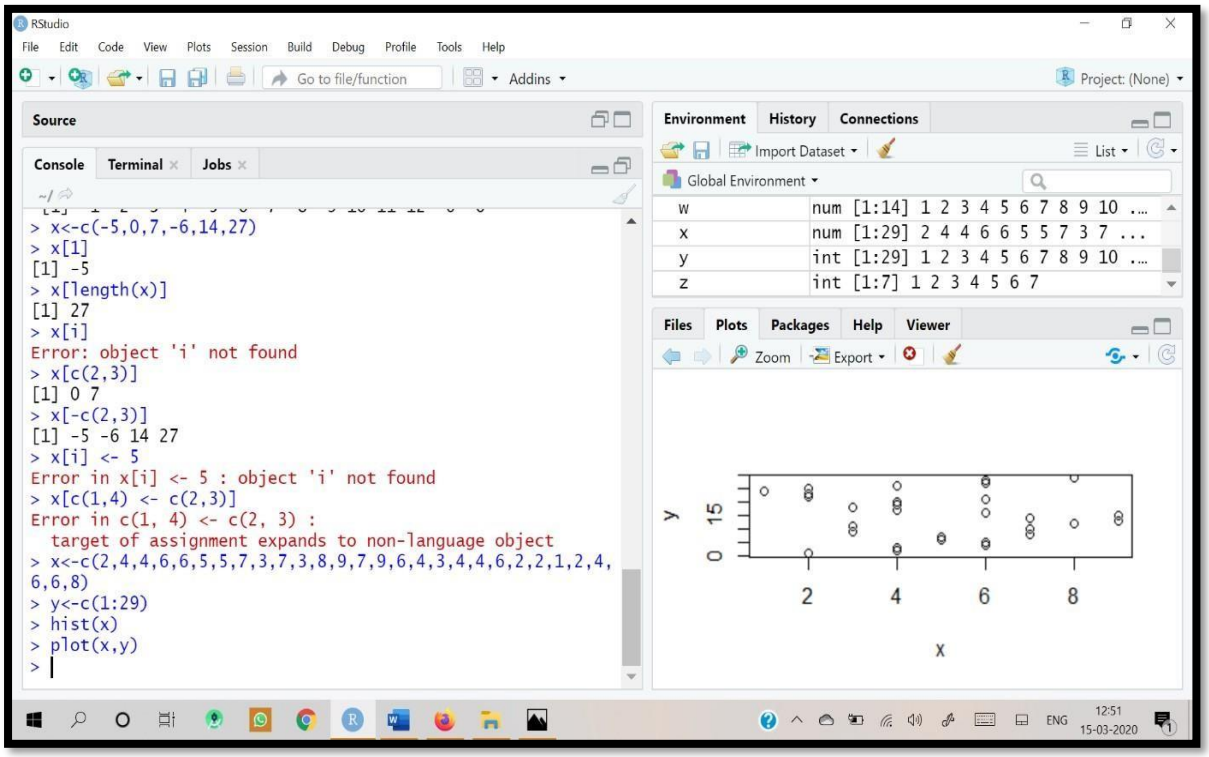


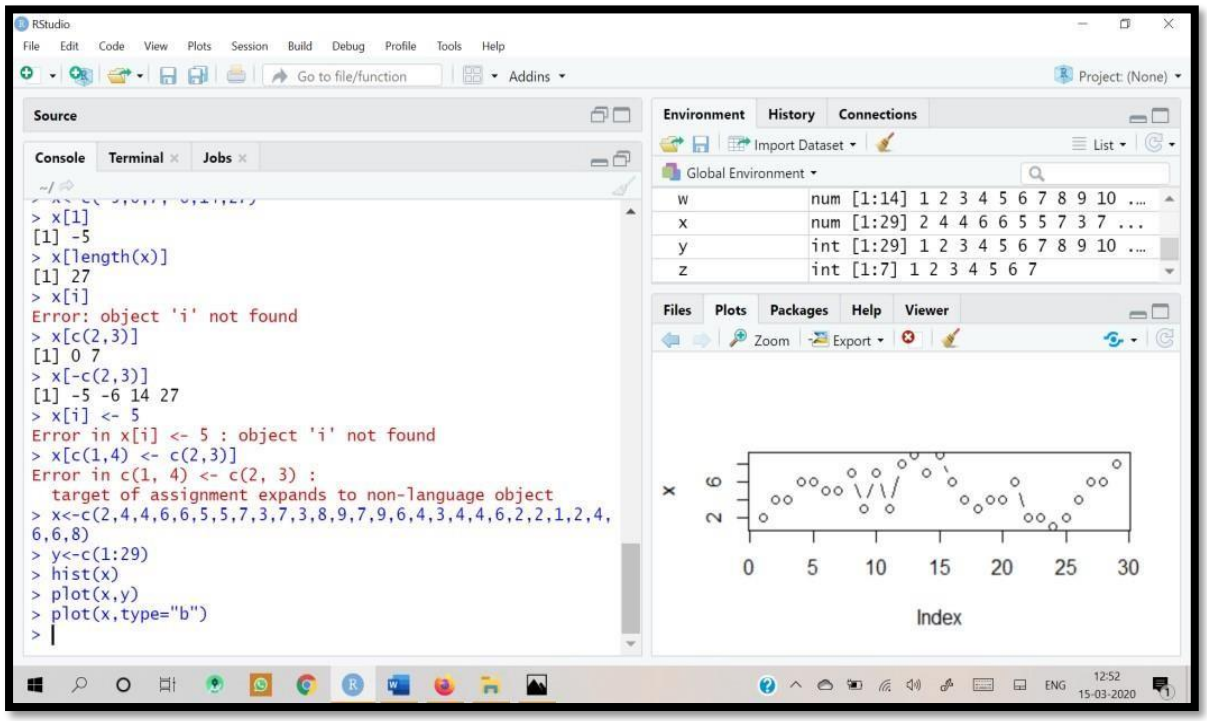


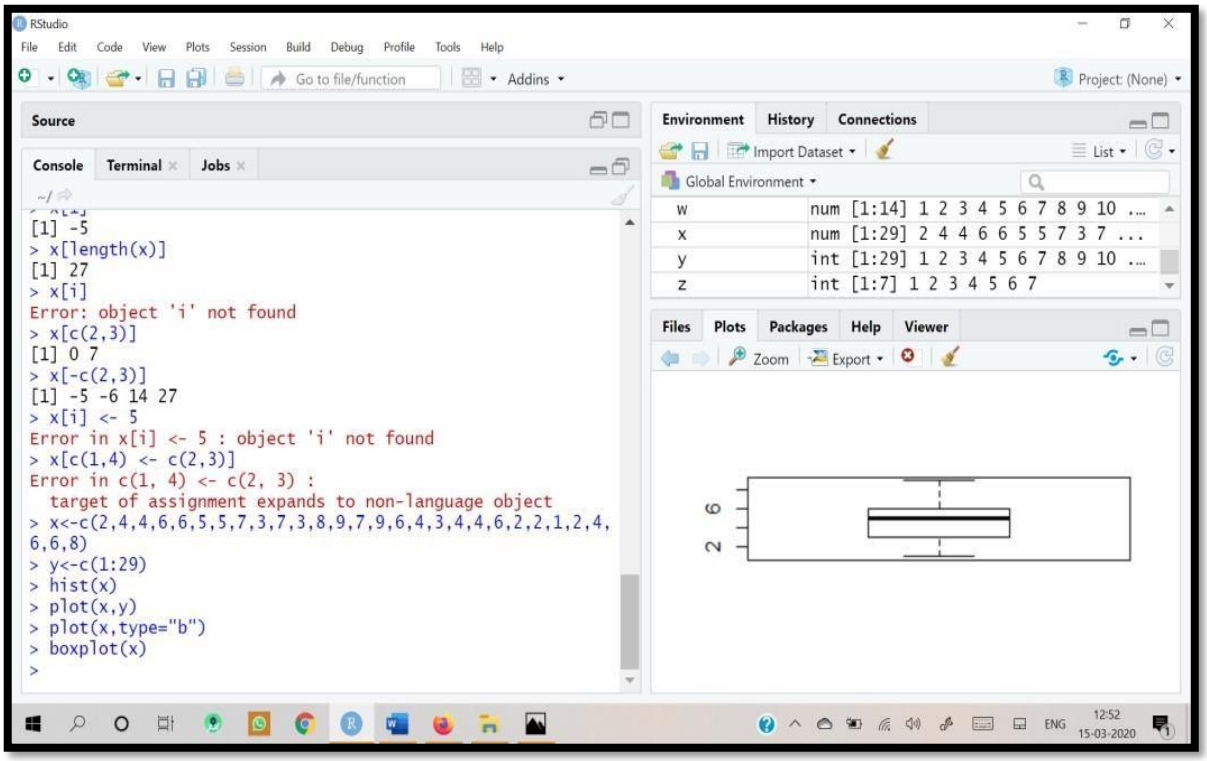

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Source
Console Terminal Jobs
> sqrt(x)
[1] 1.414214 1.732051 2.236068 1.000000 2.000000 2.000000
> x^a
[1] 8 27 125 1 64 64
> seq(1,10)
[1] 1 2 3 4 5 6 7 8 9 10
> seq(1,10,2)
[1] 1 3 5 7 9
> seq(1:10)
[1] 1 2 3 4 5 6 7 8 9 10
> seq(1,10,by=2)
[1] 1 3 5 7 9
> y<-c(1:7)
> y
[1] 1 2 3 4 5 6 7
> z<-1:7
> z
[1] 1 2 3 4 5 6 7
> w<-c(1:12,0,-6)
> w
[1] 1 2 3 4 5 6 7 8 9 10 11 12 0 -6
Environment History Connections
Import Dataset
List
Global Environment
w num [1:14] 1 2 3 4 5 6 7 8 9 10 ....
x num [1:6] 2 3 5 1 4 4
y int [1:7] 1 2 3 4 5 6 7
z int [1:7] 1 2 3 4 5 6 7
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home
Name Size Modified
.Rhistory 132 B Mar 15, 2020, 1
Custom Office Templates
GitHub
login_icon.png 7.7 KB Sep 8, 2019, 6:0
Virtual Machines
ENG 12:45 15-03-2020

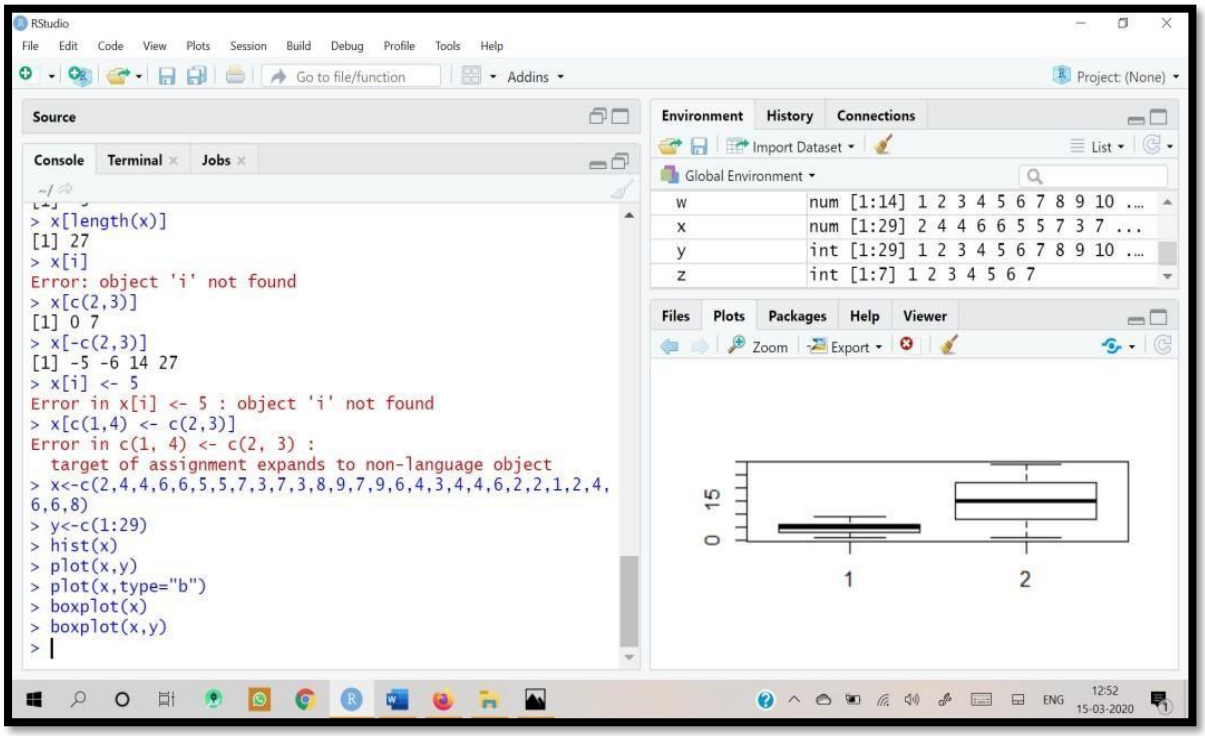

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Source
Console Terminal Jobs
> seq(1,10,2)
[1] 1 3 5 7 9
> seq(1:10)
[1] 1 2 3 4 5 6 7 8 9 10
> seq(1,10,by=2)
[1] 1 3 5 7 9
> y<-c(1:7)
> y
[1] 1 2 3 4 5 6 7
> z<-1:7
> z
[1] 1 2 3 4 5 6 7
> w<-c(1:12,0,-6)
> w
[1] 1 2 3 4 5 6 7 8 9 10 11 12 0 -6
> x<-c(-5,0,7,-6,14,27)
> x[1]
[1] -5
> x[length(x)]
[1] 27
> x[i]
Error: object 'i' not found
Environment History Connections
Import Dataset
List
Global Environment
w num [1:14] 1 2 3 4 5 6 7 8 9 10 ....
x num [1:6] -5 0 7 -6 14 27
y int [1:7] 1 2 3 4 5 6 7
z int [1:7] 1 2 3 4 5 6 7
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home
Name Size Modified
.Rhistory 132 B Mar 15, 2020, 1
Custom Office Templates
GitHub
login_icon.png 7.7 KB Sep 8, 2019, 6:0
Virtual Machines
ENG 12:49 15-03-2020

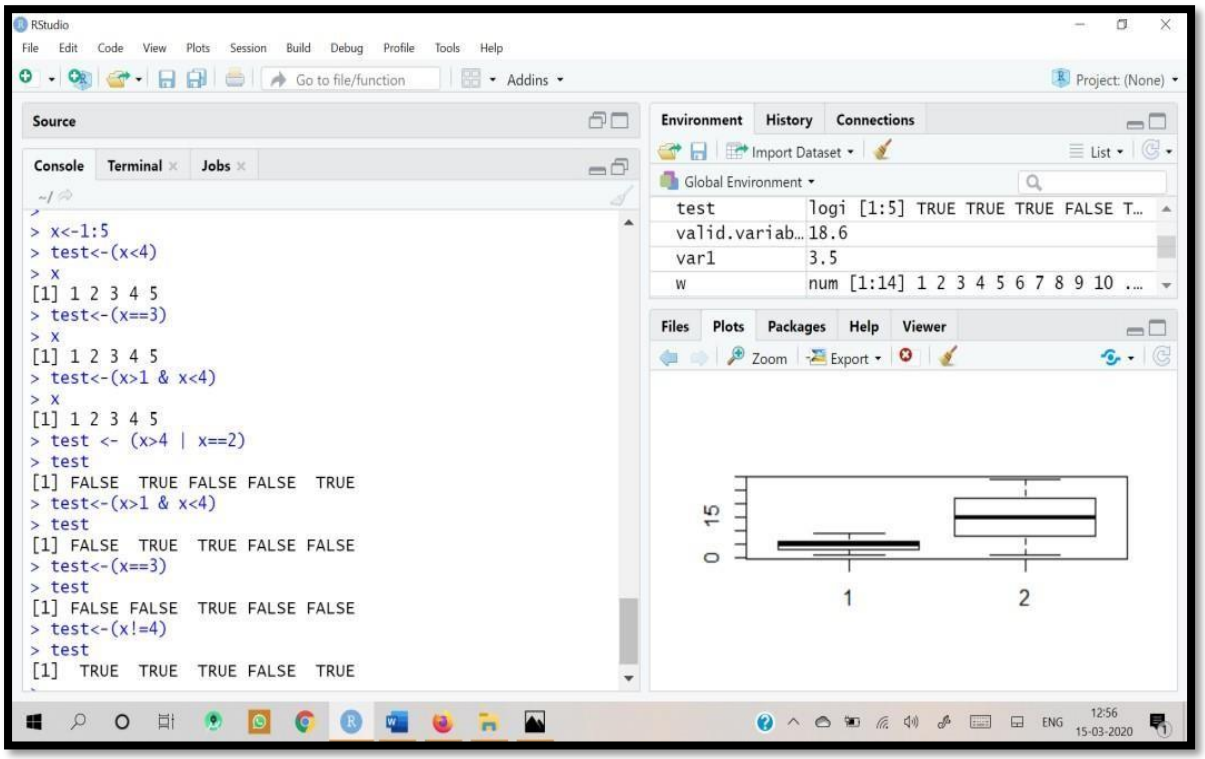

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Source
Console Terminal Jobs
[1] 1 2 3 4 5 6 7 8 9 10 11 12 0 -6
> x<-c(-5,0,7,-6,14,27)
> x[1]
[1] -5
> x[length(x)]
[1] 27
> x[i]
Error: object 'i' not found
> x[c(2,3)]
[1] 0 7
> x[-c(2,3)]
[1] -5 -6 14 27
> x[i] <- 5
Error in x[i] <- 5 : object 'i' not found
> x[c(1,4) <- c(2,3)]
Error in c(1, 4) <- c(2, 3) :
target of assignment expands to non-language object
> x<-c(2,4,4,6,6,5,5,7,3,7,3,8,9,7,9,6,4,3,4,4,6,2,2,1,2,4,6,6,8)
> y<-c(1:29)
> hist(x)
Environment History Connections
Import Dataset
List
Global Environment
w num [1:14] 1 2 3 4 5 6 7 8 9 10 ....
x num [1:29] 2 4 4 6 6 5 5 7 3 7 ...
y int [1:29] 1 2 3 4 5 6 7 8 9 10 ....
z int [1:7] 1 2 3 4 5 6 7
Files Plots Packages Help Viewer
Zoom
Export
Histogram of x
Frequency
0 3 6
2 4 6 8
x
ENG
12:51
15-03-2020

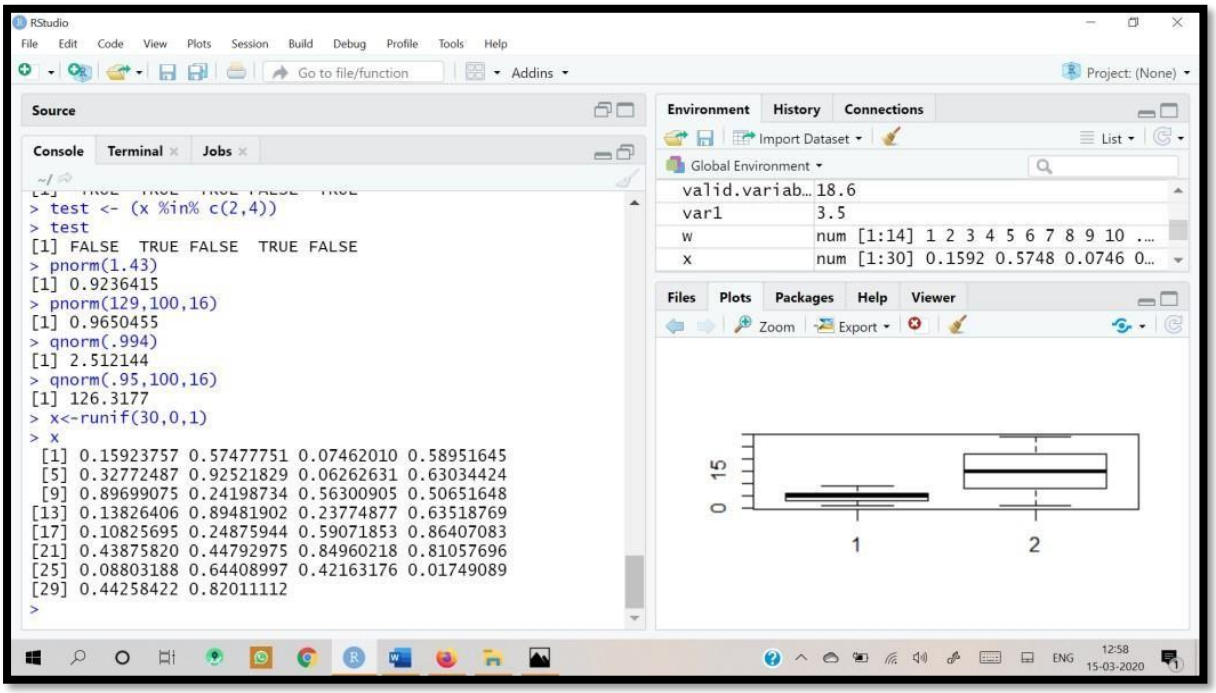

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Source
Console Terminal Jobs
> x<-c(-5,0,7,-6,14,27)
> x[1]
[1] -5
> x[length(x)]
[1] 27
> x[i]
Error: object 'i' not found
> x[c(2,3)]
[1] 0 7
> x[-c(2,3)]
[1] -5 -6 14 27
> x[i] <- 5
Error in x[i] <- 5 : object 'i' not found
> x[c(1,4) <- c(2,3)]
Error in c(1, 4) <- c(2, 3) :
target of assignment expands to non-language object
> x<-c(2,4,4,6,6,5,5,7,3,7,3,8,9,7,9,6,4,3,4,4,6,2,2,1,2,4,6,6,8)
> y<-c(1:29)
> hist(x)
> plot(x,y)
Environment History Connections
Import Dataset
List
Global Environment
w num [1:14] 1 2 3 4 5 6 7 8 9 10 ....
x num [1:29] 2 4 4 6 6 5 5 7 3 7 ...
y int [1:29] 1 2 3 4 5 6 7 8 9 10 ....
z int [1:7] 1 2 3 4 5 6 7
Files Plots Packages Help Viewer
Zoom
Export
y
0 15
2 4 6 8
x
ENG
12:51
15-03-2020

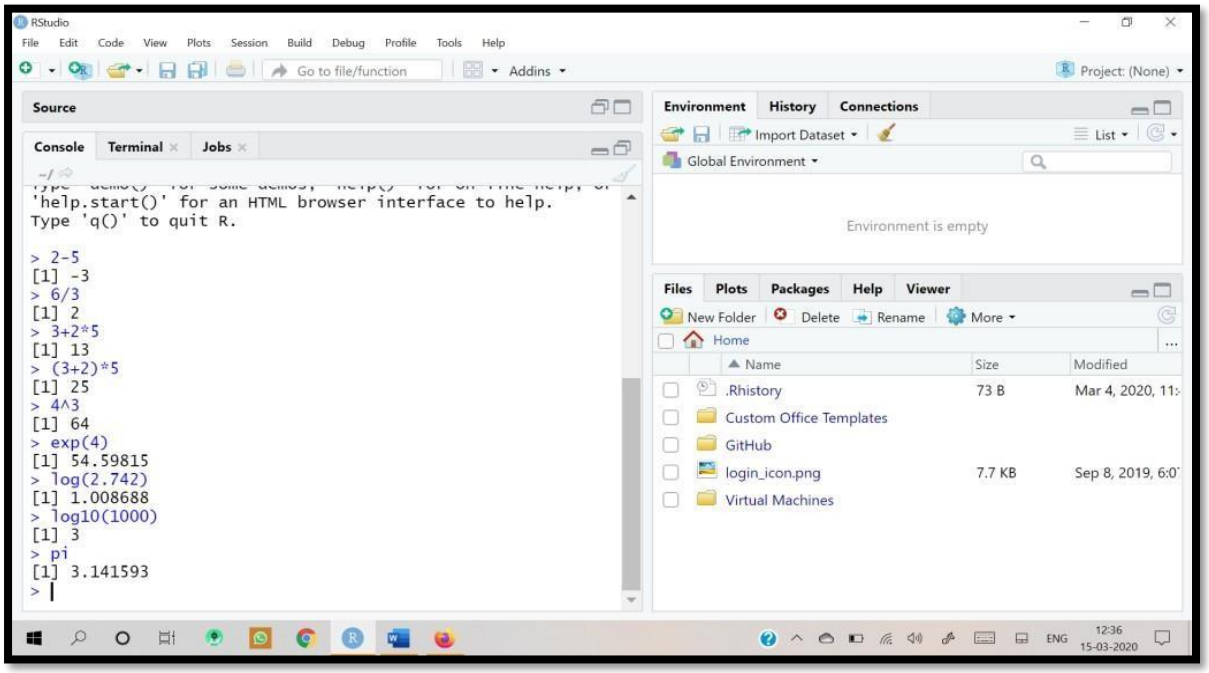

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Source
Console Terminal Jobs
~/
> x[1]
[1] -5
> x[length(x)]
[1] 27
> x[i]
Error: object 'i' not found
> x[c(2,3)]
[1] 0 7
> x[-c(2,3)]
[1] -5 -6 14 27
> x[i] <- 5
Error in x[i] <- 5 : object 'i' not found
> x[c(1,4) <- c(2,3)]
Error in c(1, 4) <- c(2, 3) :
target of assignment expands to non-language object
> x<-c(2,4,4,6,6,5,5,7,3,7,3,8,9,7,9,6,4,3,4,4,6,2,2,1,2,4,
6,6,8)
> y<-c(1:29)
> hist(x)
> plot(x,y)
> plot(x,type="b")
>
Environment History Connections
Import Dataset
List
Global Environment
w num [1:14] 1 2 3 4 5 6 7 8 9 10 ....
x num [1:29] 2 4 4 6 6 5 5 7 3 7 ...
y int [1:29] 1 2 3 4 5 6 7 8 9 10 ....
z int [1:7] 1 2 3 4 5 6 7
Files Plots Packages Help Viewer
Zoom
Export
x
2
6
0
5
10
15
20
25
30
Index
ENG
12:52
15-03-2020


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Source
Console Terminal Jobs
~/
[1] -5
> x[length(x)]
[1] 27
> x[i]
Error: object 'i' not found
> x[c(2,3)]
[1] 0 7
> x[-c(2,3)]
[1] -5 -6 14 27
> x[i] <- 5
Error in x[i] <- 5 : object 'i' not found
> x[c(1,4) <- c(2,3)]
Error in c(1, 4) <- c(2, 3) :
target of assignment expands to non-language object
> x<-c(2,4,4,6,6,5,5,7,3,7,3,8,9,7,9,6,4,3,4,4,6,2,2,1,2,4,
6,6,8)
> y<-c(1:29)
> hist(x)
> plot(x,y)
> plot(x,type="b")
> boxplot(x)
>
Environment History Connections
Import Dataset
List
Global Environment
w num [1:14] 1 2 3 4 5 6 7 8 9 10 ....
x num [1:29] 2 4 4 6 6 5 5 7 3 7 ...
y int [1:29] 1 2 3 4 5 6 7 8 9 10 ....
z int [1:7] 1 2 3 4 5 6 7
Files Plots Packages Help Viewer
Zoom
Export
2
6
ENG
12:52
15-03-2020


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Source
Console Terminal Jobs
> x[length(x)]
[1] 27
> x[i]
Error: object 'i' not found
> x[c(2,3)]
[1] 0 7
> x[-c(2,3)]
[1] -5 -6 14 27
> x[i] <- 5
Error in x[i] <- 5 : object 'i' not found
> x[c(1,4) <- c(2,3)]
Error in c(1, 4) <- c(2, 3) :
target of assignment expands to non-language object
> x<-c(2,4,4,6,6,5,5,7,3,7,3,8,9,7,9,6,4,3,4,4,6,2,2,1,2,4,
6,6,8)
> y<-c(1:29)
> hist(x)
> plot(x,y)
> plot(x,type="b")
> boxplot(x)
> boxplot(x,y)
Environment History Connections
Import Dataset
List
Global Environment
w num [1:14] 1 2 3 4 5 6 7 8 9 10 ....
x num [1:29] 2 4 4 6 6 5 5 7 3 7 ...
y int [1:29] 1 2 3 4 5 6 7 8 9 10 ....
z int [1:7] 1 2 3 4 5 6 7
Files Plots Packages Help Viewer
Zoom
Export
0 15
1 2
ENG
12:52
15-03-2020


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Source
Console Terminal Jobs
> x<-1:5
> test<-(x<4)
> x
[1] 1 2 3 4 5
> test<-(x==3)
> x
[1] 1 2 3 4 5
> test<-(x>1 & x<4)
> x
[1] 1 2 3 4 5
> test <- (x>4 | x==2)
> test
[1] FALSE TRUE FALSE FALSE TRUE
> test<-(x>1 & x<4)
> test
[1] FALSE TRUE TRUE FALSE FALSE
> test<-(x==3)
> test
[1] FALSE FALSE TRUE FALSE FALSE
> test<-(x!=4)
> test
[1] TRUE TRUE TRUE FALSE TRUE
Environment History Connections
Import Dataset
List
Global Environment
test logi [1:5] TRUE TRUE TRUE FALSE T...
valid.variab... 18.6
var1 3.5
w num [1:14] 1 2 3 4 5 6 7 8 9 10 ....
Files Plots Packages Help Viewer
Zoom
Export
0 15
1 2
ENG
12:56
15-03-2020





Conclusion: Thus, we have studied the basic functionality of R.

Experiment no:3

Aim: Basic data structures and data types in R-vector, matrices, list, data frames.

Theory:

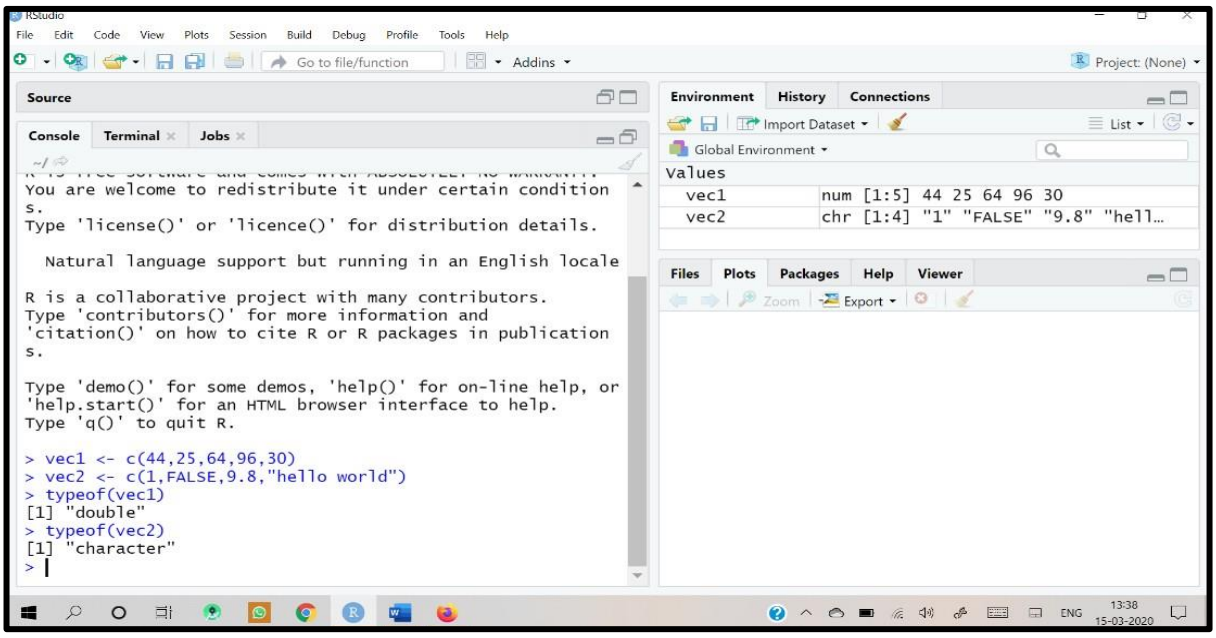
Data structures in R:

Data structures are used to store data in an organized fashion in order to make data manipulation and other data operations more efficient. There are five types of Data Structures in R Programming which are mentioned below:

- Vector
- List
- Matrix
- Data Frame

A. Vector:

Vector is one of the basic data structures in R programming. It is homogenous in nature, which means that it only contains elements of the same data type. Data types can be numeric, integer, character, complex or logical. The vector in R programming is created using the c() function. Coercion takes place in a vector from lower to top, if the elements passed are of different data types from Logical to Integer to Double to Character. The typeof() function is used to check the data type of the vector, and class() function is used to check the class of a vector.

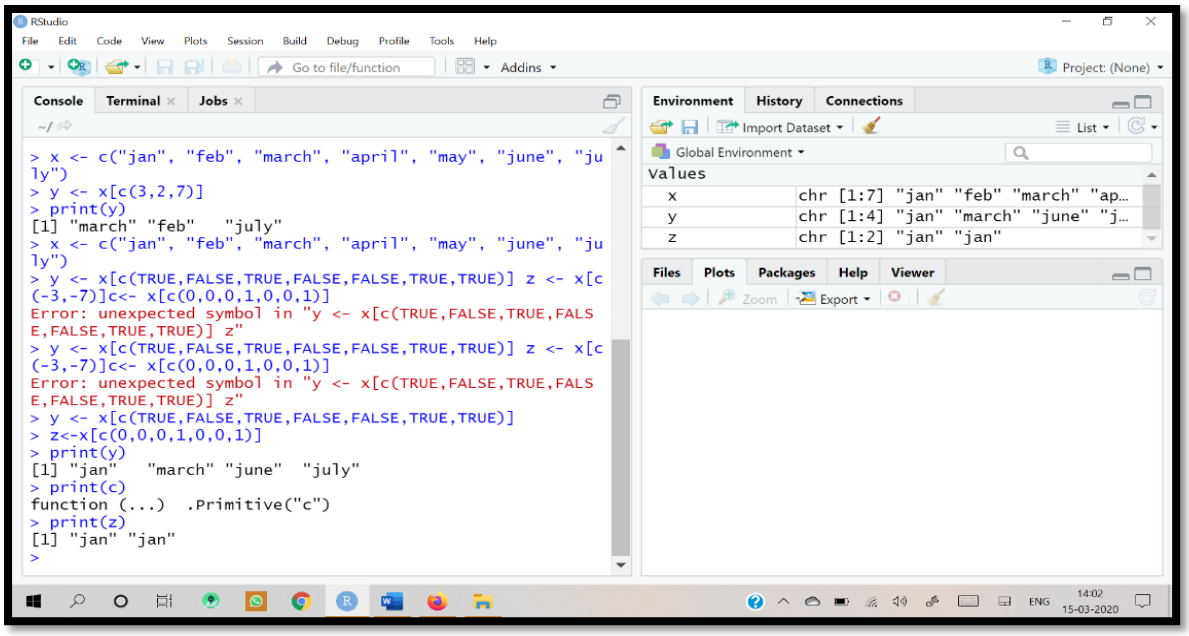


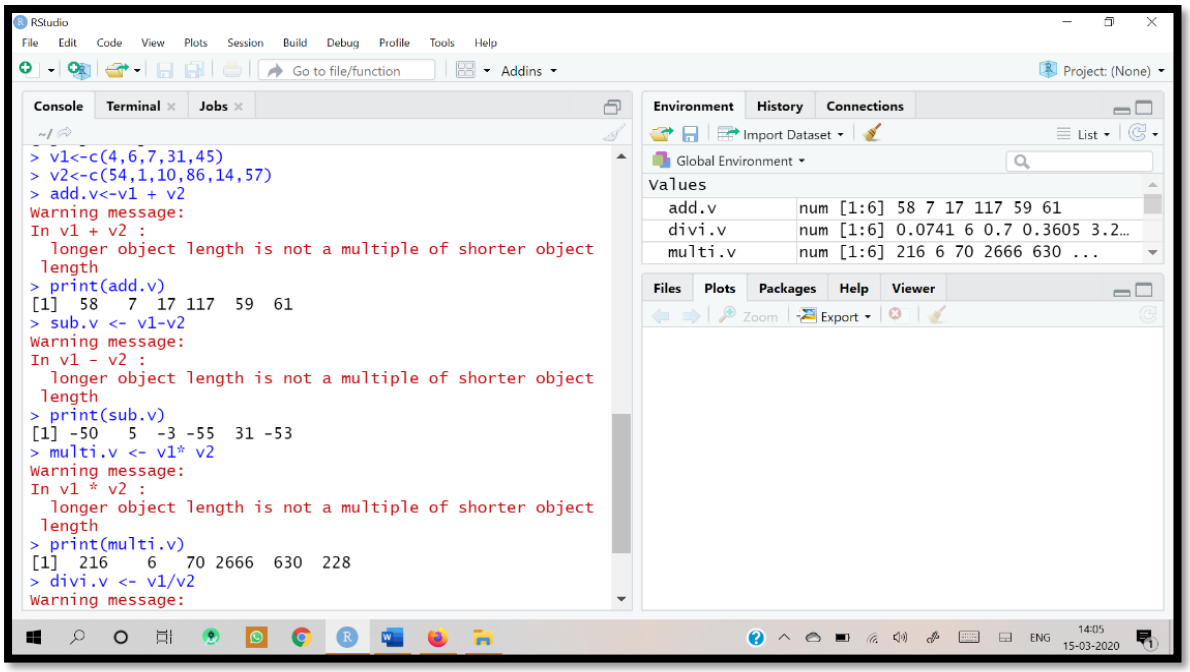
To delete a vector, we simply do the following:

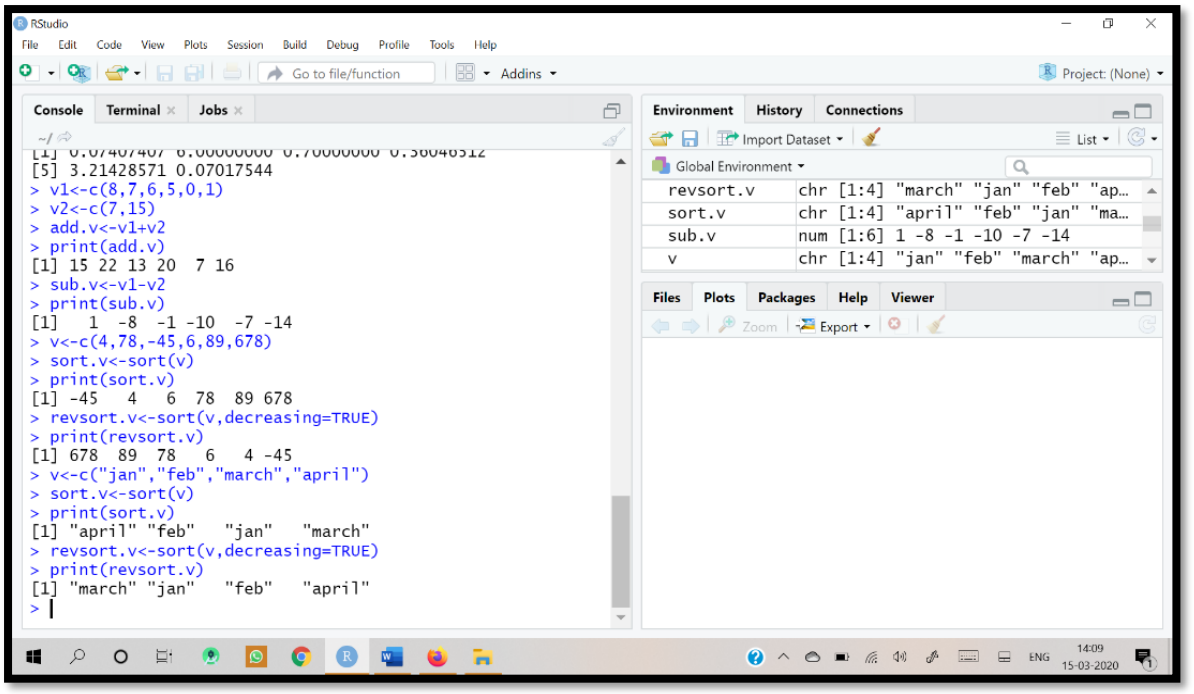
Vec1<- NULL

Vec2<- NULL

Elements of a vector can be accessed by using their respective indexes. [ ] brackets are used to specify indexes of the elements to be accessed.

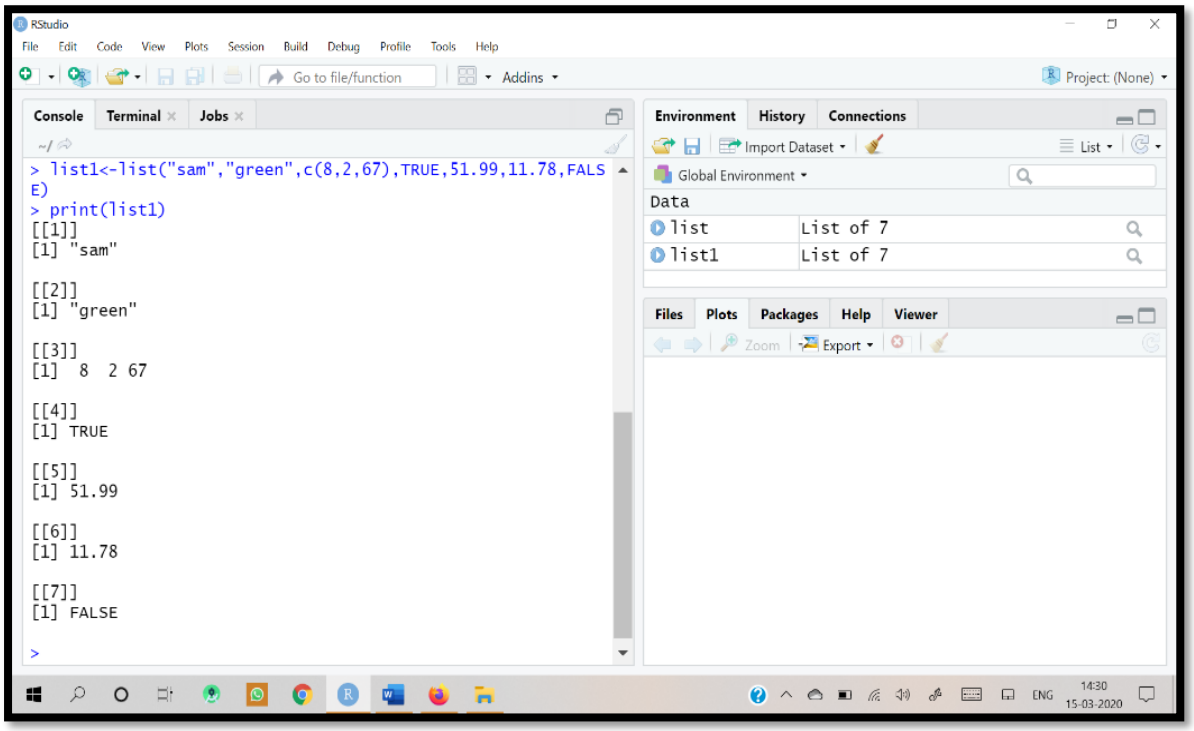


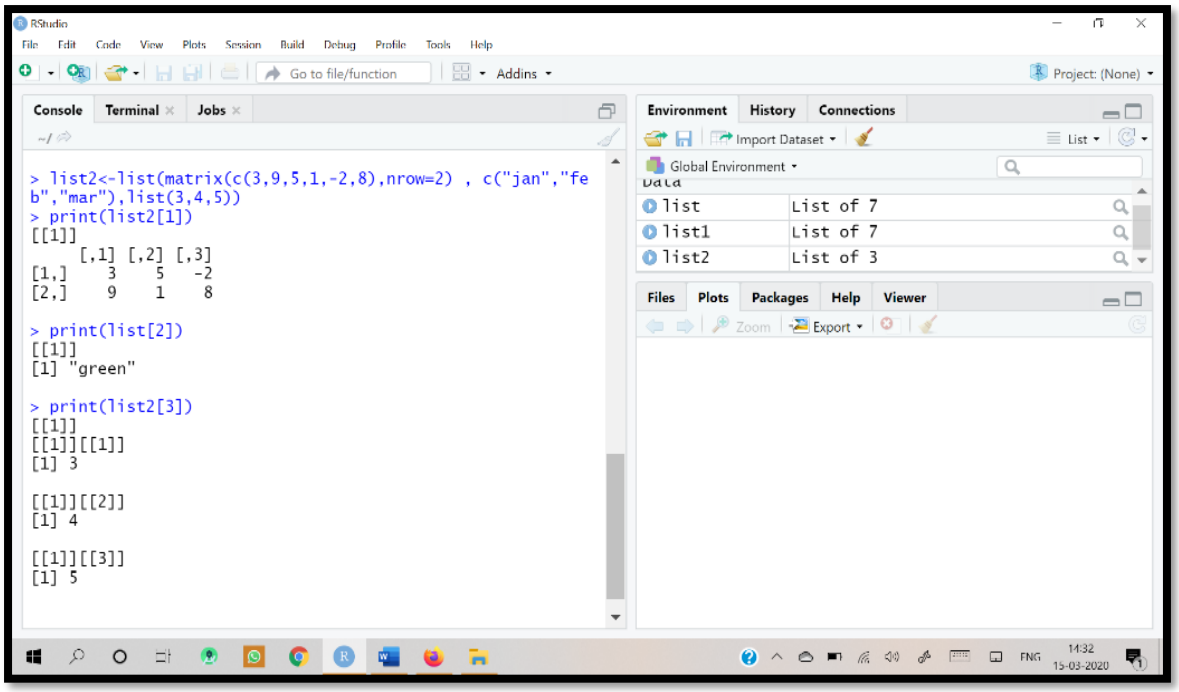


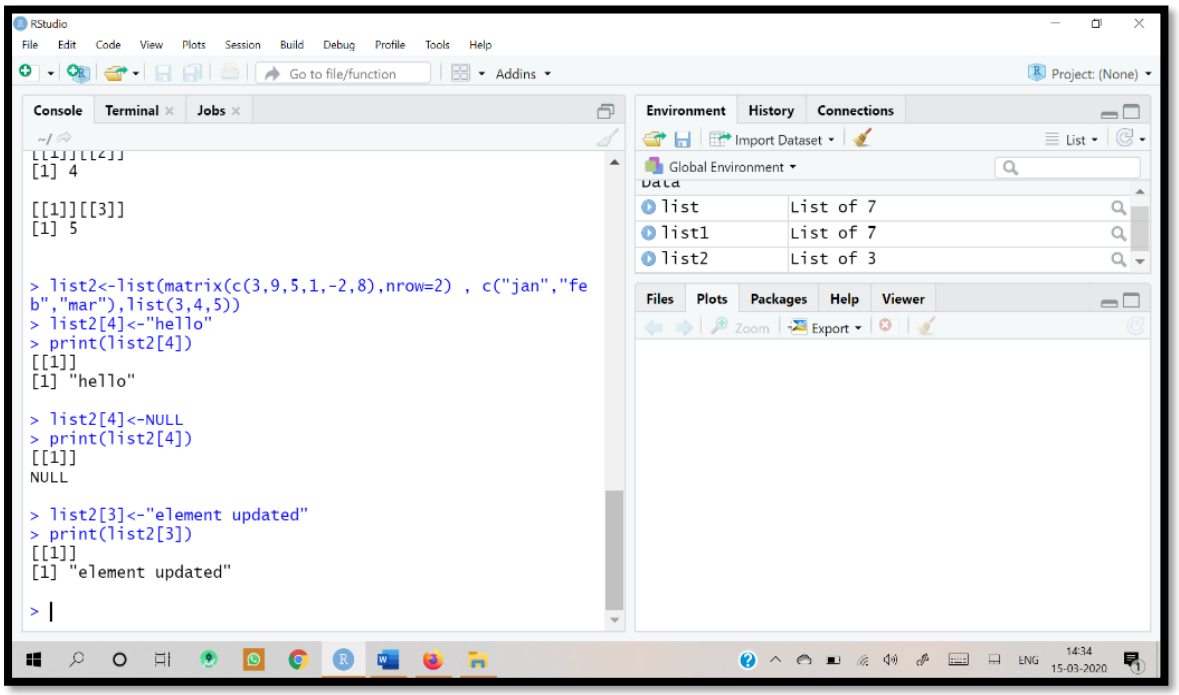


B. List:

A list in R programming is a non-homogenous data structure, which implies that it can contain elements of different data types. It accepts numbers, characters, lists, and even matrices and functions inside it. It is created using the list() function.



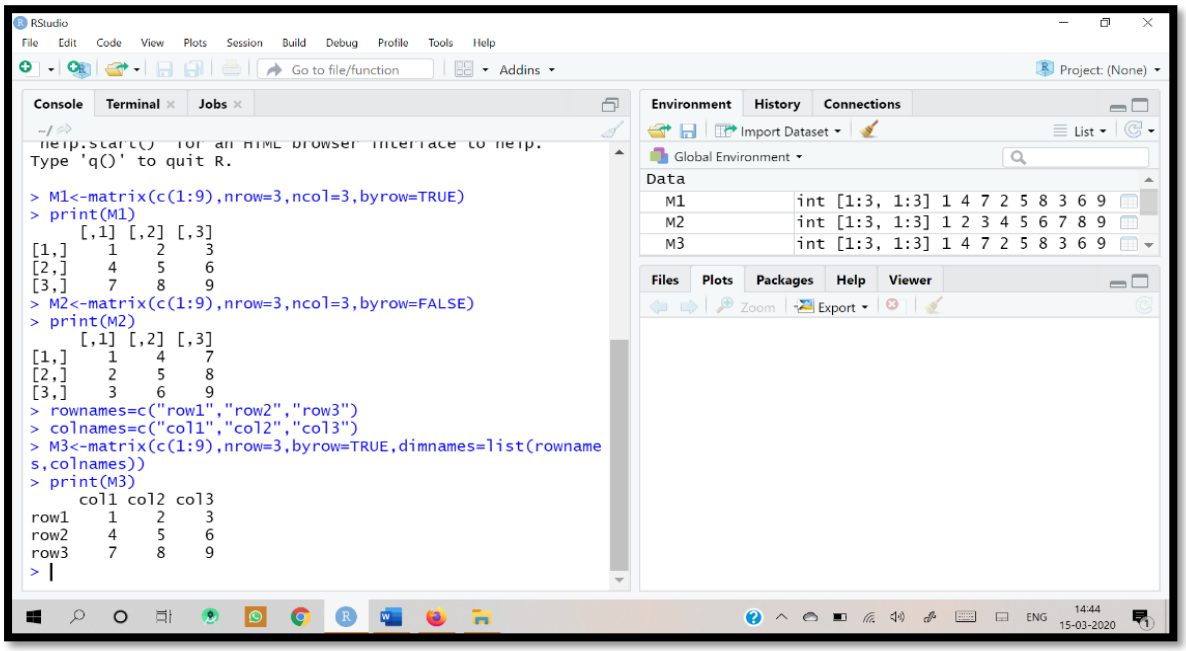


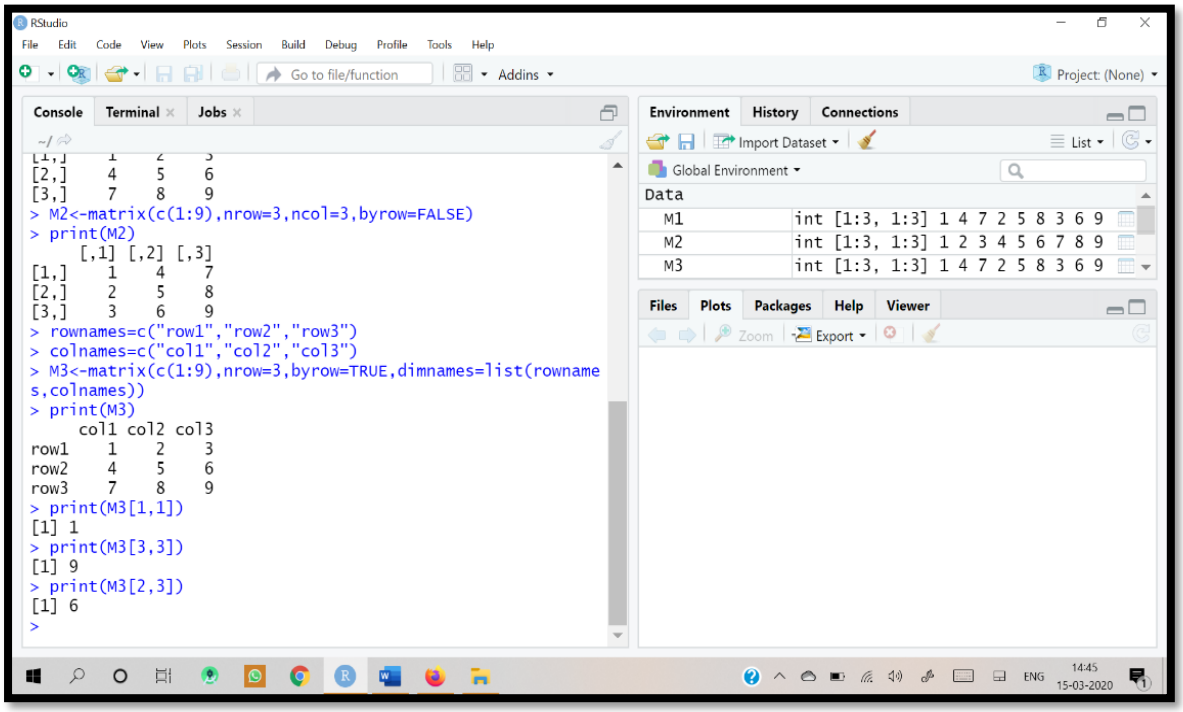


C. Matrix:

The matrix in R programming is a 2-dimensional data structure that is homogenous in nature, which means that it only accepts elements of the same data type. Coercion takes place if elements of different data types are passed. It is created using the matrix() function. The basic syntax to create a matrix is given below: matrix(data, nrow, ncol, byrow, dimnames)

where,
data = the input element of a matrix given as a vector.
nrow = the number of rows to be created.
ncol = the number of columns to be created.
byrow = the row-wise arrangement of the elements instead of column-wise
dimnames = the names of columns/rows to be created.

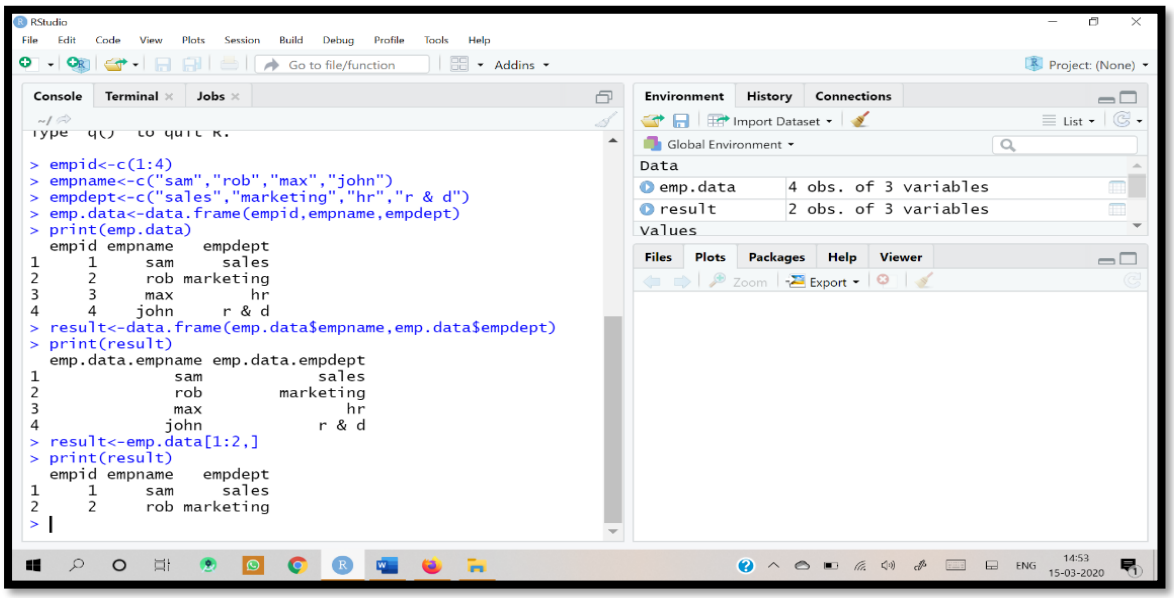


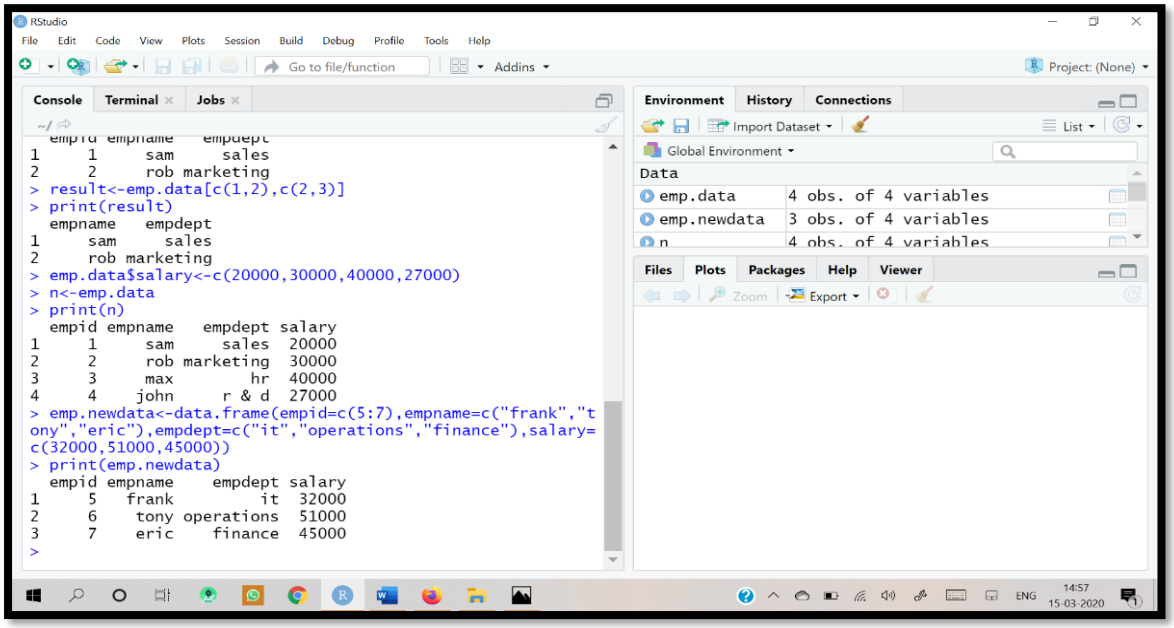


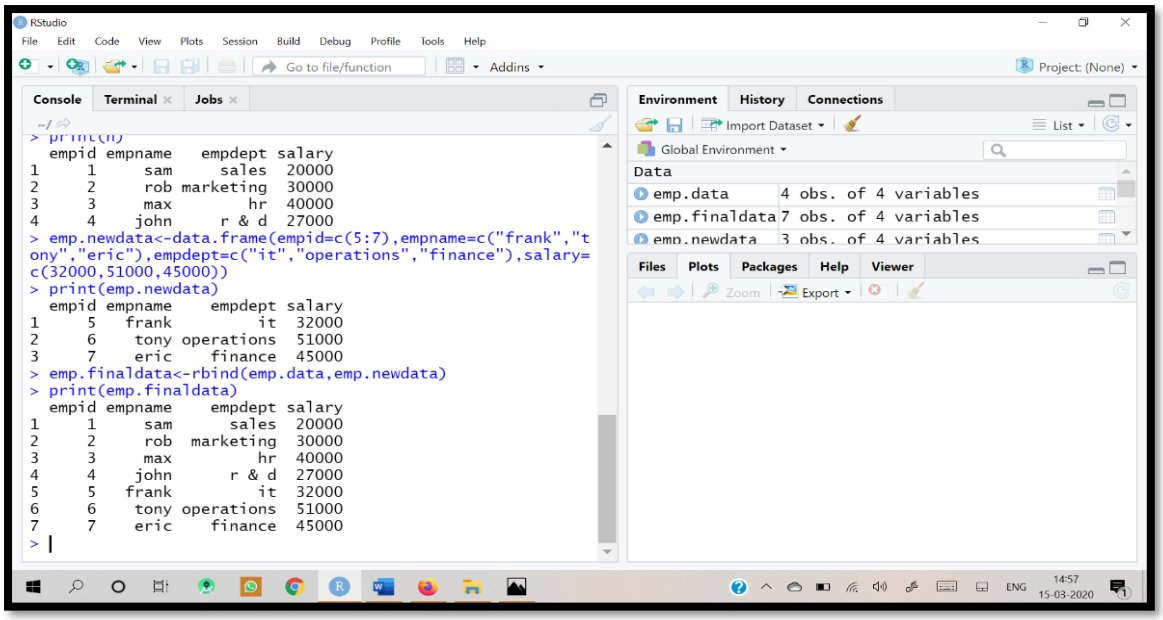
D. Data Frame:

A data frame in R programming is a 2-dimensional array-like structure that also resembles a table, in which each column contains values of one variable and each row contains one set of values from each column. A data frame has the following characteristics:

- The column names of a data frame should not be empty.
- Row names should be unique.
- Data stored in a data frame can be numeric, factor or character type.
- Each column should contain the same number of data items.







Data-Types in R:

There are many basic data types in R, which are of frequent occurrence in coding R calculations and programs. Here is the list of all the data types provided by R:

- Numeric
- Integer
- Complex
- Logical
- Character

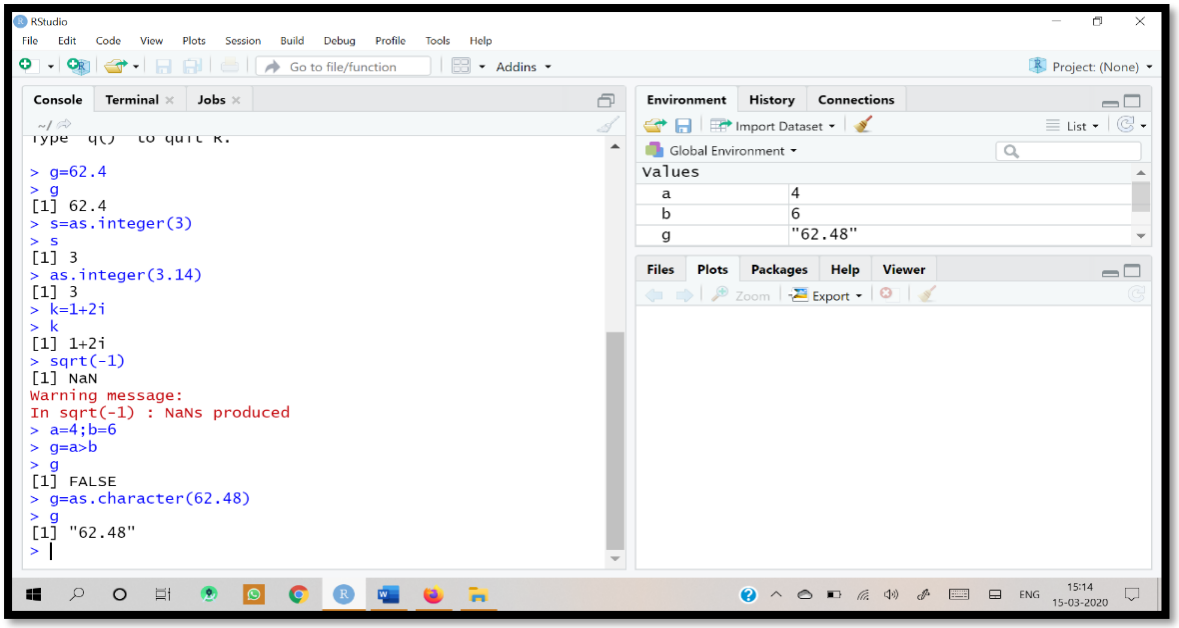
a) Number data type: Decimal values are referred to as numeric data types in R. This is the default working out data type. If you assign a decimal value for any variable x, x will become a numeric type.

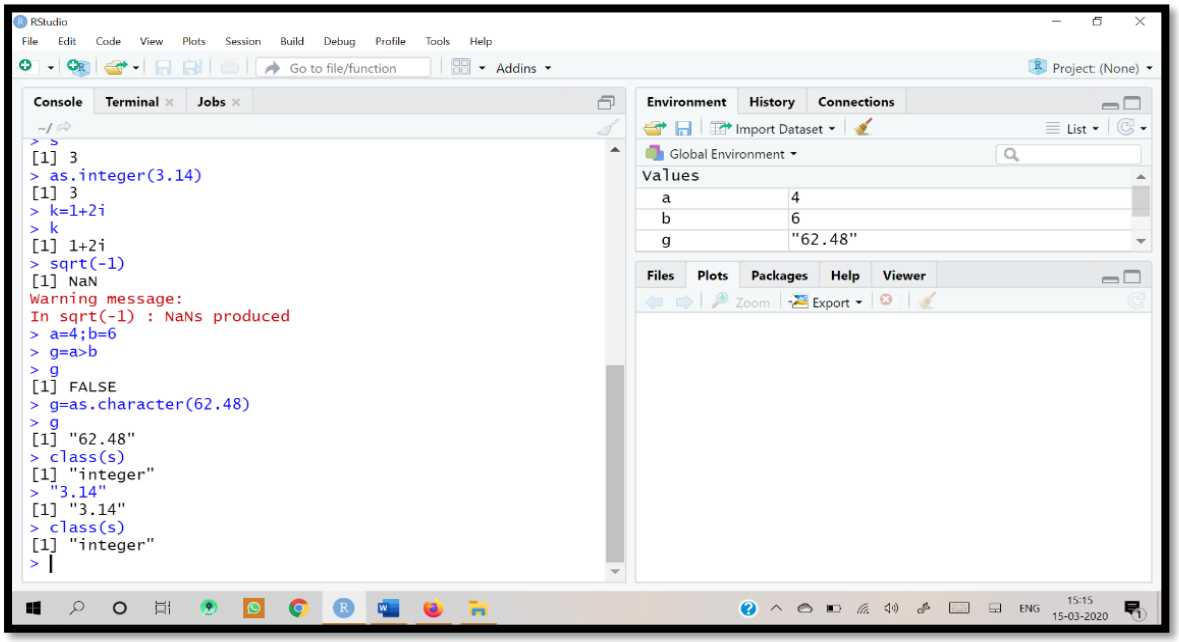
b) Integer data type: If you want to create an integer variable in R, you have to invoke the as.integer() function to define any integer type data. You can be certain that y is definitely an integer by applying the is.integer() function.

c) Complex data type: A complex value for coding in R can be defined using the pure imaginary values 'i'.

d) Logical data type: A logical value is mostly created when a comparison between variables are done.

e) Character data type: A character object can be used for representing string values in R. You have to convert objects into character values using the as.character() function within your code.





Conclusion:

Thus, we have studied data types and data structures provided by R.

Experiment no:4

Aim: Programming constructs in R-loops, conditional executions.

Theory:

A. For Loop:

A for loop is used to iterate over a vector in programming.

Syntax:

```
for (val in sequence)
{
statement
}
```

Where, sequence is a vector and val takes on each of its value during the loop. In each iteration, statement is evaluated.

Flowchart for For Loop:

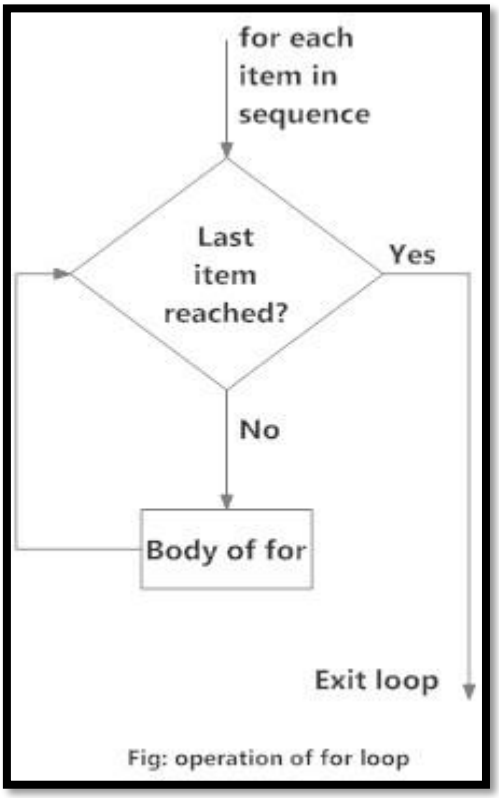
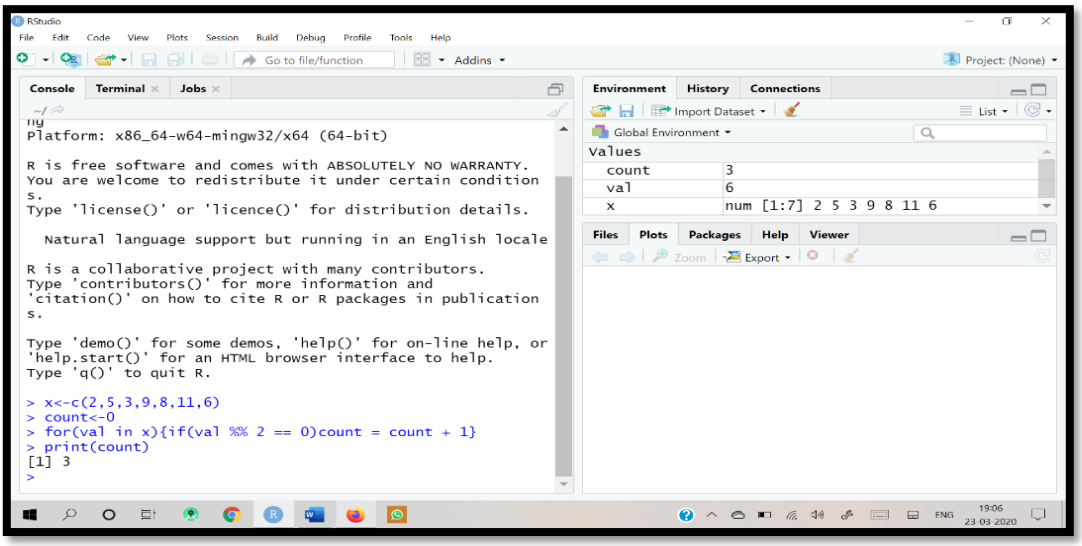


Fig: operation of for loop

Output:



B. R while Loop:

In R programming, while loops are used to loop until a specific condition is met.

Syntax:

while (test_expression)
{
statement
}

Where test_expression is evaluated and the body of the loop is entered if the result is TRUE. The statements inside the loop are executed and the flow returns to evaluate the test_expression again. This is repeated each time until test_expression evaluates to FALSE, in which case, the loop exits.

Flowchart for while loop:

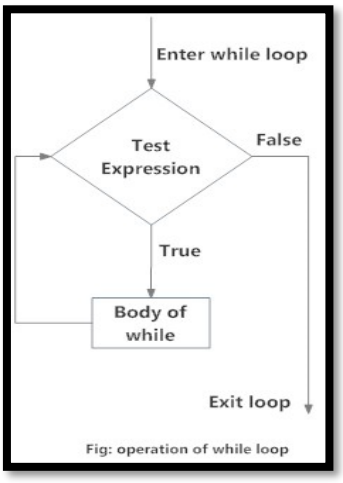
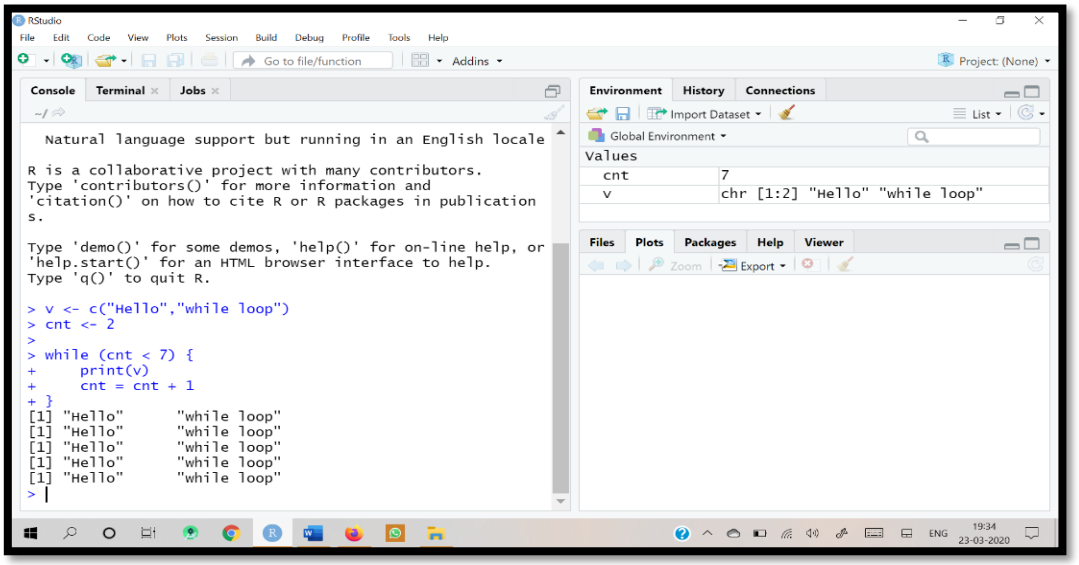


Fig: operation of while loop

Output:



C. R ifelse () Function:

Vectors form the basic building block of R programming. Most of the functions in R take vector as input and output a resultant vector. This vectorization of code, will be much faster than applying the same function to each element of the vector individually. Similar to this concept, there is a vector equivalent form of the if…else statement in R, the ifelse() function.

Syntax:

ifelse (test_expression, x, y)

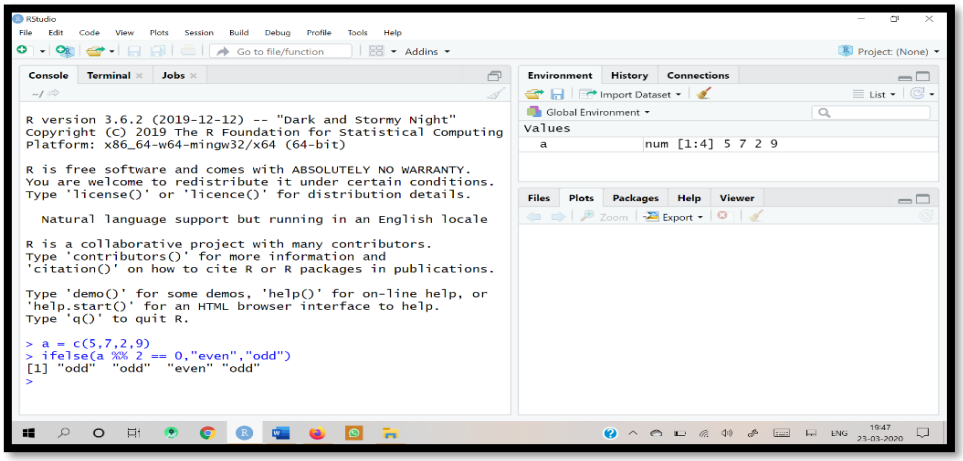
Here, test_expression must be a logical vector (or an object that can be coerced to logical). The return value is a vector with the same length as test_expression.

This returned vector has element from x if the corresponding value of test_expression is TRUE or from y if the corresponding value of test_expression is FALSE.

This is to say, the i-th element of result will be x[i] if test_expression[i] is TRUE else it will take the value of y[i].

The vectors x and y are recycled whenever necessary.

Output:



D. R break and next statement:

In R programming, a normal looping sequence can be altered using the break or the next statement.

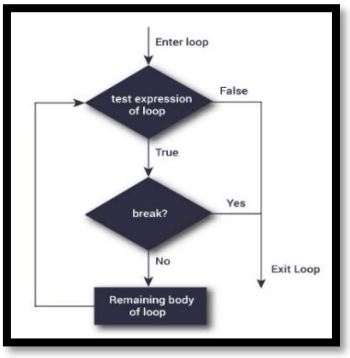
i. Break statement:

A break statement is used inside a loop (repeat, for, while) to stop the iterations and flow the control outside of the loop. In a nested looping situation, where there is a loop inside another loop, this statement exits from the innermost loop that is being evaluated.

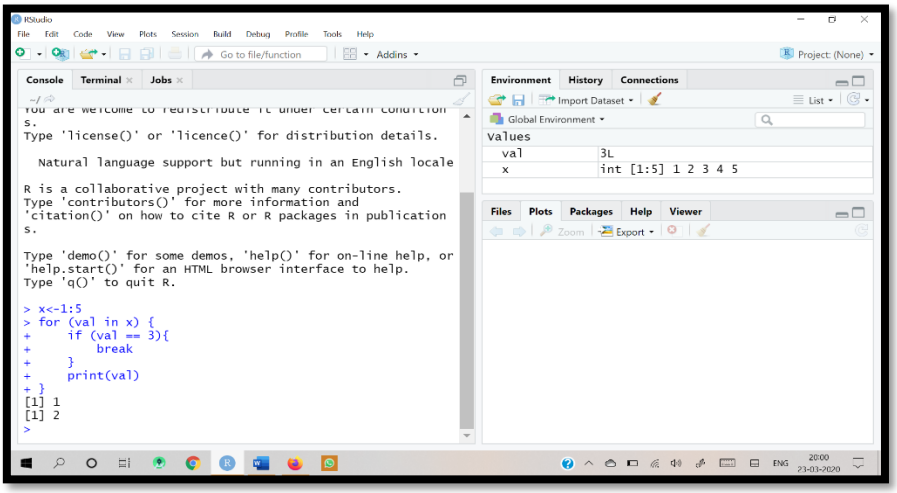
Syntax:

```
if (test_expression) {
break
}
```

Flowchart for break statement:



Output:



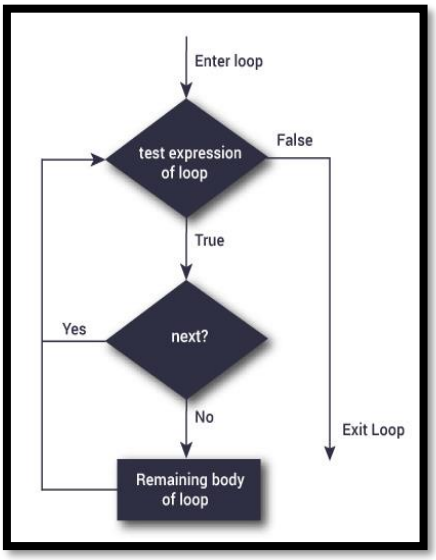
ii. Next Statement:

A next statement 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.

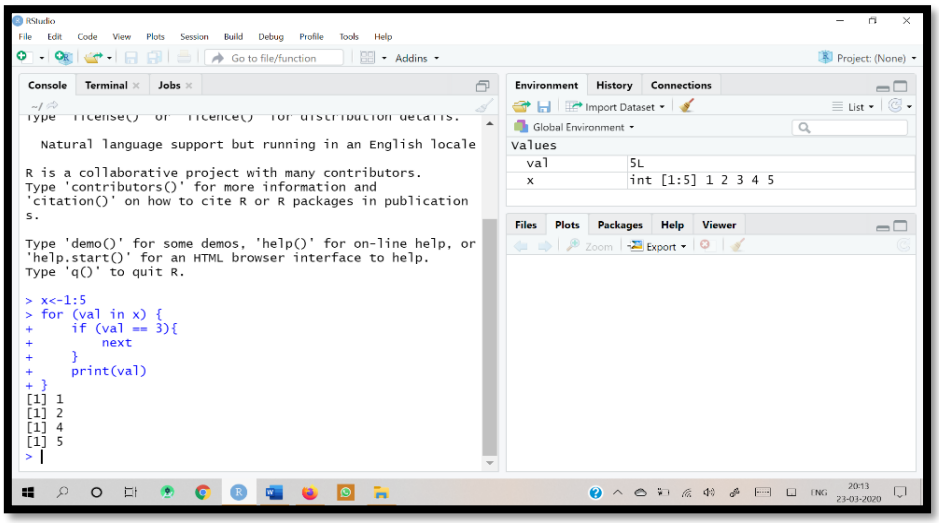
Syntax:

```
if (test_condition) {
next
}
```

Flowchart for next statement:



Output:



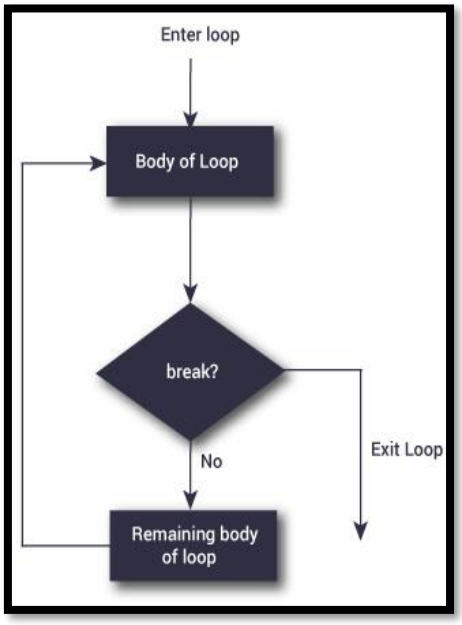
E. R repeat loop:

A repeat loop is used to iterate over a block of code multiple number of times. There is no condition check in repeat loop to exit the loop. We must ourselves put a condition explicitly inside the body of the loop and use the break statement to exit the loop. Failing to do so will result into an infinite loop.

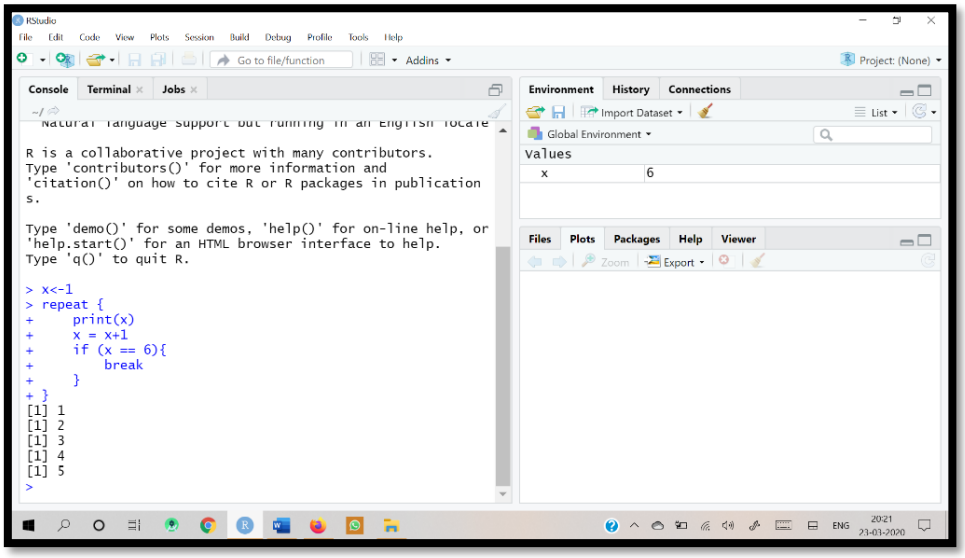
Syntax:

```
repeat {
statement
}
```

Flowchart:



Output:



R Functions:

Functions are used to logically break our code into simpler parts which become easy to maintain and understand.

Syntax:

```
func_name <- function (argument) {
statement
}
```

- Here, we can see that the reserved word function is used to declare a function in R.
- The statements within the curly braces form the body of the function. These braces are optional if the body contains only a single expression.
- Finally, this function object is given a name by assigning it to a variable, func_name.

Conclusion:

Thus, we have studied loops and functions provided by R.

Experiment no:5

Aim: Tables with labels in R - multiple response variables excel function translations.

Theory:

expss computes and displays tables with support for 'SPSS'-style labels, multiple / nested banners, weights, multiple-response variables and significance testing. There are facilities for nice output of tables in 'knitr', R notebooks, 'Shiny' and 'Jupyter' notebooks. Proper methods for labelled variables add value labels support to base R functions and to some functions from other packages. Additionally, the package offers useful functions for data processing in marketing research / social surveys - popular data transformation functions from 'SPSS' Statistics ('RECODE', 'COUNT', 'COMPUTE', 'DO IF', etc.) and 'Excel' ('COUNTIF', 'VLOOKUP', etc.). Package is intended to help people to move data processing from 'Excel'/ 'SPSS' to R. See examples below. You can get help about any function by typing function_name in the R console.

Installation:

expss is on CRAN, so for installation you can print in the consoleinstall. packages("expss").

```
library(expss)data(mtcars)mtcars = apply_labels(mtcars,
mpg = "Miles/(US) gallon",
cyl = "Number of cylinders",
disp = "Displacement (cu.in.)",
hp = "Gross horsepower",
drat = "Rear axle ratio",
wt = "Weight (1000 lbs)",
qsec = "1/4 mile time",
vs = "Engine",
vs = c("V-engine" = 0,
"Straight engine" = 1),
am = "Transmission",
am = c("Automatic" = 0,
"Manual"=1),
gear = "Number of forward gears",
carb = "Number of carburetors"
)
```

For quick cross-tabulation there are fre and cro family of function. For simplicity we demonstrate here only cro_cpct which calculates column percent. Documentation for other functions, such as cro_cases for counts, cro_rpct for row percent, cro_tpct for table percent and cro_fun for custom summary functions can be seen by typing? cro and ? cro_fun in the console.

```
# 'cro' examples
# just simple crosstabulation, similar to base R 'table' function
cro (mtcars$am, mtcars$vs)

# Table column % with multiple banners
cro_cpct (mtcars$cyl, list (total (), mtcars$am, mtcars$vs))

# or, the same result with another notation
mtcars %>% calc_cro_cpct (cyl, list (total (), am, vs)
```

We have more sophisticated interface for table construction with magrittr piping. Table construction consists of at least of three functions chained with pipe operator: %>%.
At first, we need to specify variables for which statistics will be computed with tab_cells. Secondary, we calculate statistics with one of the tabs_stat_* functions. And last, we finalize table creation with tab_pivot, e. g.: dataset %>% tab_cells(variable) %>% tab_stat_cases () %>% tab_pivot (). After that we can optionally sort table with tab_sort_asc, drop empty rows/columns with drop_rc and transpose with tab_transpose. Resulting table is just a data. Frame so we can use usual R operations on it. Detailed documentation for table creation can be seen via? tables. For significance testing see? significance. Generally, tables automatically translated to HTML for output in knitr or Jupyter notebooks. However, if we want HTML output in the R notebooks or in the RStudio viewer we need to set options for that: expss_output_rnotebook () or expss_output_viewer ().

Output:

```
Console  Terminal  Jobs
~/
> library(expss)
> data(mtcars)
> mtcars = apply_labels(mtcars,
+                       mpg = "Miles/(US) gallon",
+                       cyl = "Number of cylinders",
+                       disp = "Displacement (cu.in.)",
+                       hp = "Gross horsepower",
+                       drat = "Rear axle ratio",
+                       wt = "Weight (1000 lbs)",
+                       qsec = "1/4 mile time",
+                       vs = "Engine",
+                       vs = c("V-engine" = 0,
+                              "Straight engine" = 1),
+                       am = "Transmission",
+                       am = c("Automatic" = 0,
+                              "Manual"=1),
+                       gear = "Number of forward gears",
+                       carb = "Number of carburetors"
+ )
> cro(mtcars$am, mtcars$vs)

 |              |              |   Engine |                 |
 |              |              | V-engine | Straight engine |
 | ------------ | ------------ | -------- | --------------- |
 | Transmission |    Automatic |       12 |               7 |
 |              |       Manual |        6 |               7 |
 |              | #Total cases |       18 |              14 |
>
> cro_cpct(mtcars$cyl, list(total(), mtcars$am, mtcars$vs))

 |                    |              | #Total | Transmission |        |   Engine |                 |
 |                    |              |        |    Automatic | Manual | V-engine | Straight engine |
 | ------------------ | ------------ | ------ | ------------ | ------ | -------- | --------------- |
 | Number of cylinders |            4 |   34.4 |         15.8 |   61.5 |      5.6 |            71.4 |
 |                    |            6 |   21.9 |         21.1 |   23.1 |     16.7 |            28.6 |
 |                    |            8 |   43.8 |         63.2 |   15.4 |     77.8 |                 |
 |                    | #Total cases |   32.0 |         19.0 |   13.0 |     18.0 |            14.0 |
>
```

```
Console  Terminal  Jobs
~/
> mtcars %>% calc_cro_cpct(cyl, list(total(), am, vs))

 |                    |              | #Total | Transmission |        |   Engine |                 |
 |                    |              |        |    Automatic | Manual | V-engine | Straight engine |
 | ------------------ | ------------ | ------ | ------------ | ------ | -------- | --------------- |
 | Number of cylinders |            4 |   34.4 |         15.8 |   61.5 |      5.6 |            71.4 |
 |                    |            6 |   21.9 |         21.1 |   23.1 |     16.7 |            28.6 |
 |                    |            8 |   43.8 |         63.2 |   15.4 |     77.8 |                 |
 |                    | #Total cases |   32.0 |         19.0 |   13.0 |     18.0 |            14.0 |
>
> mtcars %>% calc_cro_cpct(cyl, list(total(), am %nest% vs))

 |                    |              | #Total | Transmission |                 |         |
 |                    |              |        |    Automatic |                 |  Manual |
 |                    |              |        |       Engine |                 |  Engine |
 |                    |              |        |    V-engine | Straight engine | V-engine |
 | ------------------ | ------------ | ------ | ------------ | --------------- | -------- |
 | Number of cylinders |            4 |   34.4 |              |            42.9 |     16.7 |
 |                    |            6 |   21.9 |              |            57.1 |     50.0 |
 |                    |            8 |   43.8 |          100 |                  |     33.3 |
 |                    | #Total cases |   32.0 |           12 |             7.0 |      6.0 |

 |
 |
 |
 | Straight engine |
 | --------------- |
 |             100 |
 |                 |
 |                 |
 |               7 |
>
> mtcars %>%
+     tab_cells(cyl) %>%
+     tab_cols(total(), am) %>%
+     tab_stat_cpct() %>%
+     tab_pivot()
```

```
 |                     |             | #Total | Transmission |        |
 |                     |             |        |    Automatic | Manual |
 | ------------------- | ----------- | ------ | ------------ | ------ |
 | Number of cylinders |           4 |   34.4 |         15.8 |   61.5 |
 |                     |           6 |   21.9 |         21.1 |   23.1 |
 |                     |           8 |   43.8 |         63.2 |   15.4 |
 |                     | #Total cases |   32.0 |         19.0 |   13.0 |
> mtcars %>%
+     tab_cells(mpg, disp, hp, wt, qsec) %>%
+     tab_cols(total(), am) %>%
+     tab_stat_mean_sd_n() %>%
+     tab_last_sig_means(subtable_marks = "both") %>%
+     tab_pivot() %>%
+     set_caption("Table with summary statistics and significance marks.")
 Table with summary statistics and significance marks.
 |                     |             | #Total | Transmission |          |
 |                     |             |        |    Automatic |   Manual |
 |                     |             |        |            A |        B |
 | ------------------- | ----------- | ------ | ------------ | -------- |
 |     Miles/(US) gallon |        Mean |   20.1 |     17.1 < B | 24.4 > A |
 |                     |    Std. dev. |    6.0 |          3.8 |      6.2 |
 |                     | Unw. valid N |   32.0 |         19.0 |     13.0 |
 | Displacement (cu.in.) |        Mean |  230.7 |    290.4 > B | 143.5 < A |
 |                     |    Std. dev. |  123.9 |        110.2 |     87.2 |
 |                     | Unw. valid N |   32.0 |         19.0 |     13.0 |
 |      Gross horsepower |        Mean |  146.7 |        160.3 |    126.8 |
 |                     |    Std. dev. |   68.6 |         53.9 |     84.1 |
 |                     | Unw. valid N |   32.0 |         19.0 |     13.0 |
 |     Weight (1000 lbs) |        Mean |    3.2 |      3.8 > B | 2.4 < A |
 |                     |    Std. dev. |    1.0 |          0.8 |      0.6 |
 |                     | Unw. valid N |   32.0 |         19.0 |     13.0 |
 |          1/4 mile time |        Mean |   17.8 |         18.2 |     17.4 |
 |                     |    Std. dev. |    1.8 |          1.8 |      1.8 |
 |                     | Unw. valid N |   32.0 |         19.0 |     13.0 |
> # Table with the same summary statistics. Statistics labels in columns.
> mtcars %>%
+     tab_cells(mpg, disp, hp, wt, qsec) %>%
+     tab_cols(total(label = "#Total| |"), am) %>%
+     tab_stat_fun(Mean = w_mean, "Std. dev." = w_sd, "Valid N" = w_n, method = list) %>%
+     tab_pivot()
```

```
 |                     | #Total |             |          | Transmission |
 |                     |        |             |          |    Automatic |
 |                     |   Mean | Std. dev.   | Valid N  |         Mean |
 | ------------------- | ------ | ----------- | -------- | ------------ |
 |     Miles/(US) gallon |   20.1 |         6.0 |       32 |         17.1 |
 | Displacement (cu.in.) |  230.7 |       123.9 |       32 |        290.4 |
 |      Gross horsepower |  146.7 |        68.6 |       32 |        160.3 |
 |     Weight (1000 lbs) |    3.2 |         1.0 |       32 |          3.8 |
 |          1/4 mile time |   17.8 |         1.8 |       32 |         18.2 |

             |          |        |             |          |
             |          | Manual |             |          |
 Std. dev.   | Valid N  |   Mean | Std. dev.   | Valid N  |
 ----------- | -------- | ------ | ----------- | -------- |
         3.8 |       19 |   24.4 |         6.2 |       13 |
       110.2 |       19 |  143.5 |        87.2 |       13 |
        53.9 |       19 |  126.8 |        84.1 |       13 |
         0.8 |       19 |    2.4 |         0.6 |       13 |
         1.8 |       19 |   17.4 |         1.8 |       13 |
> # Different statistics for different variables.
> mtcars %>%
+     tab_cols(total(), vs) %>%
+     tab_cells(mpg) %>%
+     tab_stat_mean() %>%
+     tab_stat_valid_n() %>%
+     tab_cells(am) %>%
+     tab_stat_cpct(total_row_position = "none", label = "col %") %>%
+     tab_stat_rpct(total_row_position = "none", label = "row %") %>%
+     tab_stat_tpct(total_row_position = "none", label = "table %") %>%
+     tab_pivot(stat_position = "inside_rows")

 |                     |             |          | #Total |   Engine |
 |                     |             |          |        | V-engine |
 | ------------------- | ----------- | -------- | ------ | -------- |
 | Miles/(US) gallon |        Mean |          |   20.1 |     16.6 |
 |                     |    Valid N |          |   32.0 |     18.0 |
 |        Transmission | Automatic |    col % |   59.4 |     66.7 |
 |                     |             |    row % |  100.0 |     63.2 |
 |                     |             |  table % |   59.4 |     37.5 |
 |                     |     Manual |    col % |   40.6 |     33.3 |
 |                     |             |    row % |  100.0 |     46.2 |
 |                     |             |  table % |   40.6 |     18.8 |
```

```
Console    Terminal ×    Jobs ×
~/
 Straight engine |
 --------------- |
            24.6 |
            14.0 |
            50.0 |
            36.8 |
            21.9 |
            50.0 |
            53.8 |
            21.9 |
> # Table with split by rows and with custom totals.
> mtcars %>%
+     tab_cells(cyl) %>%
+     tab_cols(total(), vs) %>%
+     tab_rows(am) %>%
+     tab_stat_cpct(total_row_position = "above",
+                   total_label = c("number of cases", "row %"),
+                   total_statistic = c("u_cases", "u_rpct")) %>%
+     tab_pivot()

 |              |           |                      |                  | #Total |
 |              |           |                      |                  |        |
 | ------------ | --------- | -------------------- | ---------------- | ------ |
 | Transmission | Automatic | Number of cylinders  | #number of cases |   19.0 |
 |              |           |                      |           #row % |  100.0 |
 |              |           |                      |                4 |   15.8 |
 |              |           |                      |                6 |   21.1 |
 |              |           |                      |                8 |   63.2 |
 |              |    Manual | Number of cylinders  | #number of cases |   13.0 |
 |              |           |                      |           #row % |  100.0 |
 |              |           |                      |                4 |   61.5 |
 |              |           |                      |                6 |   23.1 |
 |              |           |                      |                8 |   15.4 |

   Engine |                 |
 V-engine | Straight engine |
 -------- | --------------- |
     12.0 |             7.0 |
     63.2 |            36.8 |
          |            42.9 |
          |            57.1 |
    100.0 |                 |
      6.0 |             7.0 |
```

```
Console    Terminal ×    Jobs ×
~/
> data(product_test)
>
> w = product_test # shorter name to save some keystrokes
>
> # here we recode variables from first/second tested product to separate variables for each product according to their cells
> # 'h' variables - VSX123 sample, 'p' variables - 'SDF456' sample
> # also we recode preferences from first/second product to true names
> # for first cell there are no changes, for second cell we should exchange 1 and 2.
> w = w %>%
+     do_if(cell == 1, {
+         recode(a1_1 %to% a1_6, other ~ copy) %into% (h1_1 %to% h1_6)
+         recode(b1_1 %to% b1_6, other ~ copy) %into% (p1_1 %to% p1_6)
+         recode(a22, other ~ copy) %into% h22
+         recode(b22, other ~ copy) %into% p22
+         c1r = c1
+     }) %>%
+     do_if(cell == 2, {
+         recode(a1_1 %to% a1_6, other ~ copy) %into% (p1_1 %to% p1_6)
+         recode(b1_1 %to% b1_6, other ~ copy) %into% (h1_1 %to% h1_6)
+         recode(a22, other ~ copy) %into% p22
+         recode(b22, other ~ copy) %into% h22
+         recode(c1, 1 ~ 2, 2 ~ 1, other ~ copy) %into% c1r
+     }) %>%
+     compute({
+         # recode age by groups
+         age_cat = recode(s2a, lo %thru% 25 ~ 1, lo %thru% hi ~ 2)
+         # count number of likes
+         # codes 2 and 99 are ignored.
+         h_likes = count_row_if(1 | 3 %thru% 98, h1_1 %to% h1_6)
+         p_likes = count_row_if(1 | 3 %thru% 98, p1_1 %to% p1_6)
+     })
>
> # here we prepare labels for future usage
> codeframe_likes = num_lab("
+     1 Liked everything
+     2 Disliked everything
+     3 Chocolate
+     4 Appearance
+     5 Taste
+     6 Stuffing
+     7 Nuts
+     8 Consistency
+     98 Other
+     99 Hard to answer
```

Conclusion: Thus, we have studied tables with labels in R.

Experiment no:6

Aim: Working with large datasets with dplyr and data-table.

Theory:

dplyr is the next iteration of plyr, focussed on tools for working with data frames (hence the d in the name). It has three main goals:

- Identify the most important data manipulation tools needed for data analysis and make them easy to use from R.
- Provide blazing fast performance for in-memory data by writing key pieces in C++.
- Use the same interface to work with data no matter where it's stored, whether in a data frame, a data table or database.

One can install:

- The latest released version from CRAN with

```
install.packages("dplyr")
```

- The latest development version from github with

```
if (packageVersion("devtools") < 1.6)
{
  install.packages("devtools")
}
devtools::install_github("hadley/lazyeval")
devtools::install_github("hadley/dplyr")
```

tbls:

The key object in dplyr is a tbl, a representation of a tabular data structure. Currently dplyr supports:

- data frames
- [data tables]
- SQLite
- PostgreSQL
- MySQL/MariaDB

- Bigquery
- MonetDB
- data cubes with arrays (partial implementation)

```
library(dplyr)
# for functions library(hflights)
# for data head(hflights)

# Coerce to data table
hflights_dt <- tbl_dt(hflights)

# Caches data in local SQLite db
hflights_db1 <- tbl(hflights_sqlite(), "hflights")

# Caches data in local postgres db
hflights_db2 <- tbl(hflights_postgres(), "hflights")
```

Each tbl also comes in a grouped variant which allows one to easily perform operations "by group":

```
carriers_df  <- group_by(hflights, UniqueCarrier)
carriers_dt  <- group_by(hflights_dt, UniqueCarrier)
carriers_db1 <- group_by(hflights_db1, UniqueCarrier)
carriers_db2 <- group_by(hflights_db2, UniqueCarrier)
```

dplyr can be effectively used to

- Pick observations by their values (filter ()).
- Reorder the rows (arrange ()).
- Pick variables by their names (select ()).
- Create new variables with functions of existing variables (mutate ())
- Collapse many values down to a single summary (summarise ()).

These can all be used in conjunction with group_by () which changes the scope of each function from operating on the entire dataset to operating on its group-by-group. These six functions provide the verbs for a language of data manipulation.

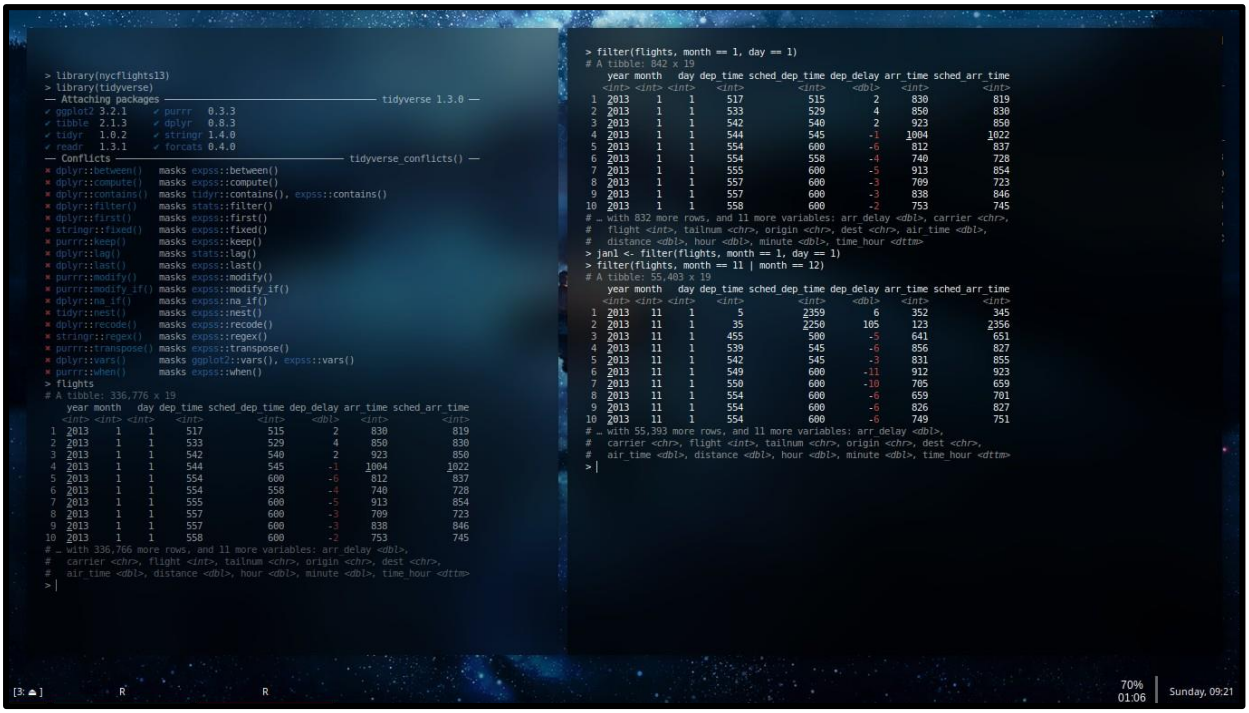
<u>All verbs work similarly:</u>

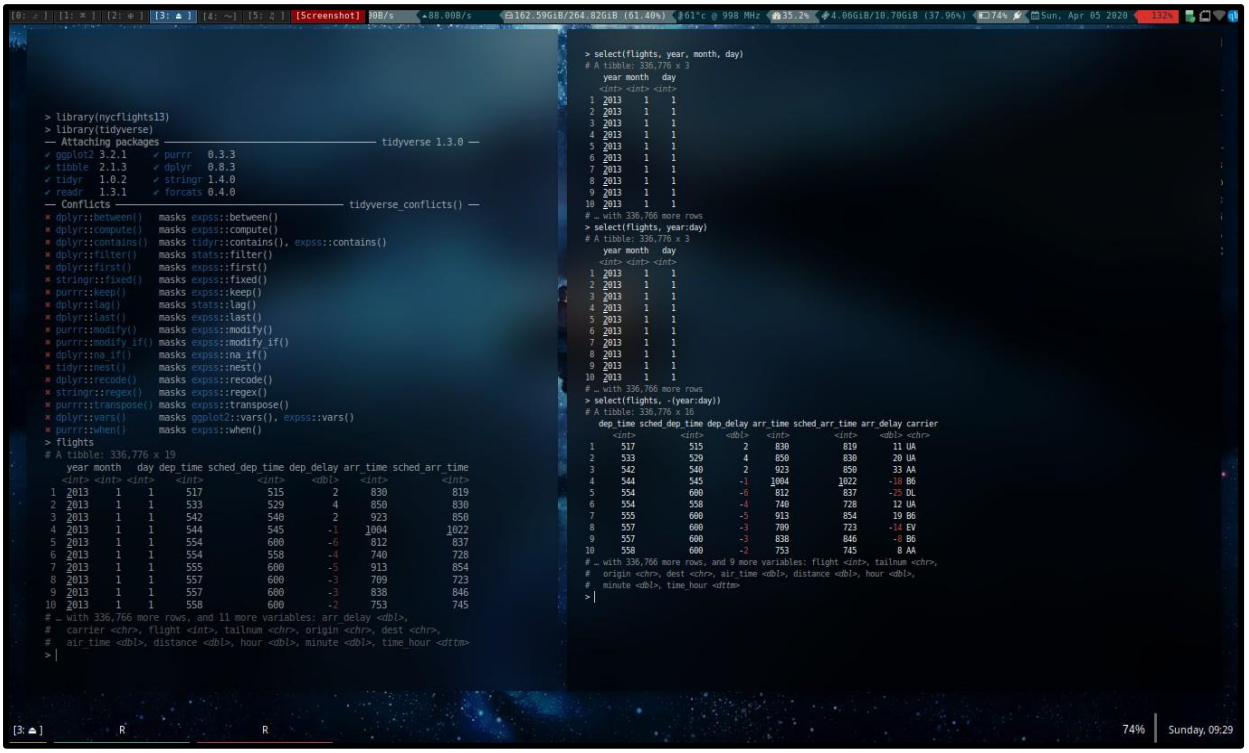
1. The first argument is a data frame.

2. The subsequent arguments describe what to do with the data frame, using the variable names (without quotes).

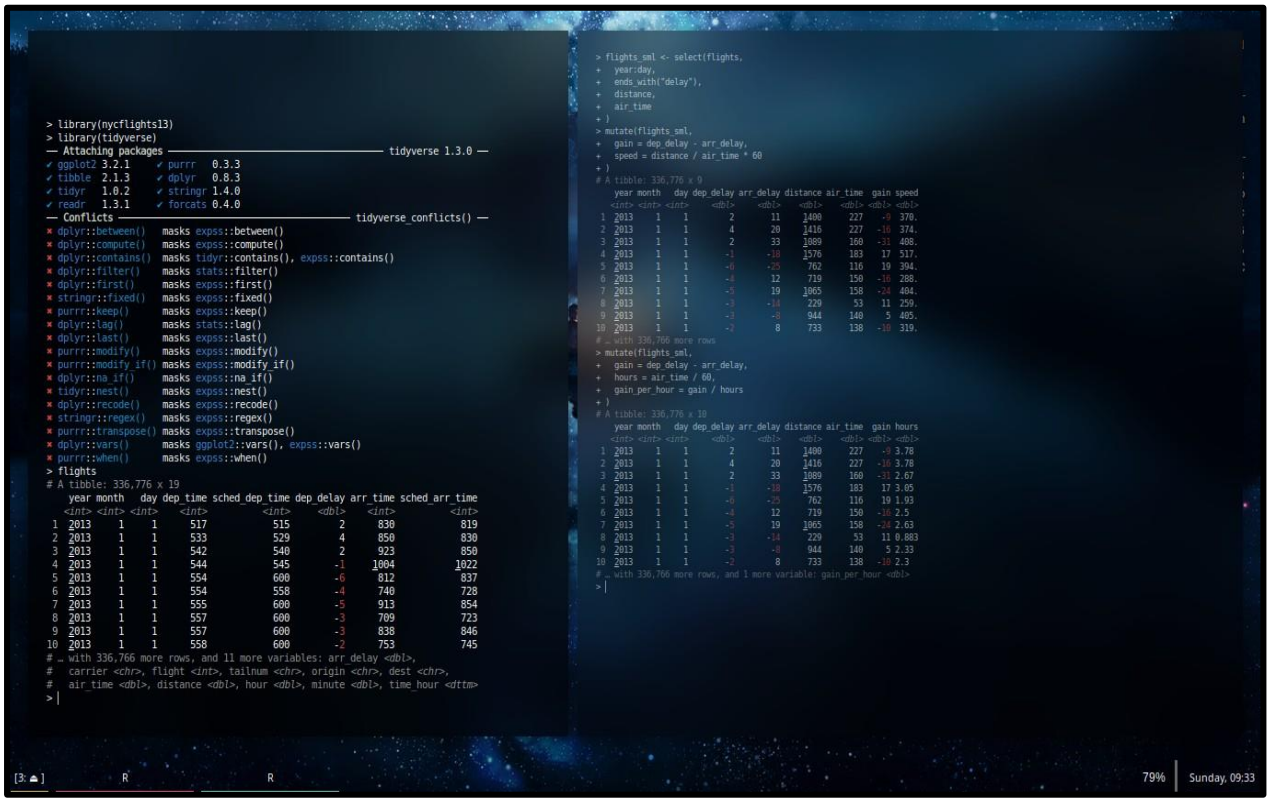
3. The result is a new data frame.

Output:






> library(nycflights13)
> library(tidyverse)
── Attaching packages ──────────────────────────────────── tidyverse 1.3.0 ──
✔ ggplot2 3.2.1 ✔ purrr 0.3.3
✔ tibble 2.1.3 ✔ dplyr 0.8.3
✔ tidyr 1.0.2 ✔ stringr 1.4.0
✔ readr 1.3.1 ✔ forcats 0.4.0
── Conflicts ─────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::between() masks expss::between()
✖ dplyr::compute() masks expss::compute()
✖ dplyr::contains() masks tidyr::contains(), expss::contains()
✖ dplyr::filter() masks stats::filter()
✖ dplyr::first() masks expss::first()
✖ stringr::fixed() masks expss::fixed()
✖ purrr::keep() masks expss::keep()
✖ dplyr::lag() masks stats::lag()
✖ dplyr::last() masks expss::last()
✖ purrr::modify() masks expss::modify()
✖ purrr::modify_if() masks expss::modify_if()
✖ dplyr::na_if() masks expss::na_if()
✖ tidyr::nest() masks expss::nest()
✖ dplyr::recode() masks expss::recode()
✖ stringr::regex() masks expss::regex()
✖ purrr::transpose() masks expss::transpose()
✖ dplyr::vars() masks ggplot2::vars(), expss::vars()
✖ purrr::when() masks expss::when()
> flights
A tibble: 336,776 x 19
year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time
<int> <int> <int> <int> <int> <dbl> <int> <int>
1 2013 1 1 517 515 2 830 819
2 2013 1 1 533 529 4 850 830
3 2013 1 1 542 540 2 923 850
4 2013 1 1 544 545 -1 1004 1022
5 2013 1 1 554 600 -6 812 837
6 2013 1 1 554 558 -4 740 728
7 2013 1 1 555 600 -5 913 854
8 2013 1 1 557 600 -3 709 723
9 2013 1 1 557 600 -3 838 846
10 2013 1 1 558 600 -2 753 745
… with 336,766 more rows, and 11 more variables: arr_delay <dbl>,
carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
> arrange(flights, year, month, day)
A tibble: 336,776 x 19
year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time
<int> <int> <int> <int> <int> <dbl> <int> <int>
1 2013 1 1 517 515 2 830 819
2 2013 1 1 533 529 4 850 830
3 2013 1 1 542 540 2 923 850
4 2013 1 1 544 545 -1 1004 1022
5 2013 1 1 554 600 -6 812 837
6 2013 1 1 554 558 -4 740 728
7 2013 1 1 555 600 -5 913 854
8 2013 1 1 557 600 -3 709 723
9 2013 1 1 557 600 -3 838 846
10 2013 1 1 558 600 -2 753 745
… with 336,766 more rows, and 11 more variables: arr_delay <dbl>,
carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
> arrange(flights, desc(dep_delay))
A tibble: 336,776 x 19
year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time
<int> <int> <int> <int> <int> <dbl> <int> <int>
1 2013 1 9 641 900 1301 1242 1530
2 2013 6 15 1432 1935 1137 1607 2120
3 2013 1 10 1121 1635 1126 1239 1810
4 2013 9 20 1139 1845 1014 1457 2210
5 2013 7 22 845 1600 1005 1044 1815
6 2013 4 10 1100 1900 960 1342 2211
7 2013 3 17 2321 810 911 135 1020
8 2013 6 27 959 1900 899 1236 2226
9 2013 7 22 2257 759 898 121 1026
10 2013 12 5 756 1700 896 1058 2020
… with 336,766 more rows, and 11 more variables: arr_delay <dbl>,
carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
> arrange(flights, year, month, day)
A tibble: 336,776 x 19
year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time
<int> <int> <int> <int> <int> <dbl> <int> <int>
1 2013 1 1 517 515 2 830 819
2 2013 1 1 533 529 4 850 830
3 2013 1 1 542 540 2 923 850
4 2013 1 1 544 545 -1 1004 1022
5 2013 1 1 554 600 -6 812 837
6 2013 1 1 554 558 -4 740 728
7 2013 1 1 555 600 -5 913 854
8 2013 1 1 557 600 -3 709 723
9 2013 1 1 557 600 -3 838 846
10 2013 1 1 558 600 -2 753 745
… with 336,766 more rows, and 11 more variables: arr_delay <dbl>,
carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
[3: ⏏] R R
69% Sunday, 09:25


> library(nycflights13)
> library(tidyverse)
── Attaching packages ──────────────────────────────────── tidyverse 1.3.0 ──
✔ ggplot2 3.2.1 ✔ purrr 0.3.3
✔ tibble 2.1.3 ✔ dplyr 0.8.3
✔ tidyr 1.0.2 ✔ stringr 1.4.0
✔ readr 1.3.1 ✔ forcats 0.4.0
── Conflicts ─────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::between() masks expss::between()
✖ dplyr::compute() masks expss::compute()
✖ dplyr::contains() masks tidyr::contains(), expss::contains()
✖ dplyr::filter() masks stats::filter()
✖ dplyr::first() masks expss::first()
✖ stringr::fixed() masks expss::fixed()
✖ purrr::keep() masks expss::keep()
✖ dplyr::lag() masks stats::lag()
✖ dplyr::last() masks expss::last()
✖ purrr::modify() masks expss::modify()
✖ purrr::modify_if() masks expss::modify_if()
✖ dplyr::na_if() masks expss::na_if()
✖ tidyr::nest() masks expss::nest()
✖ dplyr::recode() masks expss::recode()
✖ stringr::regex() masks expss::regex()
✖ purrr::transpose() masks expss::transpose()
✖ dplyr::vars() masks ggplot2::vars(), expss::vars()
✖ purrr::when() masks expss::when()
> flights
A tibble: 336,776 x 19
year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time
<int> <int> <int> <int> <int> <dbl> <int> <int>
1 2013 1 1 517 515 2 830 819
2 2013 1 1 533 529 4 850 830
3 2013 1 1 542 540 2 923 850
4 2013 1 1 544 545 -1 1004 1022
5 2013 1 1 554 600 -6 812 837
6 2013 1 1 554 558 -4 740 728
7 2013 1 1 555 600 -5 913 854
8 2013 1 1 557 600 -3 709 723
9 2013 1 1 557 600 -3 838 846
10 2013 1 1 558 600 -2 753 745
… with 336,766 more rows, and 11 more variables: arr_delay <dbl>,
carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
> flights_sml <- select(flights,
+ year:day,
+ ends_with("delay"),
+ distance,
+ air_time
+)
> mutate(flights_sml,
+ gain = dep_delay - arr_delay,
+ speed = distance / air_time * 60
+)
A tibble: 336,776 x 9
year month day dep_delay arr_delay distance air_time gain speed
<int> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 2013 1 1 2 11 1400 227 -9 370.
2 2013 1 1 4 20 1416 227 -16 374.
3 2013 1 1 2 33 1089 160 -31 408.
4 2013 1 1 -1 -18 1576 183 17 517.
5 2013 1 1 -6 -25 762 116 19 394.
6 2013 1 1 -4 12 719 150 -16 288.
7 2013 1 1 -5 19 1065 158 -24 404.
8 2013 1 1 -3 -14 229 53 11 259.
9 2013 1 1 -3 -8 944 140 5 405.
10 2013 1 1 -2 8 733 138 -10 319.
… with 336,766 more rows
> mutate(flights_sml,
+ gain = dep_delay - arr_delay,
+ hours = air_time / 60,
+ gain_per_hour = gain / hours
+)
A tibble: 336,776 x 10
year month day dep_delay arr_delay distance air_time gain hours
<int> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 2013 1 1 2 11 1400 227 -9 3.78
2 2013 1 1 4 20 1416 227 -16 3.78
3 2013 1 1 2 33 1089 160 -31 2.67
4 2013 1 1 -1 -18 1576 183 17 3.05
5 2013 1 1 -6 -25 762 116 19 1.93
6 2013 1 1 -4 12 719 150 -16 2.5
7 2013 1 1 -5 19 1065 158 -24 2.63
8 2013 1 1 -3 -14 229 53 11 0.883
9 2013 1 1 -3 -8 944 140 5 2.33
10 2013 1 1 -2 8 733 138 -10 2.3
… with 336,766 more rows, and 1 more variable: gain_per_hour <dbl>
[3: ⏏] R R
79% Sunday, 09:33

Conclusion: Thus, we have worked with large datasets.

Experiment no:7

Aim: EDA with R- Range, Mean, Variance, Median, Standard Deviation, Histogram, Box-Plot, Scatter-Plot.

Theory:

Exploratory Data Analysis:

Exploratory Data Analysis (**EDA**) is the process of analysing and visualizing the data to get a better understanding of the data and glean insight from it. There are various steps involved when doing EDA but the following are the common steps that a data analyst can take when performing EDA:

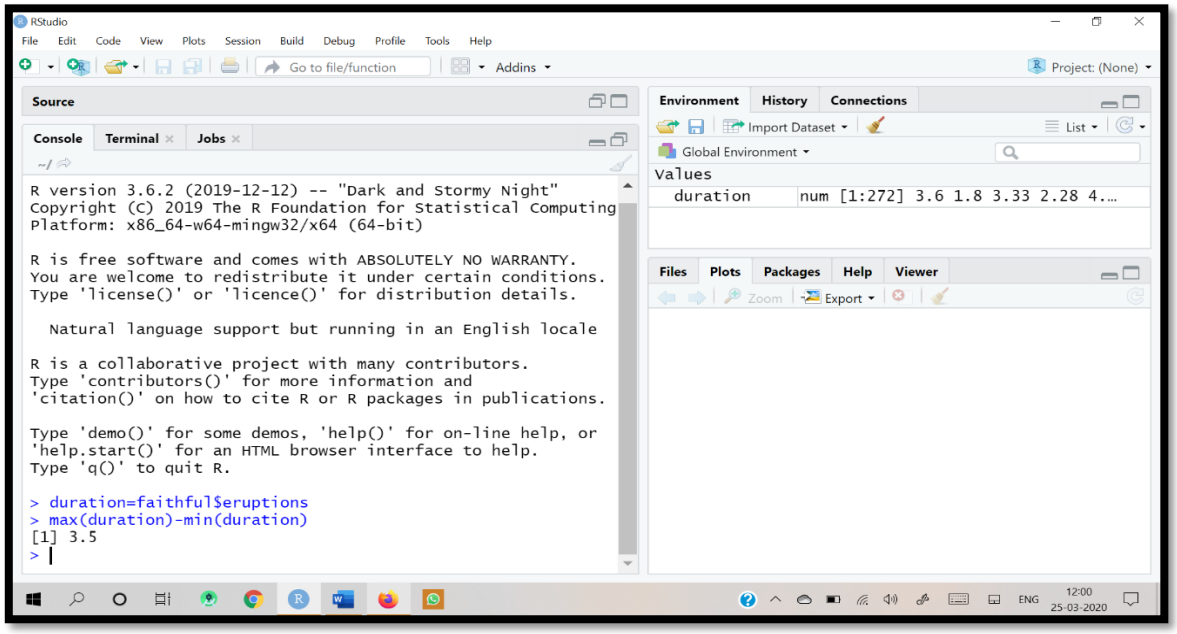
1. Import the data
2. Clean the data
3. Process the data
4. Visualize the data

A. Range in R:

The range of an observation variable is the difference of its largest and smallest data values. It is a measure of how far apart the entire data spreads in value.

Range = Largest Value – Smallest Value

Output:



B. Mean in R:

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 this in R.

Syntax:

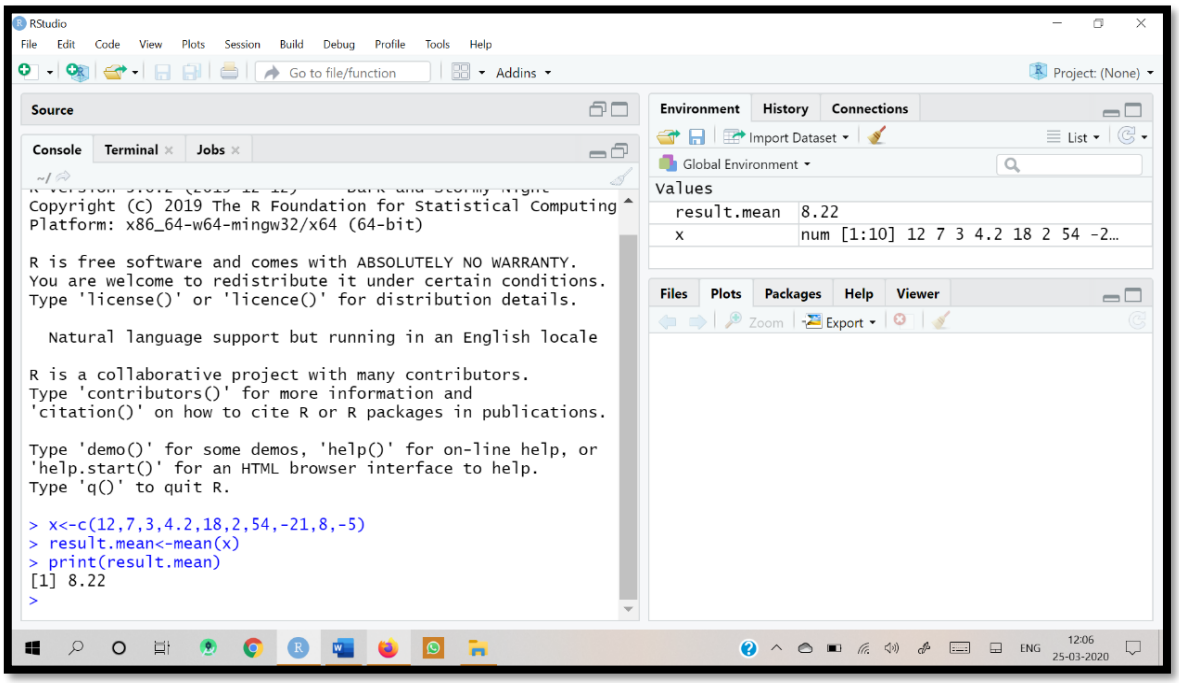
The basic syntax for calculating mean in R is −

mean (x, trim = 0, na.rm = FALSE, ...)

Following is the description of the parameters used −

- **x** is the input vector.
- **trim** is used to drop some observations from both end of the sorted vector.
- **na.rm** is used to remove the missing values from the input vector.

Output:

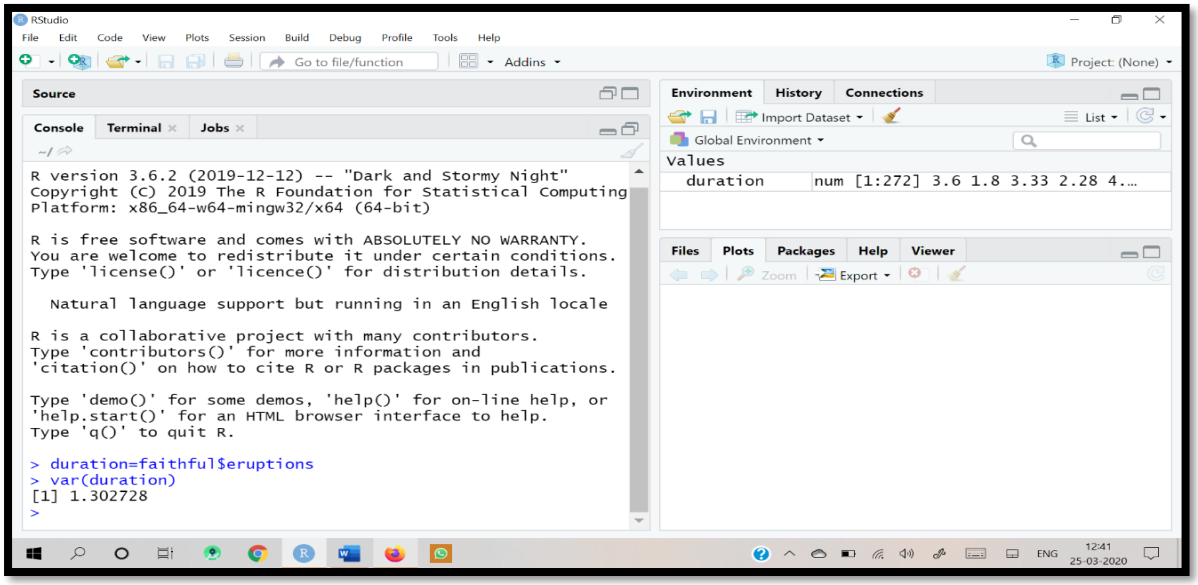


C. Variance in R:

The variance is a numerical measure of how the data values is dispersed around the mean. In particular, the sample variance is defined as:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

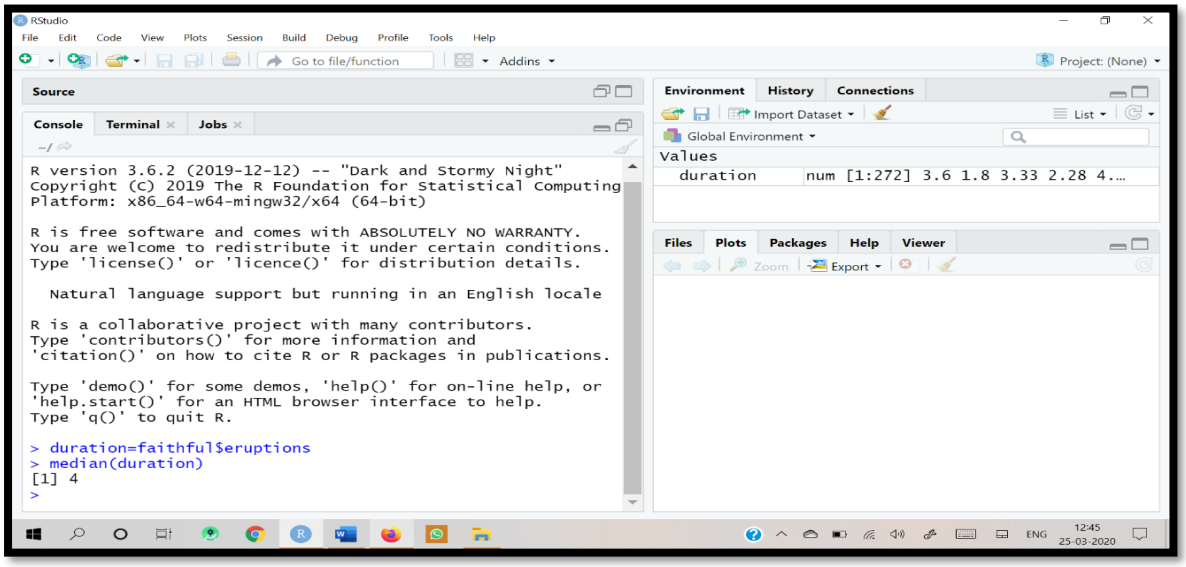
Output:



D. Median in R:

The median of an observation variable is the value at the middle when the data is sorted in ascending order. It is an ordinal measure of the central location of the data values.

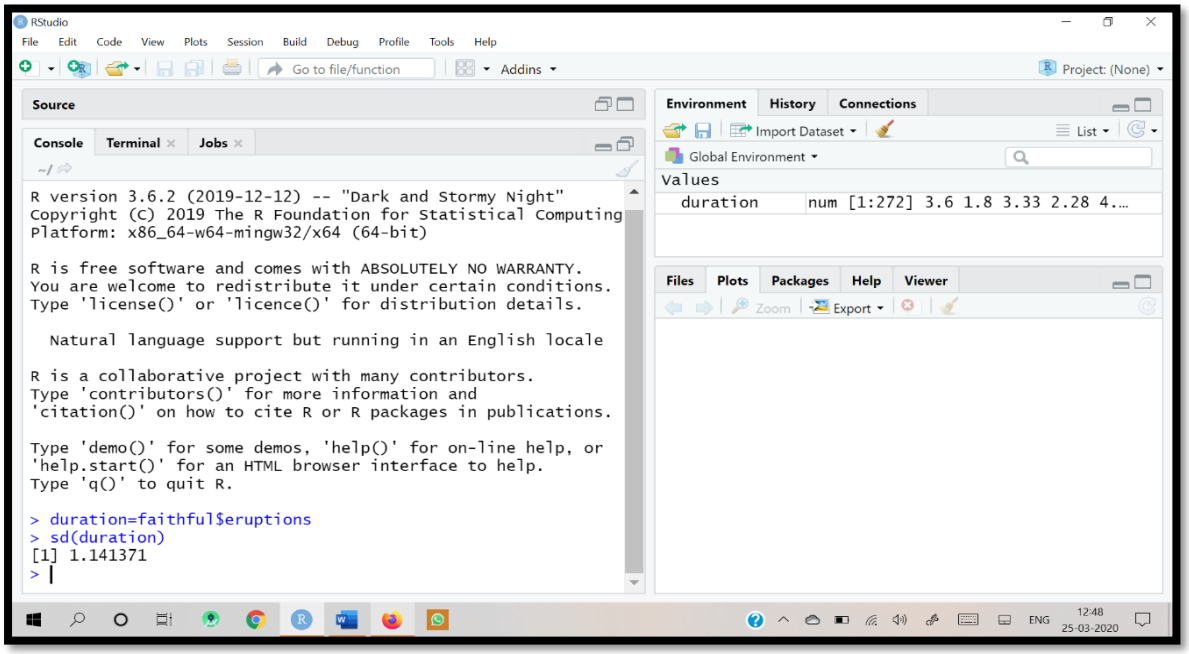
Output:



E. Standard Deviation in R:

The standard deviation of an observation variable is the square root of its variance.

Output:



F. Histogram in R:

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. Each bar in histogram represents the height of the number of values present in that range. R creates histogram using **hist()** function. This function takes a vector as an input and uses some more parameters to plot histograms.

Syntax:

The basic syntax for creating a histogram using R is –

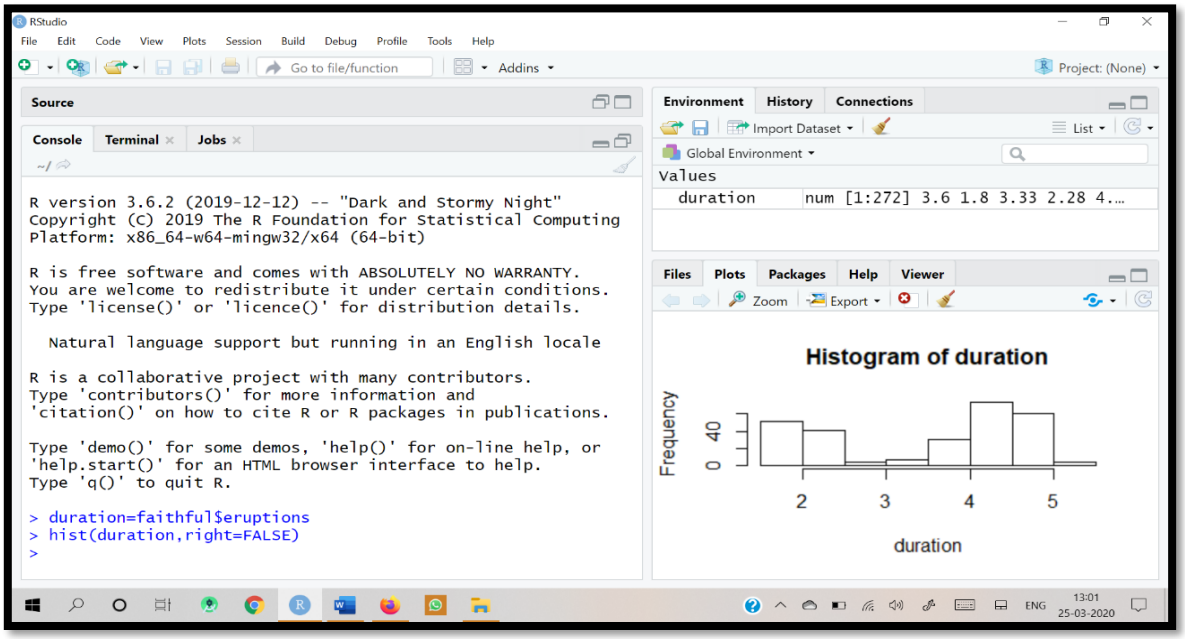
hist (v, main, xlab, xlim, ylim, breaks, col, border)

Following is the description of the parameters used –

- **v** is a vector containing numeric values used in histogram.
- **main** indicates title of the chart.

- **col** is used to set color of the bars.
- **border** is used to set border color of each bar.
- **xlab** is used to give description of x-axis.
- **xlim** is used to specify the range of values on the x-axis.
- **ylim** is used to specify the range of values on the y-axis.
- **breaks** are used to mention the width of each bar.

Output:



G. Box-plot in R:

Boxplots are a measure of how well distributed is the data in a data set. It divides the data set into three quartiles. This graph represents the minimum, maximum, median, first quartile and third quartile in the data set. It is also useful in comparing the distribution of data across data sets by drawing boxplots for each of them. Boxplots are created in R by using the **boxplot()** function.

Syntax:

The basic syntax to create a boxplot in R is –

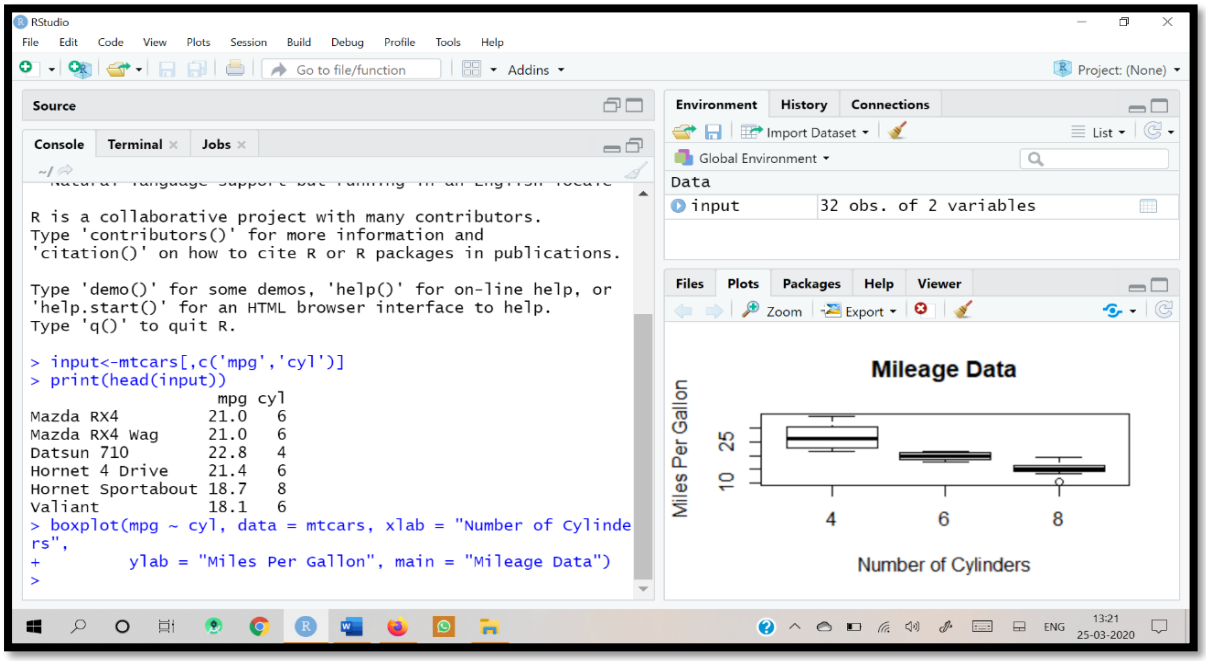
boxplot (x, data, notch, varwidth, names, main)

Following is the description of the parameters used –

- **x** is a vector or a formula.

- **data** is the data frame.
- **notch** is a logical value. Set as TRUE to draw a notch.
- **varwidth** is a logical value. Set as true to draw width of the box proportionate to the sample size.
- **names** are the group labels which will be printed under each boxplot.
- **main** is used to give a title to the graph.

Output:



H. Scatter-plot in R:

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. The simple scatterplot is created using the **plot ()** function.

Syntax:

The basic syntax for creating scatterplot in R is −

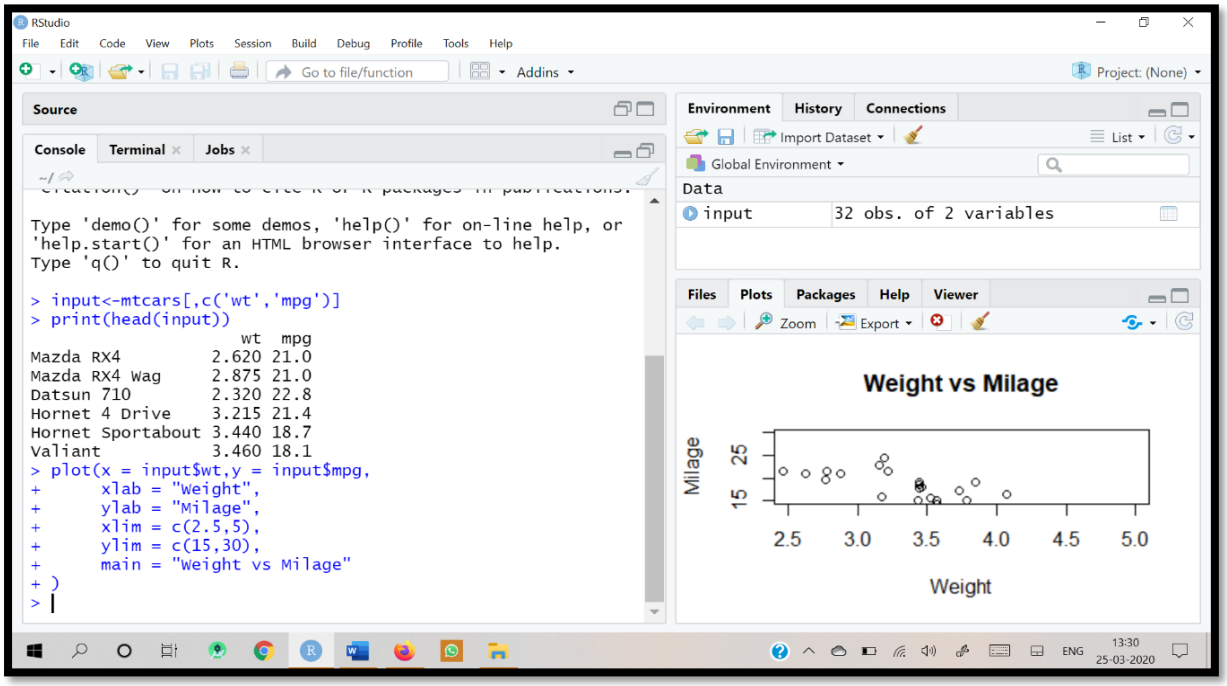
plot (x, y, main, xlab, ylab, xlim, ylim, axes)

Following is the description of the parameters used −

- **x** is the data set whose values are the horizontal coordinates.
- **y** is the data set whose values are the vertical coordinates.

- **main** is the tile of the graph.
- **xlab** is the label in the horizontal axis.
- **ylab** is the label in the vertical axis.
- **xlim** is the limits of the values of x used for plotting.
- **ylim** is the limits of the values of y used for plotting.
- **axes** indicate whether both axes should be drawn on the plot.

Output:



Conclusion:

Thus, we have studied exploratory data analysis in R.

Experiment no:8

Aim: Basics of Advanced graphics in R.

Theory:

Graphics are important for conveying important features of the data. R includes at least three graphical systems, the standard graphics package, the lattice package for Trellis graphs. R has good graphical capabilities but there are some alternatives like gun plot.

R provides the usual range of standard statistical plots, including scatterplots, boxplots, histograms, bar plots, pie charts, and basic 3D plots.

They can be used to examine

• Marginal distributions

• Relationships between variables

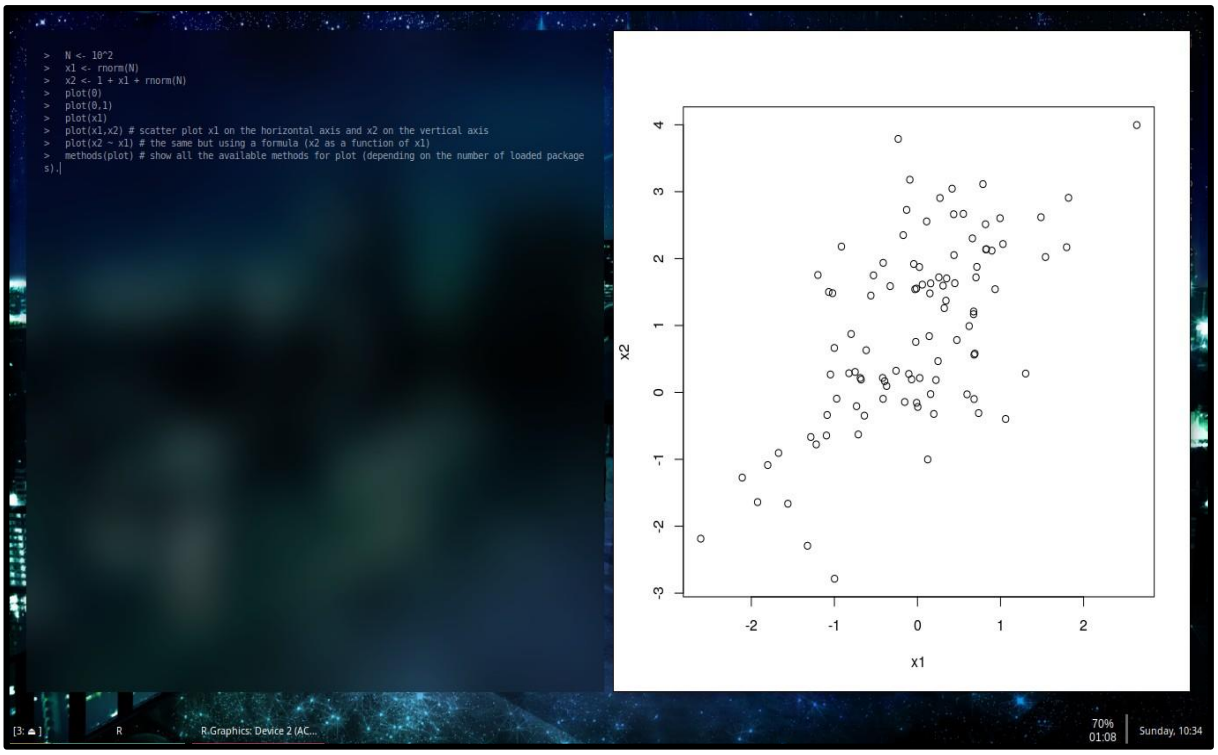
• Summary of very large data

Important complement to many statistical and computational techniques.

Graphics in R:

- plot () is the main function for graphics. The arguments can be a single point such as 0 or c (.3,.7), a single vector, a pair of vectors or many other R objects.
- par () is another important function which defines the default settings for plots.
- There are many other plot functions which are specific to some tasks such as hist (), boxplot (), etc. Most of them take the same arguments as the plot () function.

```
N <- 10^2
x1 <- rnorm(N)
x2 <- 1 + x1 + rnorm(N)
plot (0)
plot (0,1)
plot(x1)
plot (x1, x2) # scatter plot x1 on the horizontal axis and x2 on the vertical axis
plot (x2 ~ x1) # the same but using a formula (x2 as a function of x1)
```



R Histograms

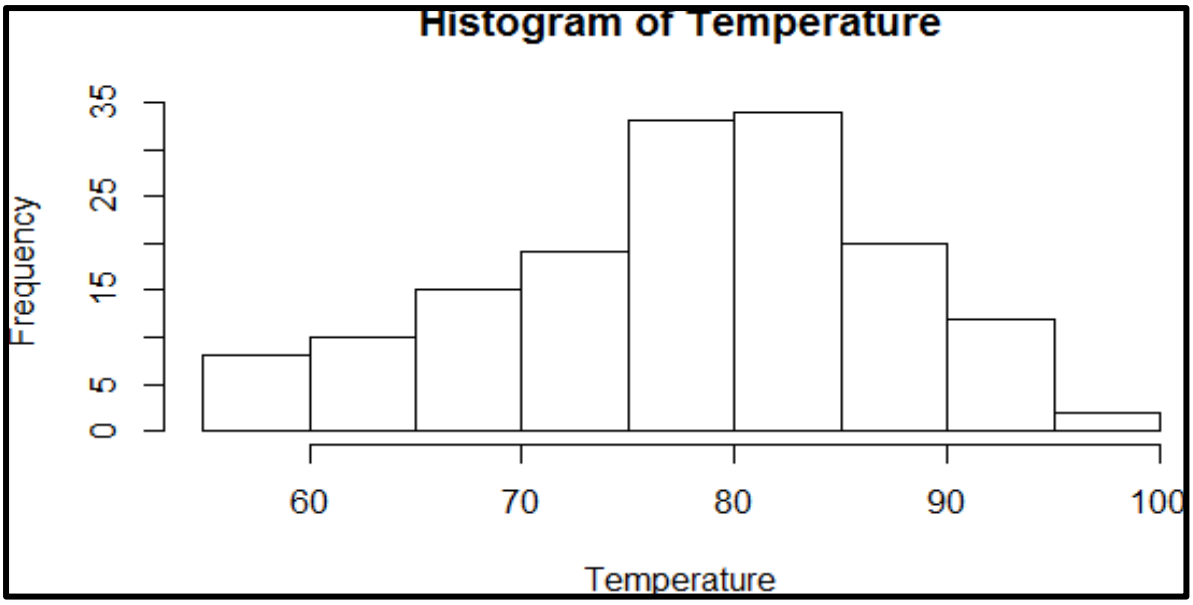
In this article, you'll learn to use hist () function to create histograms in R programming with the help of numerous examples.

Histogram can be created using the hist () function in R programming language. This function takes in a vector of values for which the histogram is plotted.

Let us use the built-in dataset air quality which has Daily air quality measurements in New York, May to September 1973.-R documentation.

str(airquality)
'data. frame':153 obs. of 6 variables: $ Ozone: int 41 36 12 18 NA 28 23 19 8 NA ...$
Solar.R: int 190 118 149 313 NA NA 299 99 19 194 ...$
Wind: num 7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...$
Temp: int 67 72 74 62 56 66 65 59 61 69 ...$
Month: int 5 5 5 5 5 5 5 5 5 5 ...$
Day: int 1 2 3 4 5 6 7 8 9 10 ...

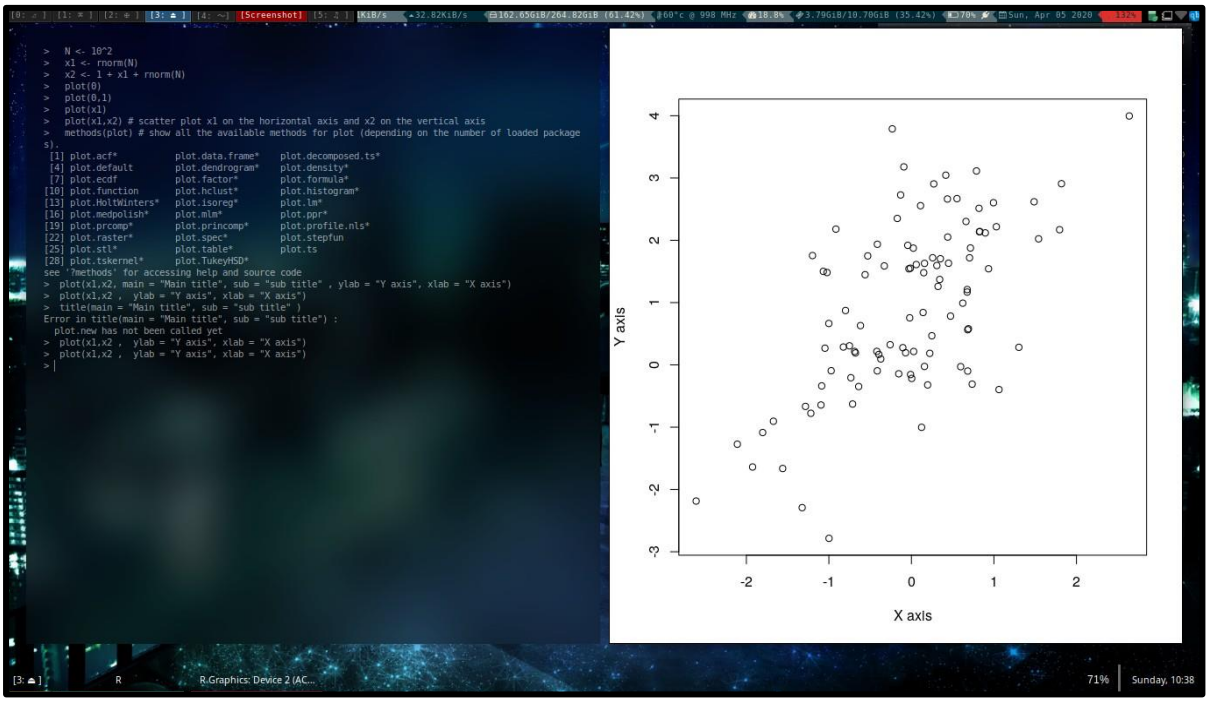
Temperature <- airquality$Temphist (Temperature)



Titles, legends and annotations

Titles main gives the main title, sub the subtitle. They can be passed as argument of the plot () function or using the title () function. xlab the name of the x axis and ylab the name of the y axis.

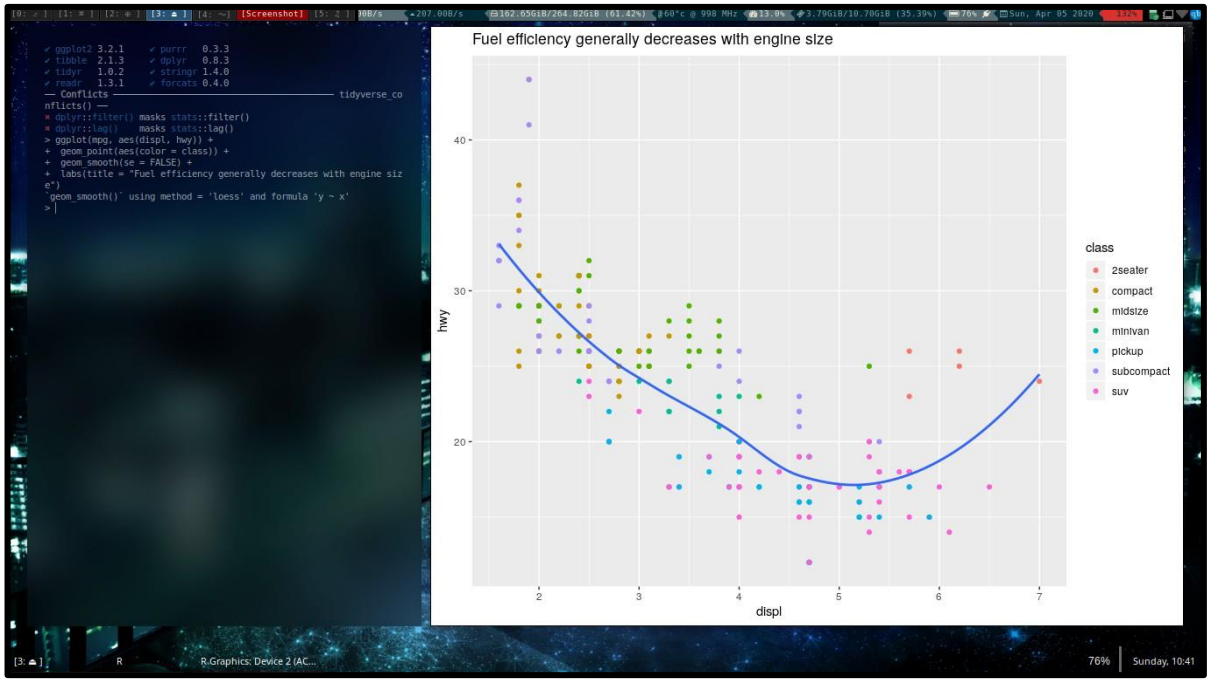
Plot (x1, x2, main = "Main title", sub = "sub title", ylab = "Y axis", xlab = "X axis")
plot (x1, x2, ylab = "Y axis", xlab = "X axis")
title (main = "Main title", sub = "sub title")



Label

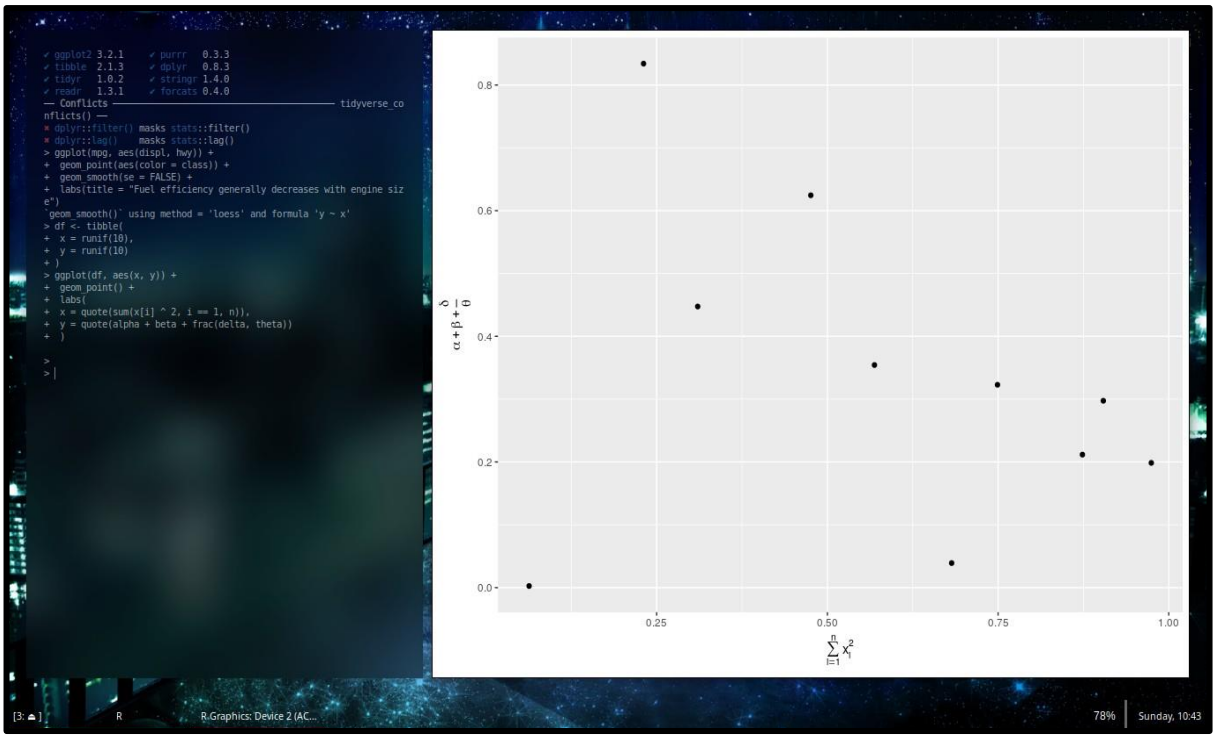
The easiest place to start when turning an exploratory graphic into an expository graphic is with good labels. You add labels with the labs () function. This example adds a plot title:

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(color = class)) +
  geom_smooth(se = FALSE) +
  labs (title = "Fuel efficiency generally decreases with engine size")
```



It's possible to use mathematical equations instead of text strings. Just switch "" out for quote () and read about the available options in? plotmath:

```
df <- tibble (
    x = runif (10),
    y = runif (10)
)
ggplot (df, aes (x, y)) +
  geom_point () +
  labs (
   x = quote(sum(x[i] ^ 2, i == 1, n)),
   y = quote (alpha + beta + frac (delta, theta))
)
```

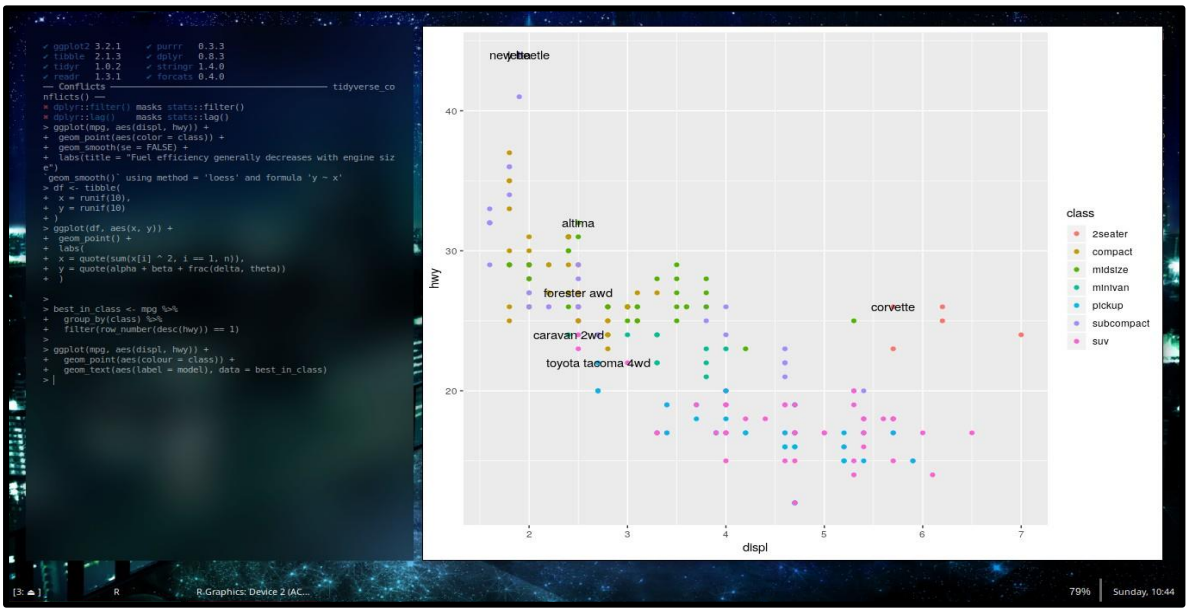


Annotations

In addition to labelling major components of your plot, it's often useful to label individual observations or groups of observations. The first tool you have at your disposal is geom_text (). geom_text () is similar to geom_point (), but it has an additional aesthetic: label. This makes it possible to add textual labels to your plots.

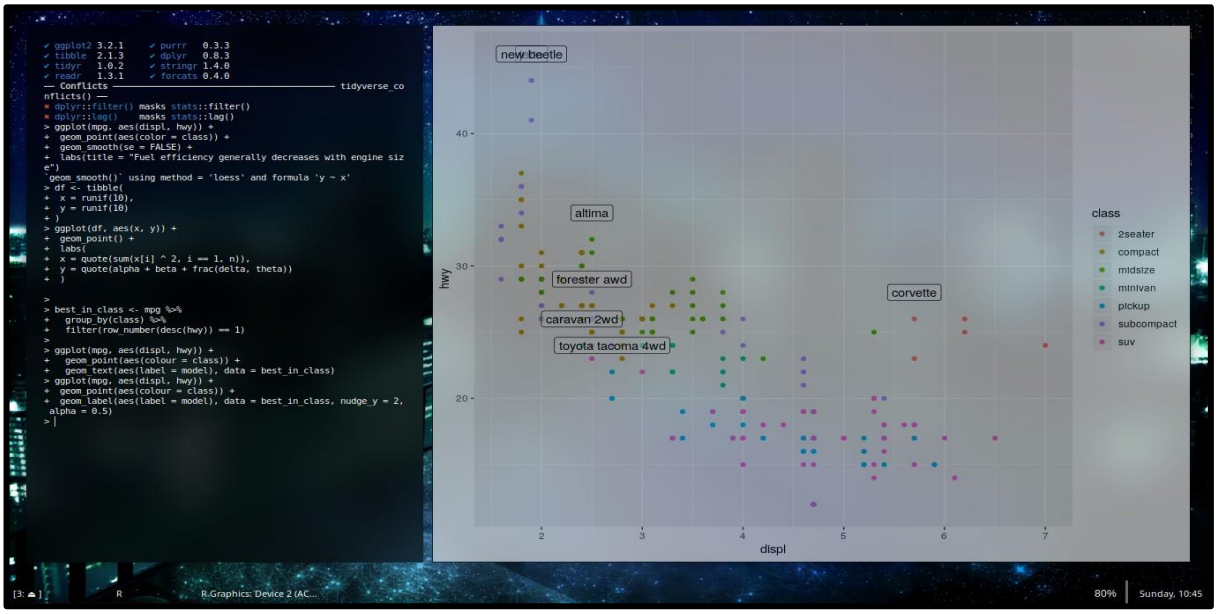
There are two possible sources of labels. First, you might have a tibble that provides labels. The plot below isn't terribly useful, but it illustrates a useful approach: pull out the most efficient car in each class with dplyr, and then label it on the plot:

```
best_in_class <- mpg %>%
    group_by(class) %>%
    filter(row_number(desc(hwy)) == 1)

ggplot (mpg, aes(displ, hwy)) +
   geom_point (aes(colour = class)) +
   geom_text (aes(label = model), data = best_in_class)
```



This is hard to read because the labels overlap with each other, and with the points. We can make things a little better by switching to geom_label () which draws a rectangle behind the text. We also use the nudge_y parameter to move the labels slightly above the corresponding points:

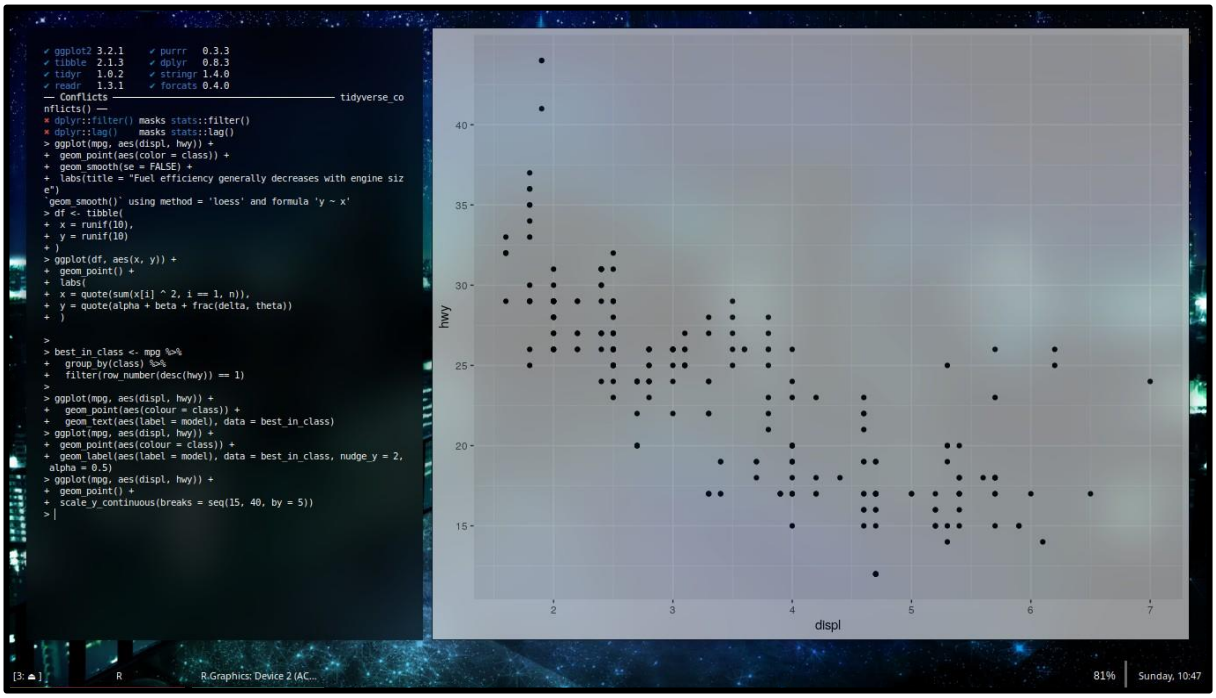
```
ggplot (mpg, aes(displ, hwy)) +
  geom_point (aes(colour = class)) +
  geom_label (aes(label = model), data = best_in_class, nudge_y = 2, alpha = 0.5)
```



Axis ticks and legend keys

There are two primary arguments that affect the appearance of the ticks on the axes and the keys on the legend: breaks and labels. Breaks controls the position of the ticks, or the values associated with the keys. Labels controls the text label associated with each tick/key. The most common use of breaks is to override the default choice:

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  scale_y_continuous (breaks = seq (15, 40, by = 5))
```

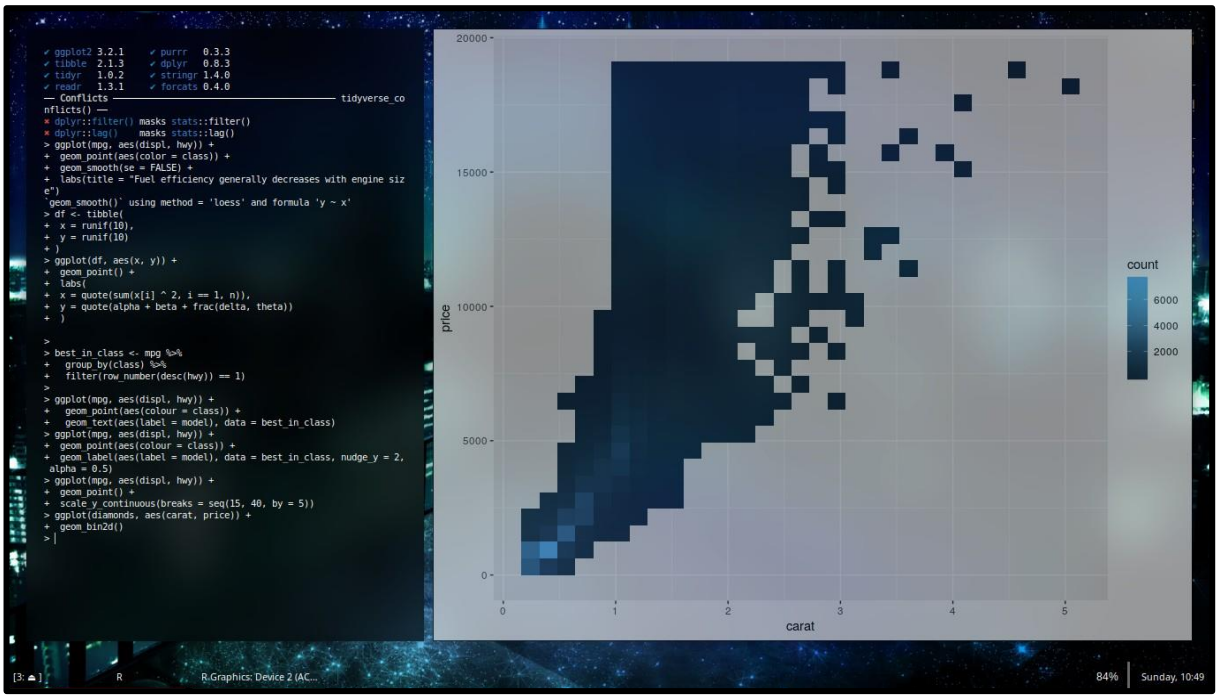


Replacing a scale

Instead of just tweaking the details a little, you can instead replace the scale altogether. There are two types of scales you're mostly likely to want to switch out: continuous position scales and colour scales. Fortunately, the same principles apply to all the other aesthetics, so once you've mastered position and colour, you'll be able to quickly pick up other scale replacements.

It's very useful to plot transformations of your variable. For example, as we've seen in diamond prices it's easier to see the precise relationship between carat and price if we log transform them:

```
ggplot (diamonds, aes (carat, price)) +
   geom_bin2d ()
```



Saving your plots

There are two main ways to get your plots out of R and into your final write-up: ggsave () and knitr. ggsave () will save the most recent plot to disk:

```
ggplot (mpg, aes (displ, hwy)) + geom_point ()
```

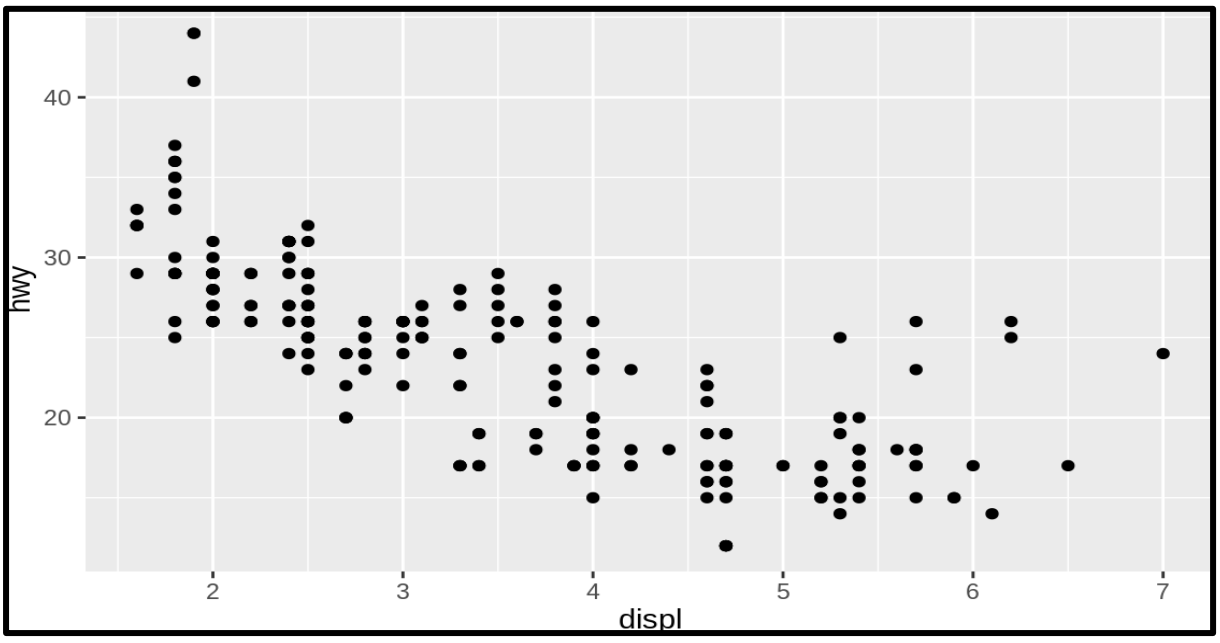
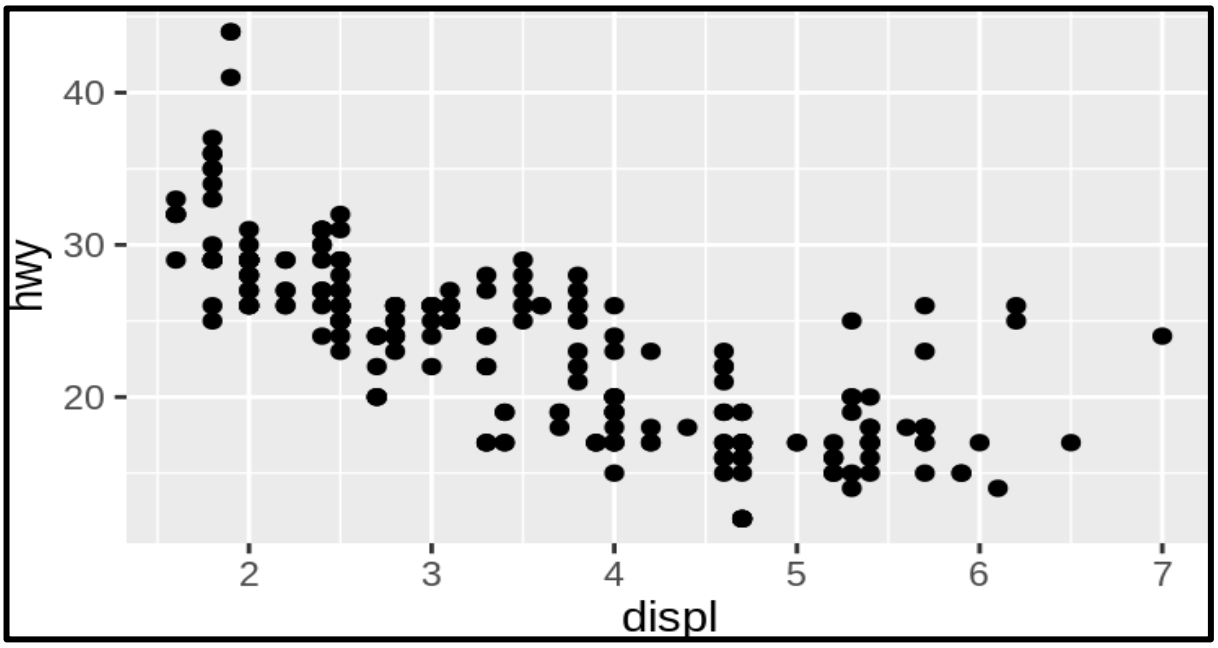
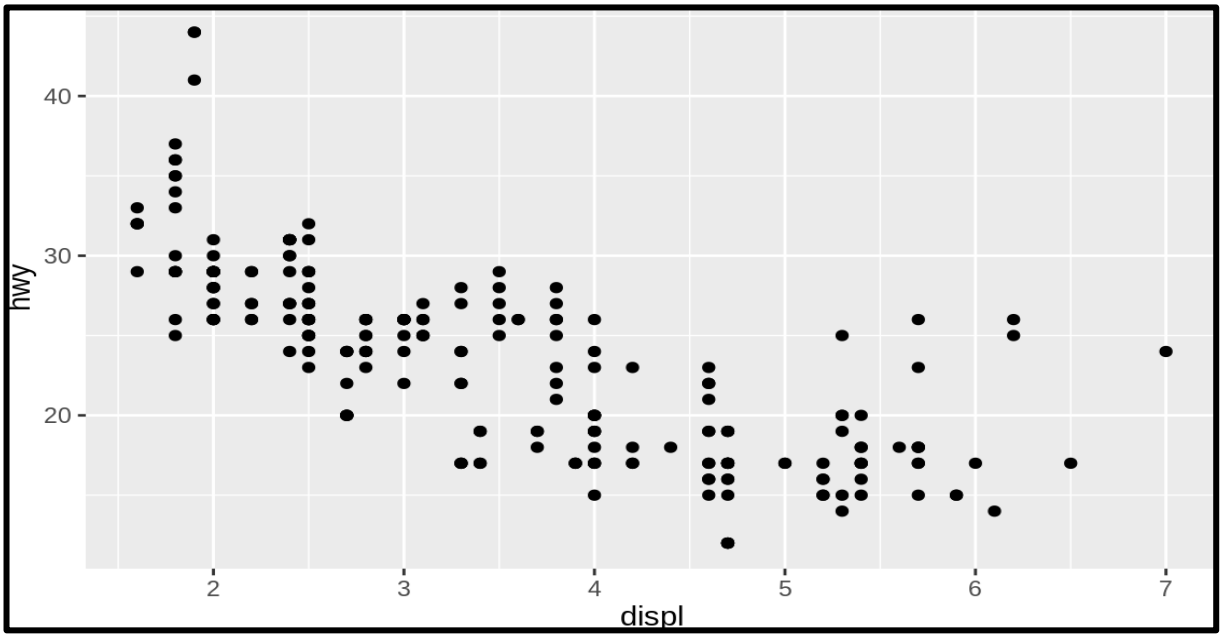
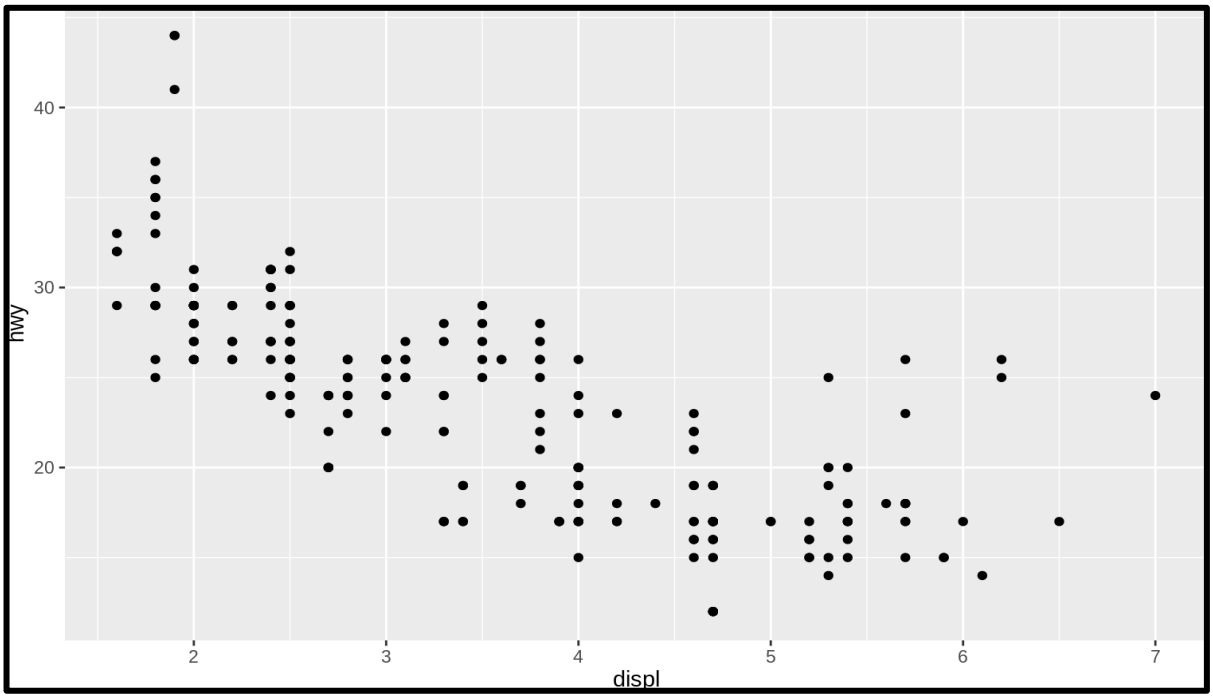


Figure sizing

The biggest challenge of graphics in R Markdown is getting your figures the right size and shape. There are five main options that control figure sizing: fig. width, fig. height, fig.asp, out. width and out. height. Image sizing is challenging because there are two sizes (the size of the figure created by R and the size at which it is inserted in the output document), and multiple ways of specifying the size (i.e., height, width, and aspect ratio: pick two of three).







Conclusion: Thus, we have studied basics of advanced graphics in R.

Experiment no:9

Aim: Basic of correlation, simple and multiple regression in R.

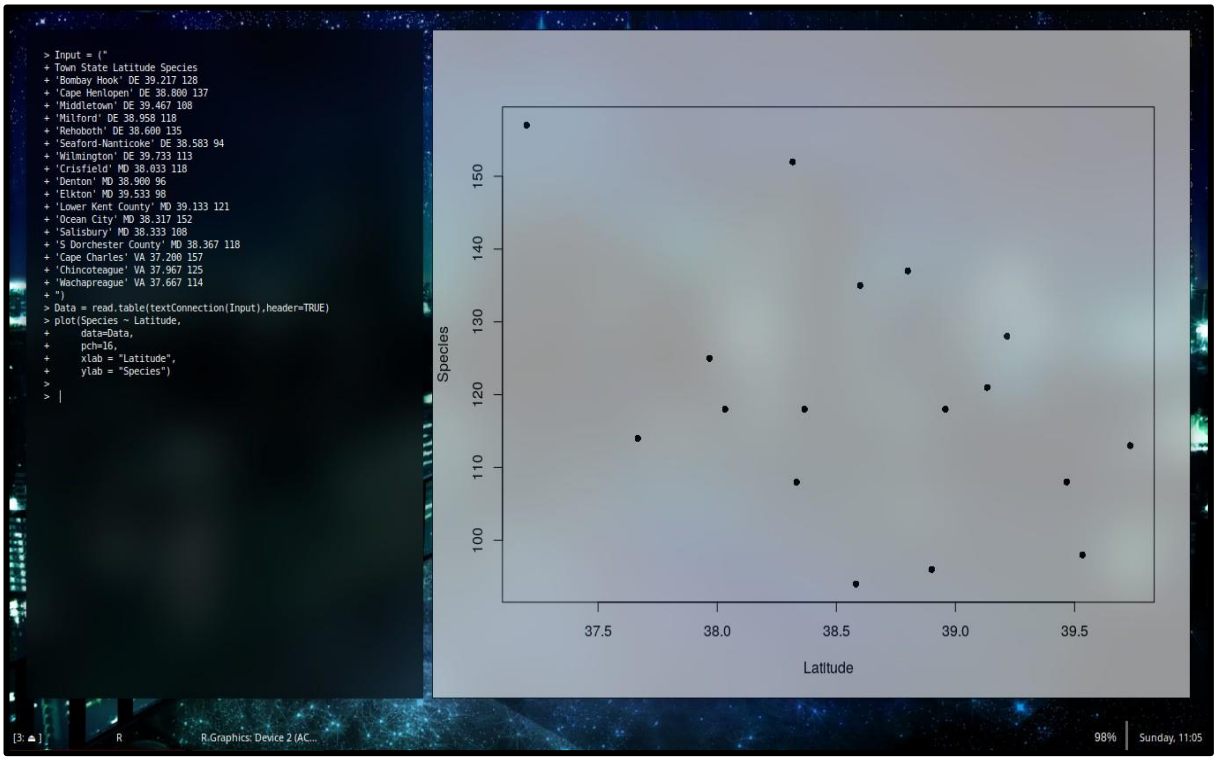
Theory:

Correlation and linear regression each explore the relationship between two quantitative variables. Both are very common analyses.

Correlation determines if one variable varies systematically as another variable changes. It does not specify that one variable is the dependent variable and the other is the independent variable. Often, it is useful to look at which variables are correlated to others in a data set, and it is especially useful to see which variables correlate to a particular variable of interest.

So as to generate the simple plot for the data:

```
plot(Species ~ Latitude,
     data=Data,
     pch=16,
     xlab = "Latitude",
     ylab = "Species")
```



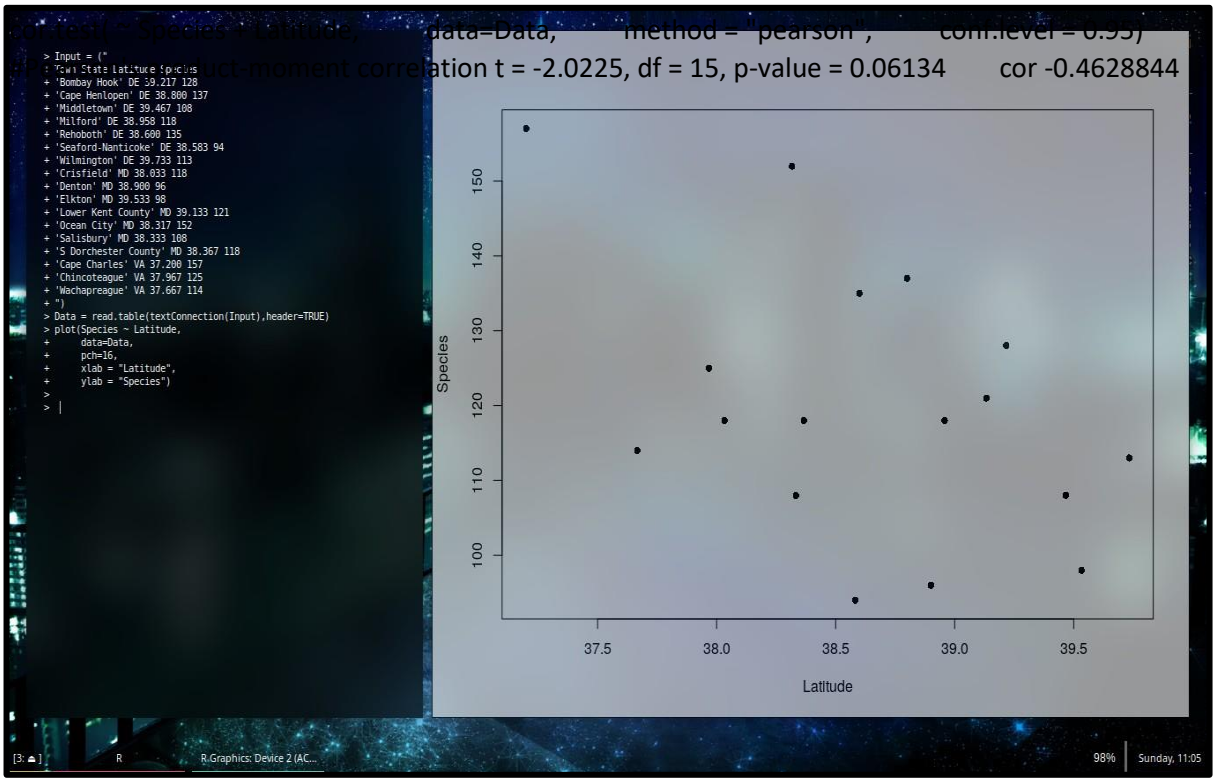
Correlation

Correlation can be performed with the cor.test function in the native stats package. It can perform Pearson, Kendall, and Spearman correlation procedures. Methods for multiple correlation of several variables simultaneously are discussed in the Multiple regression chapter.

Pearson correlation

Pearson correlation is the most common form of correlation. It is a parametric test, and assumes that the data are linearly related and that the residuals are normally distributed.

```
cor.test( ~ Species + Latitude,
    data=Data,
    method = "pearson",
    conf.level = 0.95)
#Pearson's product-moment correlation
t = -2.0225, df = 15, p-value = 0.06134
cor -0.4628844
```



Kendall correlation

Kendall rank correlation is a non-parametric test that does not assume a distribution of the data or that the data are linearly related. It ranks the data to determine the degree of correlation.

```
cor.test (~ Species + Latitude,
    data=Data,
    method = "kendall",
    continuity = FALSE,
    conf. level = 0.95)
Kendall's rank correlation tau
z = -1.3234, p-value = 0.1857
tau -0.2388326
```

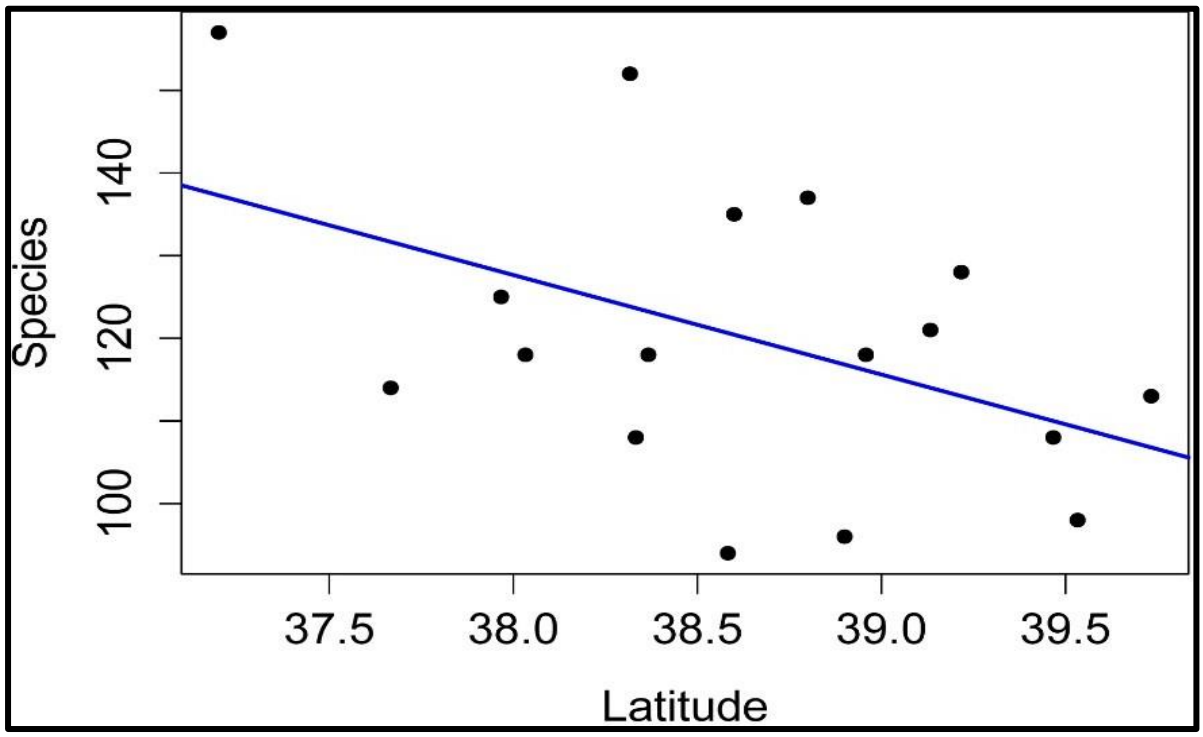
Spearman correlation

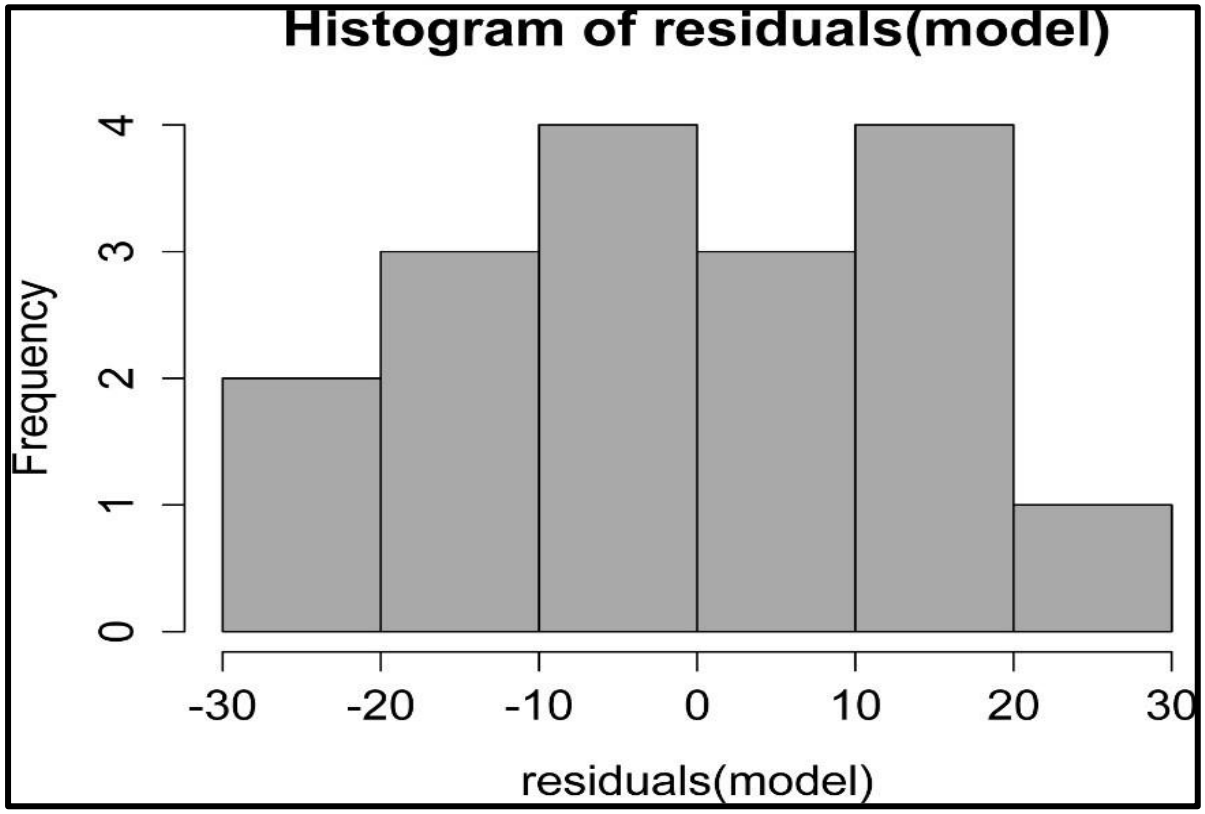
Spearman rank correlation is a non-parametric test that does not assume a distribution of the data or that the data are linearly related. It ranks the data to determine the degree of correlation, and is appropriate for ordinal measurements.

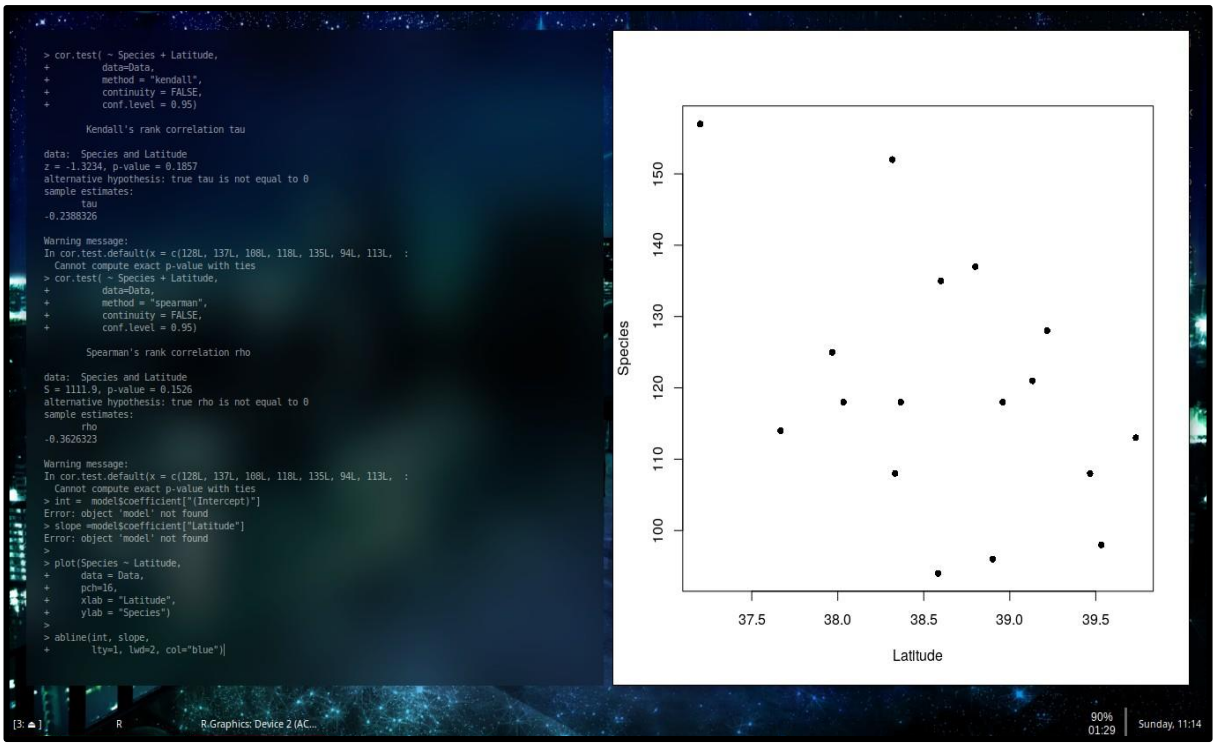
```
cor.test (~ Species + Latitude,
   data=Data,
   method = "spearman",
   continuity = FALSE,
   conf. level = 0.95)

Spearman's rank correlation rho
S = 1111.908, p-value = 0.1526
  rho -0.3626323
```

Linear regression

Linear regression can be performed with the lm function in the native stats package. A robust regression can be performed with the lmrob function in the robust base package.


Species
100
120
140
37.5
38.0
38.5
39.0
39.5
Latitude


Histogram of residuals(model)
Frequency
0
1
2
3
4
-30
-20
-10
0
10
20
30
residuals(model)



Conclusion: Thus, we have successfully studied Basics of correlation and regression in R.

Experiment no:10

Aim: Clustering in R: K-means

Theory:

K Means Clustering is an unsupervised learning algorithm that tries to cluster data based on their similarity. Unsupervised learning means that there is no outcome to be predicted, and the algorithm just tries to find patterns in the data. In k means clustering, we have the specify the number of clusters we want the data to be grouped into. The algorithm randomly assigns each observation to a cluster, and finds the centroid of each cluster. Then, the algorithm iterates through two steps:

- Reassign data points to the cluster whose centroid is closest.
- Calculate new centroid of each cluster.

These two steps are repeated till the within cluster variation cannot be reduced any further. The within cluster variation is calculated as the sum of the Euclidean distance between the data points and their respective cluster centroids.

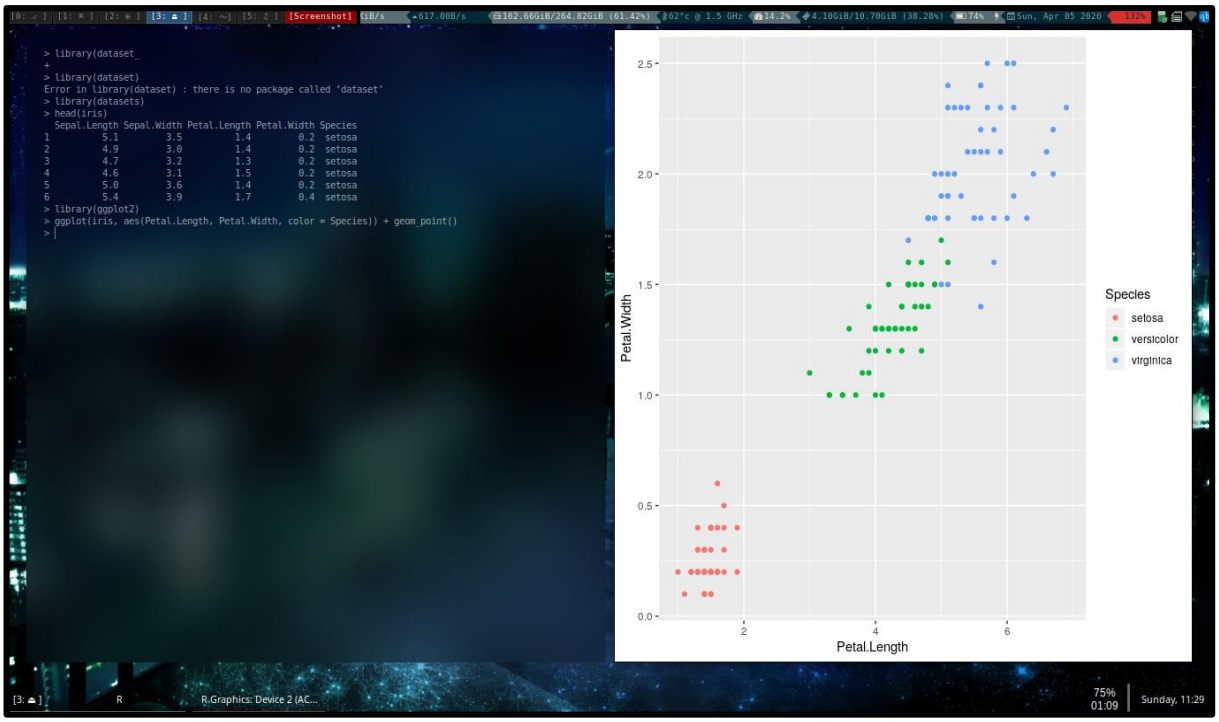
Exploring the data

The iris dataset contains data about sepal length, sepal width, petal length, and petal width of flowers of different species.

Let us see what it looks like:

```
library(datasets)
head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

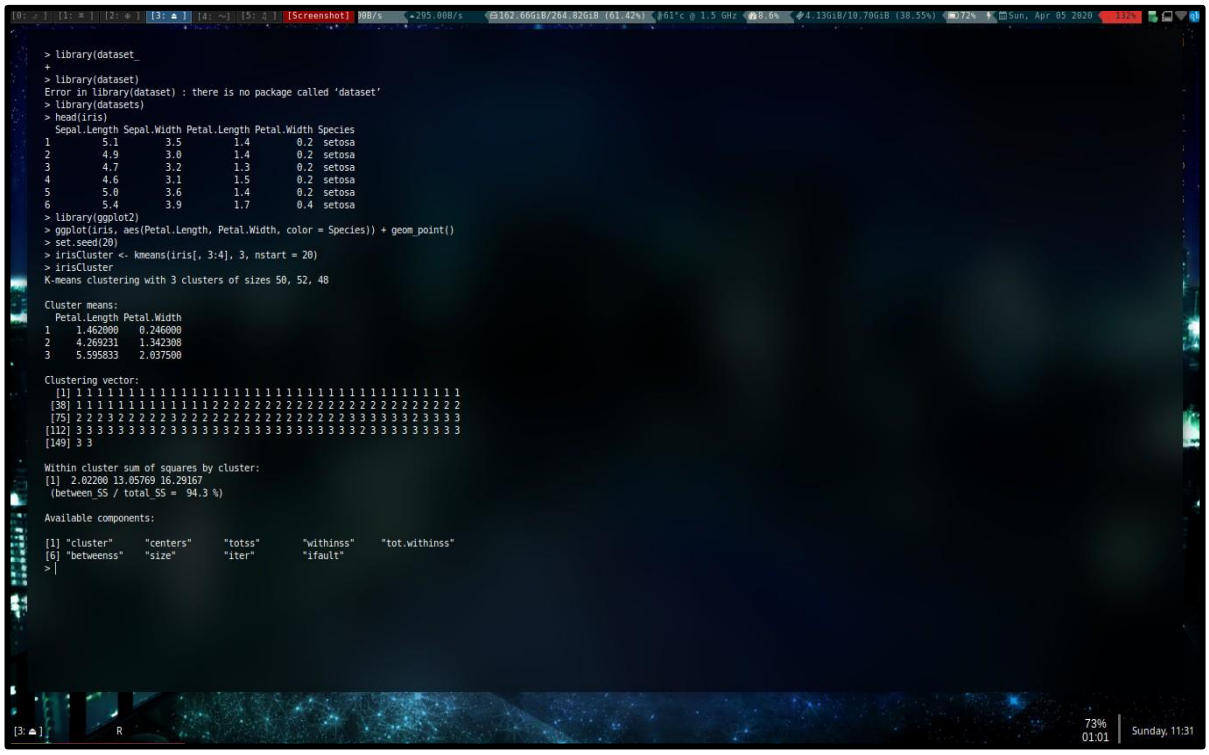
```
library(ggplot2)
ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()
```



Since we have seen the data, now we shall start and create a cluster of them.

```
set.seed(20)
irisCluster <- kmeans(iris[, 3:4], 3, nstart = 20)
irisCluster
K-means clustering with 3 clusters of sizes 46, 54, 50

Cluster means:
  Petal.Length Petal.Width
1     5.626087    2.047826
2     4.292593    1.359259
3     1.462000    0.246000

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [35] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [69] 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1
[103] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 2 2 1 1 1 1 1 1 1 1 1 1
[137] 1 1 2 1 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1] 15.16348 14.22741  2.02200
 (between_SS / total_SS =  94.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
```

Since we know that there are 3 species involved, we ask the algorithm to group the data into 3 clusters, and since the starting assignments are random, we specify nstart = 20 . This means that R will try 20 different random starting assignments and then select the one with the lowest within cluster variation. We can see the cluster centroids, the clusters that each data point was assigned to, and the within cluster variation.



Let us compare the clusters with the species.

```
table(irisCluster$cluster, iris$Species)
    setosa versicolor virginica
  1      0          2        44
  2      0         48         6
  3     50          0         0
```

As we can see, the data belonging to the setosa species got grouped into cluster 3, versicolor into cluster 2, and virginica into cluster 1. The algorithm wrongly classified two data points belonging to versicolor and six data points belonging to virginica.

We can also plot the data to see the clusters:

```
irisCluster$cluster <- as.factor(irisCluster$cluster)
ggplot(iris, aes(Petal.Length, Petal.Width, color = iris$cluster)) +
geom_point()
```

<u>Conclusion:</u> Thus, we have successfully studied K-means clustering on iris dataset in R.

Experiment no:11

Aim: Classification on large and noisy dataset with R - logistic regression and naive Bayes.

Theory:

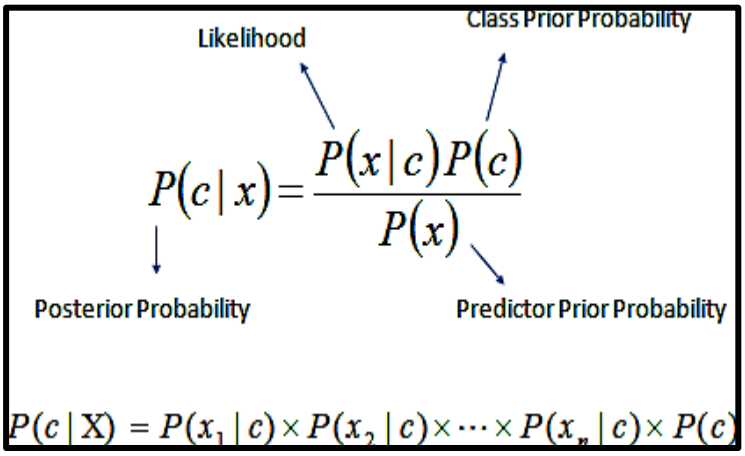
Logic Regression:

Logistic regression is yet another technique borrowed by machine learning from the field of statistics. It's a powerful statistical way of modelling a binomial outcome with one or more explanatory variables. It measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution.

Logistic regression is an instance of classification technique that you can use to predict a qualitative response. More specifically, logistic regression models the probability that belongs to a particular category.

That means that, if you are trying to do gender classification, where the response falls into one of the two categories, male or female, you'll use logistic regression models to estimate the probability that belongs to a particular category.

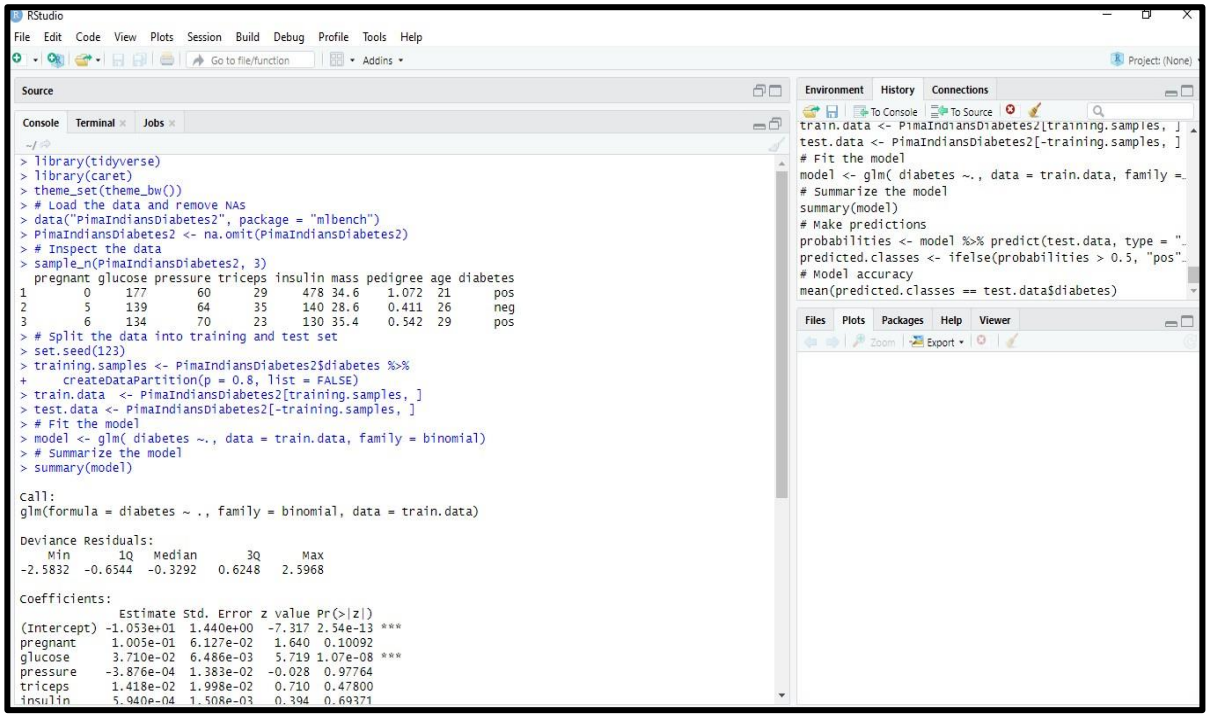
Naïve Bayes Classifier

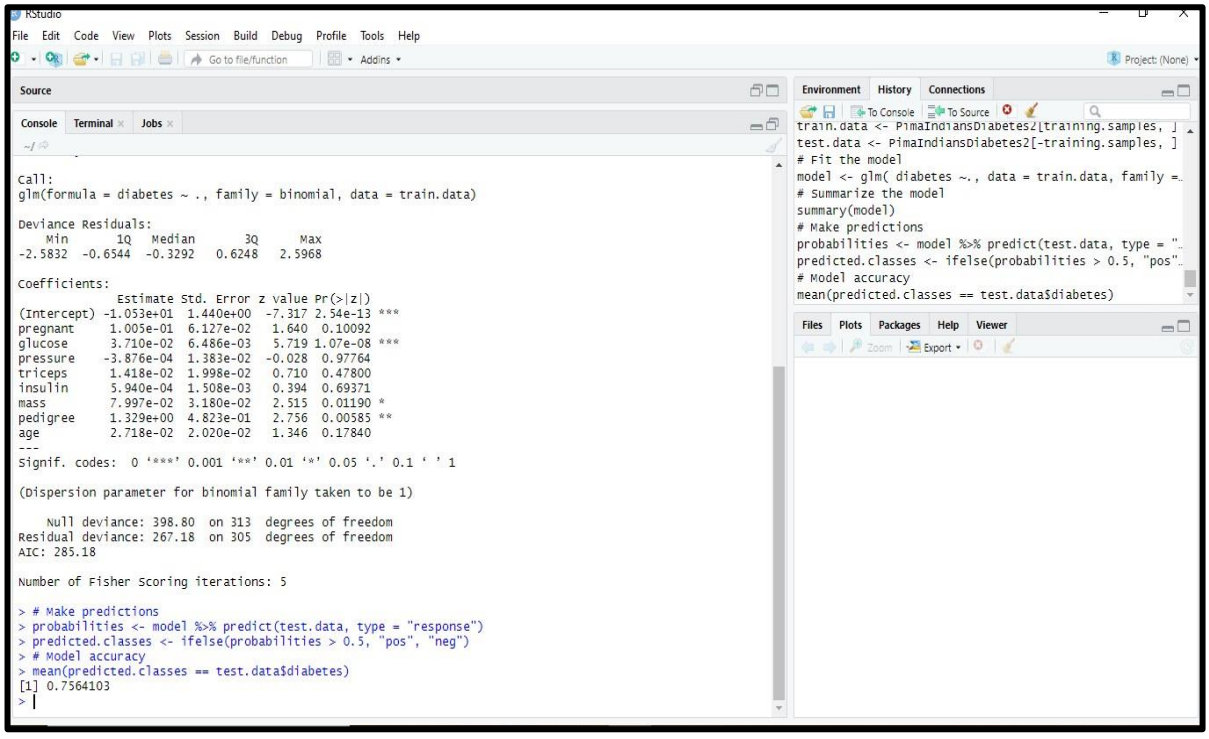


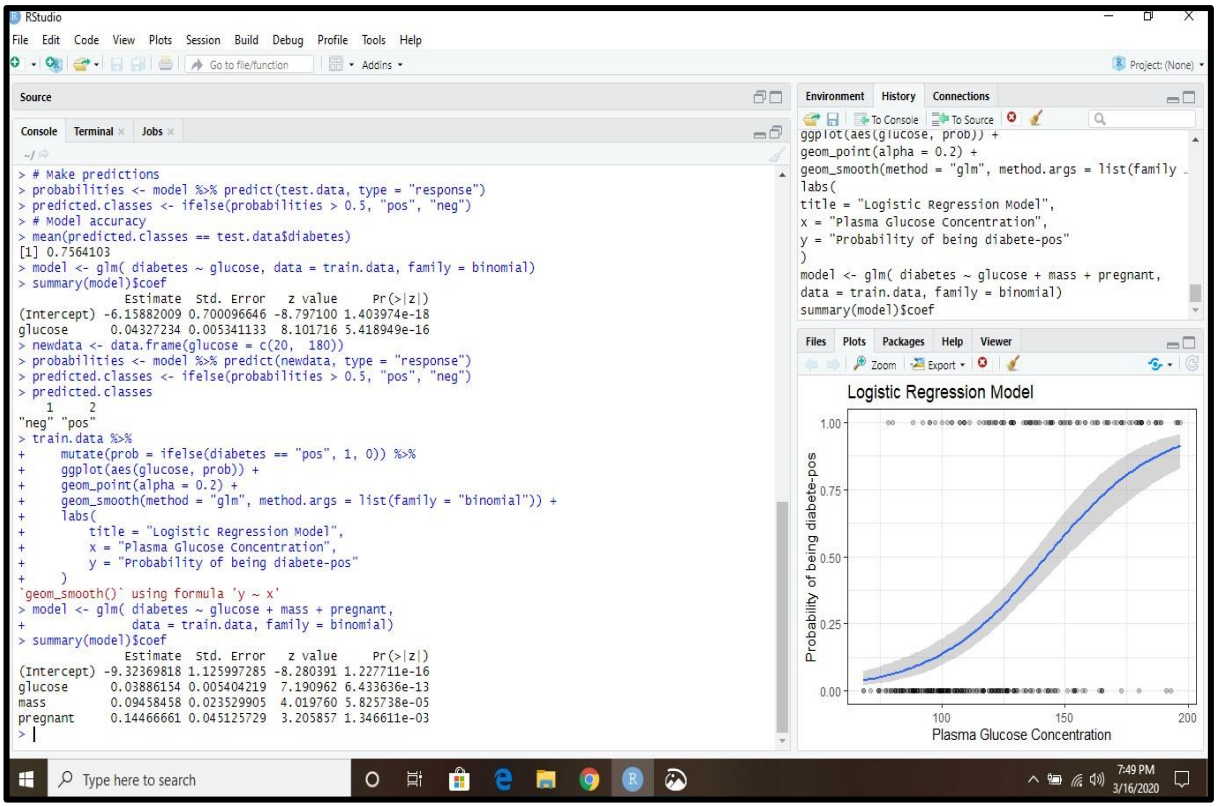
The Naïve Bayes classifier is a simple probabilistic classifier which is based on Bayes theorem but with strong assumptions regarding independence. Historically, this technique became popular with applications in email filtering, spam detection, and document categorization. Although it is often

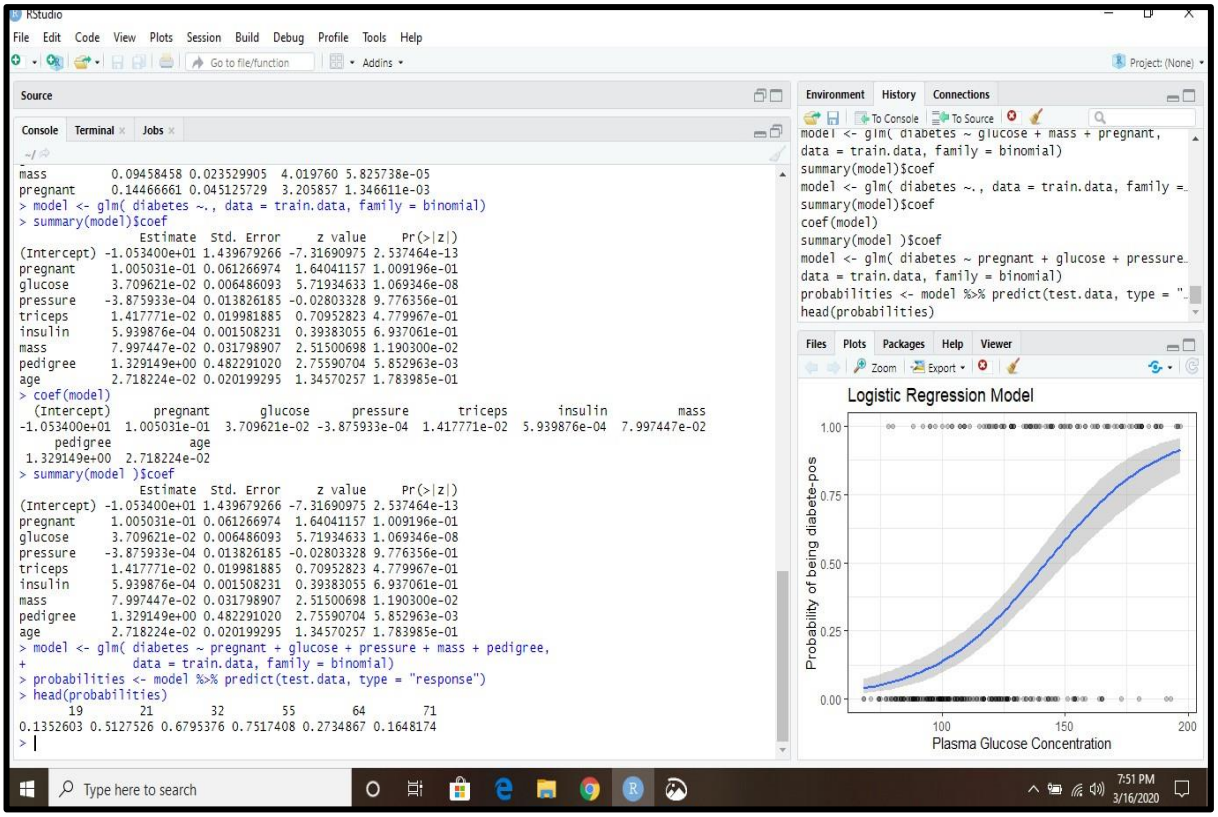
outperformed by other techniques, and despite the naïve design and oversimplified assumptions, this classifier can perform well in many complex real-world problems. And since it is a resource efficient algorithm that is fast and scales well, it is definitely a machine learning algorithm to have in your toolkit.

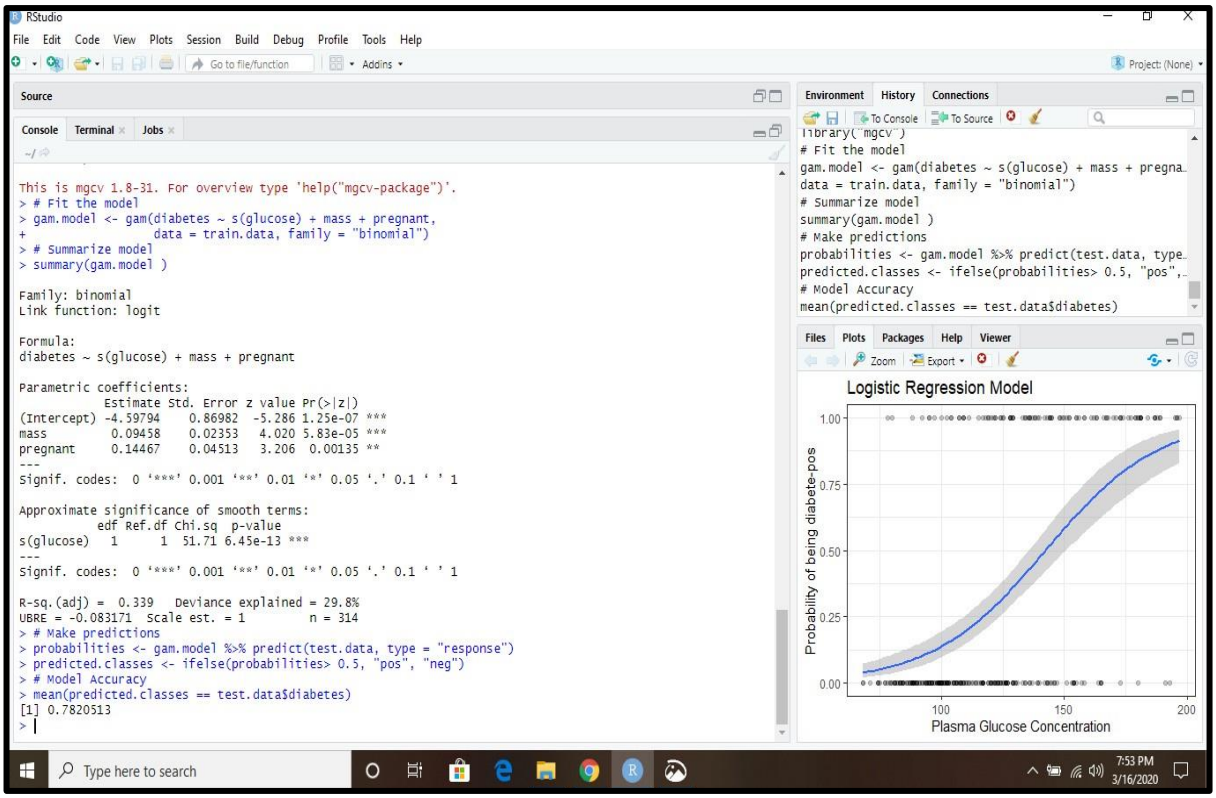
Output:






RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Jobs
> # Make predictions
> probabilities <- model %>% predict(test.data, type = "response")
> predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
> # Model accuracy
> mean(predicted.classes == test.data$diabetes)
[1] 0.7564103
> model <- glm(diabetes ~ glucose, data = train.data, family = binomial)
> summary(model)$coef
Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.15882009 0.700096646 -8.797100 1.403974e-18
glucose 0.04327234 0.005341133 8.101716 5.418949e-16
> newdata <- data.frame(glucose = c(20, 180))
> probabilities <- model %>% predict(newdata, type = "response")
> predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
> predicted.classes
1 2
"neg" "pos"
> train.data %>%
+ mutate(prob = ifelse(diabetes == "pos", 1, 0)) %>%
+ ggplot(aes(glucose, prob)) +
+ geom_point(alpha = 0.2) +
+ geom_smooth(method = "glm", method.args = list(family = "binomial")) +
+ labs(
+ title = "Logistic Regression Model",
+ x = "Plasma Glucose Concentration",
+ y = "Probability of being diabete-pos"
+)
`geom_smooth()` using formula 'y ~ x'
> model <- glm(diabetes ~ glucose + mass + pregnant,
+ data = train.data, family = binomial)
> summary(model)$coef
Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.32369818 1.125997285 -8.280391 1.227711e-16
glucose 0.03886154 0.005404219 7.190962 6.433636e-13
mass 0.09458458 0.023529905 4.019760 5.825738e-05
pregnant 0.14466661 0.045125729 3.205857 1.346611e-03
Environment History Connections
To Console To Source
ggplot(aes(glucose, prob)) +
geom_point(alpha = 0.2) +
geom_smooth(method = "glm", method.args = list(family
labs(
title = "Logistic Regression Model",
x = "Plasma Glucose Concentration",
y = "Probability of being diabete-pos"
)
model <- glm(diabetes ~ glucose + mass + pregnant,
data = train.data, family = binomial)
summary(model)$coef
Files Plots Packages Help Viewer
Zoom Export
Logistic Regression Model
Probability of being diabete-pos
1.00
0.75
0.50
0.25
0.00
100
150
200
Plasma Glucose Concentration
Type here to search
7:49 PM
3/16/2020


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Jobs
mass 0.09458458 0.023529905 4.019760 5.825738e-05
pregnant 0.14466661 0.045125729 3.205857 1.346611e-03
> model <- glm(diabetes ~., data = train.data, family = binomial)
> summary(model)$coef
Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.053400e+01 1.439679266 -7.31690975 2.537464e-13
pregnant 1.005031e-01 0.061266974 1.64041157 1.009196e-01
glucose 3.709621e-02 0.006486093 5.71934633 1.069346e-08
pressure -3.875933e-04 0.013826185 -0.02803328 9.776356e-01
triceps 1.417771e-02 0.019981885 0.70952823 4.779967e-01
insulin 5.939876e-04 0.001508231 0.39383055 6.937061e-01
mass 7.997447e-02 0.031798907 2.51500698 1.190300e-02
pedigree 1.329149e+00 0.482291020 2.75590704 5.852963e-03
age 2.718224e-02 0.020199295 1.34570257 1.783985e-01
> coef(model)
(Intercept) pregnant glucose pressure triceps insulin mass
-1.053400e+01 1.005031e-01 3.709621e-02 -3.875933e-04 1.417771e-02 5.939876e-04 7.997447e-02
pedigree age
1.329149e+00 2.718224e-02
> summary(model)$coef
Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.053400e+01 1.439679266 -7.31690975 2.537464e-13
pregnant 1.005031e-01 0.061266974 1.64041157 1.009196e-01
glucose 3.709621e-02 0.006486093 5.71934633 1.069346e-08
pressure -3.875933e-04 0.013826185 -0.02803328 9.776356e-01
triceps 1.417771e-02 0.019981885 0.70952823 4.779967e-01
insulin 5.939876e-04 0.001508231 0.39383055 6.937061e-01
mass 7.997447e-02 0.031798907 2.51500698 1.190300e-02
pedigree 1.329149e+00 0.482291020 2.75590704 5.852963e-03
age 2.718224e-02 0.020199295 1.34570257 1.783985e-01
> model <- glm(diabetes ~ pregnant + glucose + pressure + mass + pedigree,
+ data = train.data, family = binomial)
> probabilities <- model %>% predict(test.data, type = "response")
> head(probabilities)
19 21 32 55 64 71
0.1352603 0.5127526 0.6795376 0.7517408 0.2734867 0.1648174
Environment History Connections
To Console To Source
model <- glm(diabetes ~ glucose + mass + pregnant,
data = train.data, family = binomial)
summary(model)$coef
model <- glm(diabetes ~., data = train.data, family =
summary(model)$coef
coef(model)
summary(model)$coef
model <- glm(diabetes ~ pregnant + glucose + pressure
data = train.data, family = binomial)
probabilities <- model %>% predict(test.data, type = "
head(probabilities)
Files Plots Packages Help Viewer
Zoom Export
Logistic Regression Model
Probability of being diabete-pos
1.00
0.75
0.50
0.25
0.00
100
150
200
Plasma Glucose Concentration
Type here to search
7:51 PM
3/16/2020



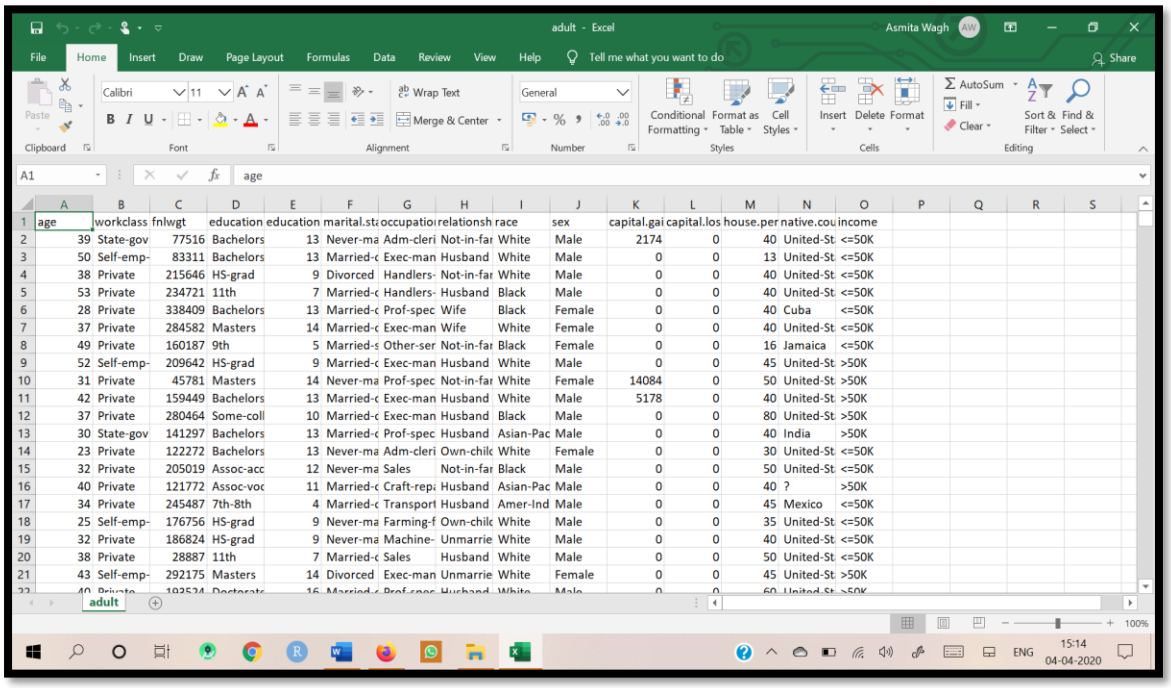
Conclusion: Thus, we have studied Classification on large and noisy dataset with R.

Experiment no:12

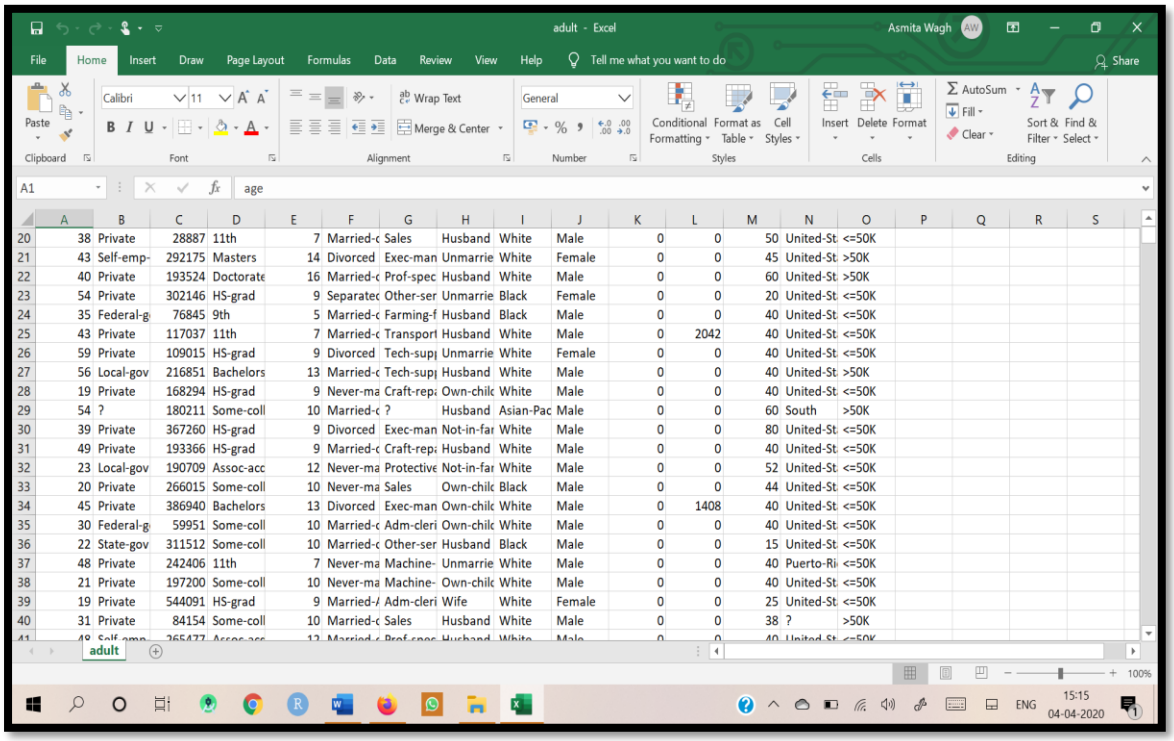
Project: Mini-Project for R-Lab

Project-Name: Salary Prediction

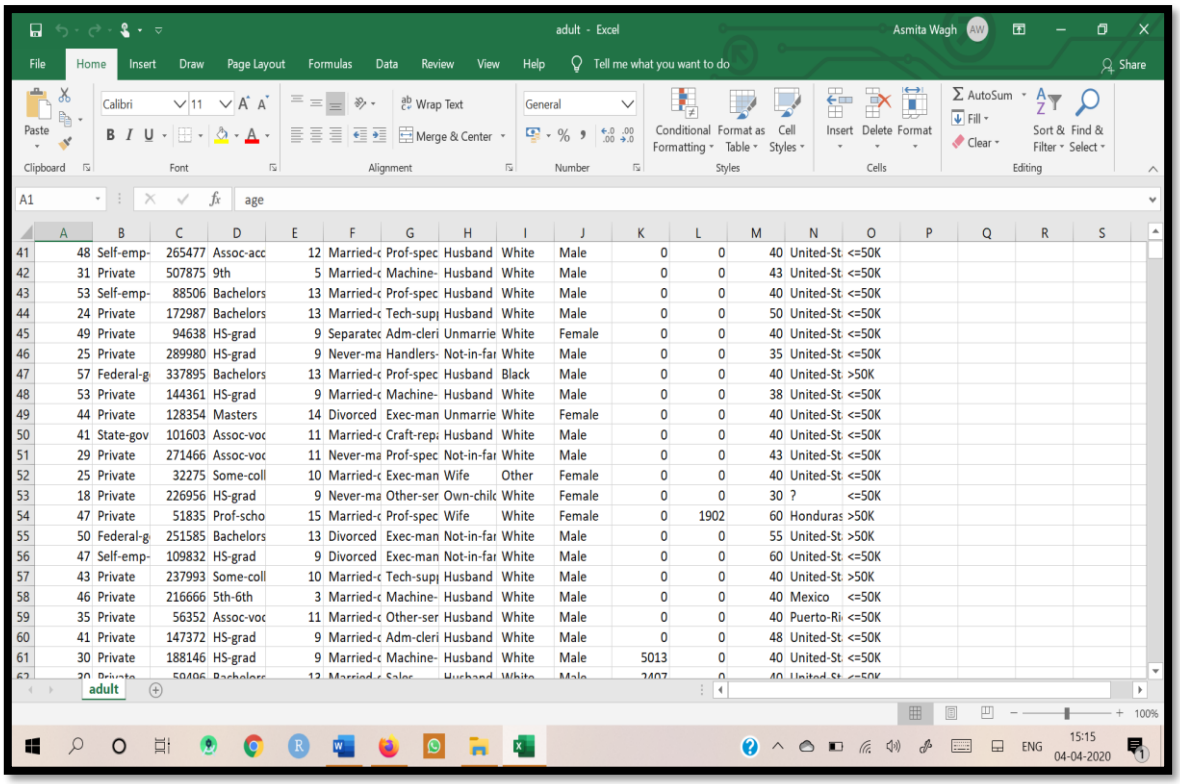
Dataset:



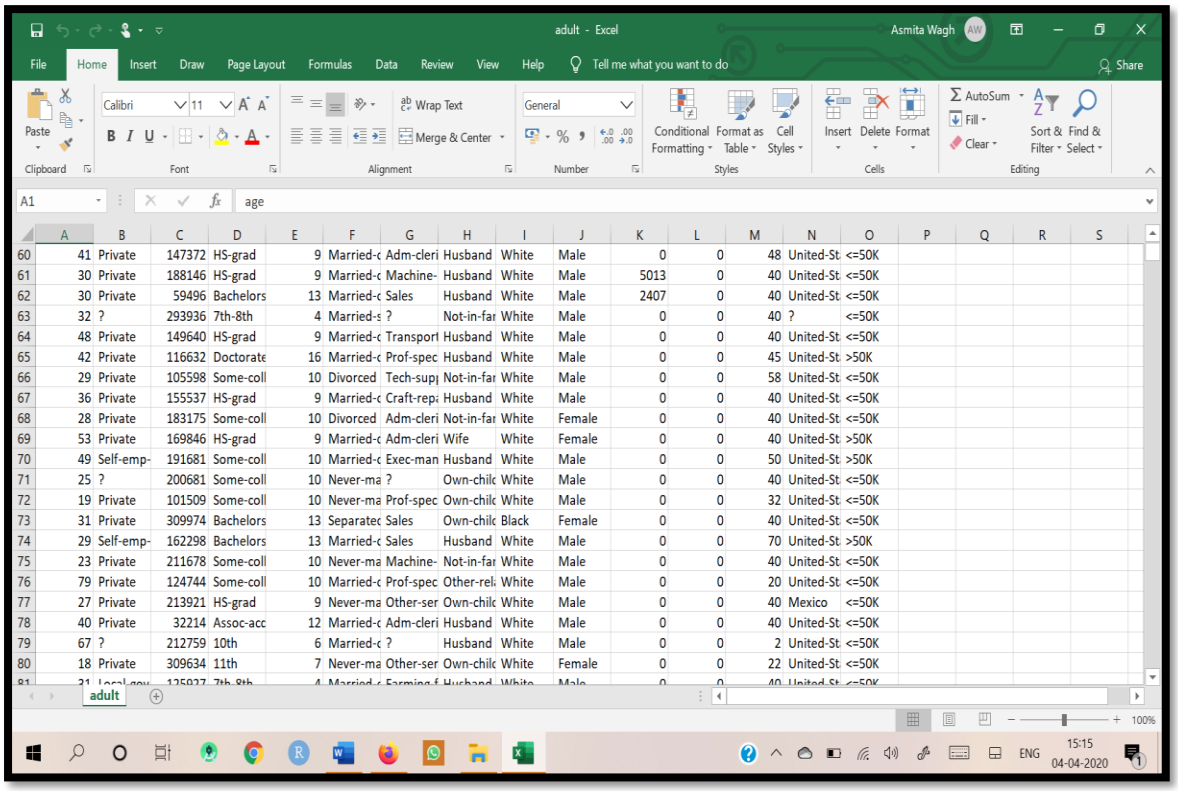
age	workclass	fnlwgt	education	education	marital.sta	occupatio	relationsh	race	sex	capital.gai	capital.los	house.per	native.cou	income
39	State-gov	77516	Bachelors	13	Never-ma	Adm-cleri	Not-in-fa	White	Male	2174	0	40	United-St	<=50K
50	Self-emp-	83311	Bachelors	13	Married-c	Exec-man	Husband	White	Male	0	0	13	United-St	<=50K
38	Private	215646	HS-grad	9	Divorced	Handlers-	Not-in-fa	White	Male	0	0	40	United-St	<=50K
53	Private	234721	11th	7	Married-c	Handlers-	Husband	Black	Male	0	0	40	United-St	<=50K
28	Private	338409	Bachelors	13	Married-c	Prof-spec	Wife	Black	Female	0	0	40	Cuba	<=50K
37	Private	284582	Masters	14	Married-c	Exec-man	Wife	White	Female	0	0	40	United-St	<=50K
49	Private	160187	9th	5	Married-s	Other-ser	Not-in-fa	Black	Female	0	0	16	Jamaica	<=50K
52	Self-emp-	209642	HS-grad	9	Married-c	Exec-man	Husband	White	Male	0	0	45	United-St	>50K
31	Private	45781	Masters	14	Never-ma	Prof-spec	Not-in-fa	White	Female	14084	0	50	United-St	>50K
42	Private	159449	Bachelors	13	Married-c	Exec-man	Husband	White	Male	5178	0	40	United-St	>50K
37	Private	280464	Some-coll	10	Married-c	Exec-man	Husband	Black	Male	0	0	80	United-St	>50K
30	State-gov	141297	Bachelors	13	Married-c	Prof-spec	Husband	Asian-Pac	Male	0	0	40	India	>50K
23	Private	122272	Bachelors	13	Never-ma	Adm-cleri	Own-chil	White	Female	0	0	30	United-St	<=50K
32	Private	205019	Assoc-acc	12	Never-ma	Sales	Not-in-fa	Black	Male	0	0	50	United-St	<=50K
40	Private	121772	Assoc-voc	11	Married-c	Craft-repa	Husband	Asian-Pac	Male	0	0	40	?	>50K
34	Private	245487	7th-8th	4	Married-c	Transport	Husband	Amer-Ind	Male	0	0	45	Mexico	<=50K
25	Self-emp-	176756	HS-grad	9	Never-ma	Farming-f	Own-chil	White	Male	0	0	35	United-St	<=50K
32	Private	186824	HS-grad	9	Never-ma	Machine-	Unmarrie	White	Male	0	0	40	United-St	<=50K
38	Private	28887	11th	7	Married-c	Sales	Husband	White	Male	0	0	50	United-St	<=50K
43	Self-emp-	292175	Masters	14	Divorced	Exec-man	Unmarrie	White	Female	0	0	45	United-St	>50K



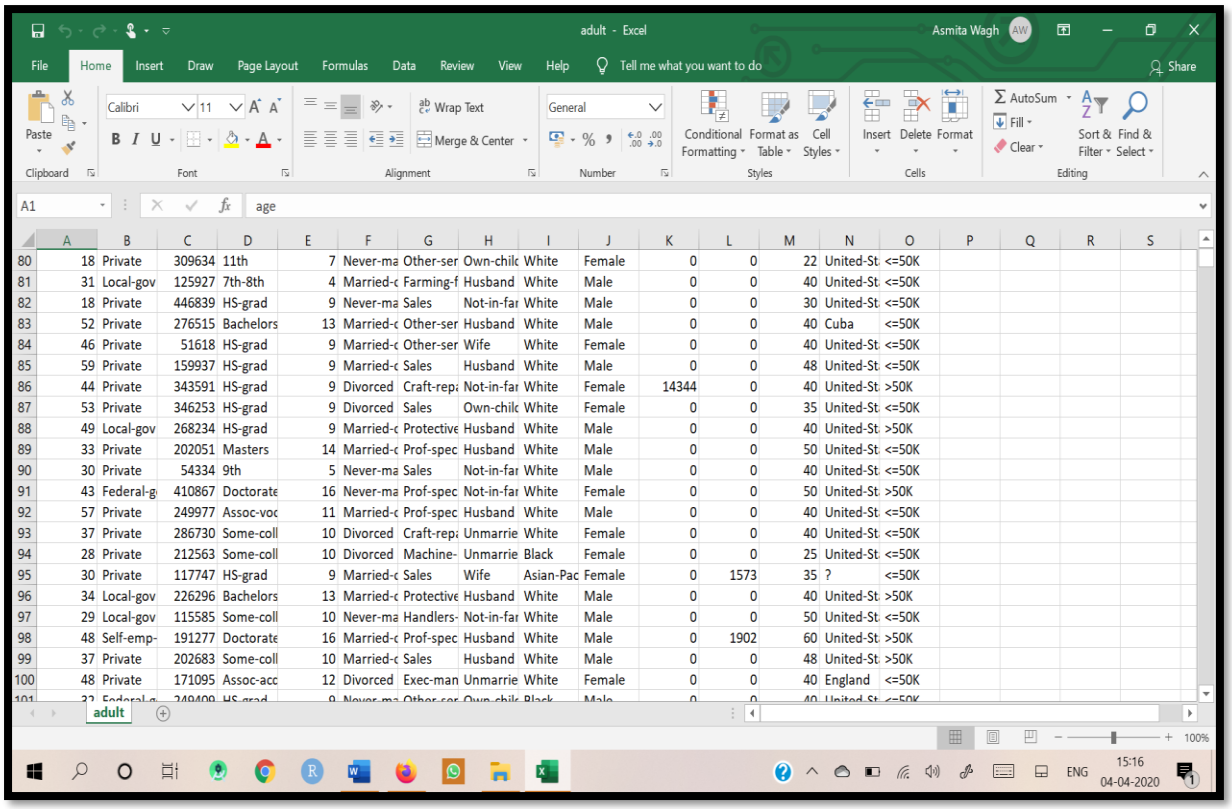
38	Private	28887	11th	7	Married-c	Sales	Husband	White	Male	0	0	50	United-St	<=50K
43	Self-emp-	292175	Masters	14	Divorced	Exec-man	Unmarrie	White	Female	0	0	45	United-St	>50K
40	Private	193524	Doctorate	16	Married-c	Prof-spec	Husband	White	Male	0	0	60	United-St	>50K
54	Private	302146	HS-grad	9	Separatec	Other-ser	Unmarrie	Black	Female	0	0	20	United-St	<=50K
35	Federal-g	76845	9th	5	Married-c	Farming-f	Husband	Black	Male	0	0	40	United-St	<=50K
43	Private	117037	11th	7	Married-c	Transport	Husband	White	Male	0	2042	40	United-St	<=50K
59	Private	109015	HS-grad	9	Divorced	Tech-sup	Unmarrie	White	Female	0	0	40	United-St	<=50K
56	Local-gov	216851	Bachelors	13	Married-c	Tech-sup	Husband	White	Male	0	0	40	United-St	>50K
19	Private	168294	HS-grad	9	Never-ma	Craft-repa	Own-chil	White	Male	0	0	40	United-St	<=50K
54	?	180211	Some-coll	10	Married-c	?	Husband	Asian-Pac	Male	0	0	60	South	>50K
39	Private	367260	HS-grad	9	Divorced	Exec-man	Not-in-fa	White	Male	0	0	80	United-St	<=50K
49	Private	193366	HS-grad	9	Married-c	Craft-repa	Husband	White	Male	0	0	40	United-St	<=50K
23	Local-gov	190709	Assoc-acc	12	Never-ma	Protective	Not-in-fa	White	Male	0	0	52	United-St	<=50K
20	Private	266015	Some-coll	10	Never-ma	Sales	Own-chil	Black	Male	0	0	44	United-St	<=50K
45	Private	386940	Bachelors	13	Divorced	Exec-man	Own-chil	White	Male	0	1408	40	United-St	<=50K
30	Federal-g	59951	Some-coll	10	Married-c	Adm-cleri	Own-chil	White	Male	0	0	40	United-St	<=50K
22	State-gov	311512	Some-coll	10	Married-c	Other-ser	Husband	Black	Male	0	0	15	United-St	<=50K
48	Private	242406	11th	7	Never-ma	Machine-	Unmarrie	White	Male	0	0	40	Puerto-Ri	<=50K
21	Private	197200	Some-coll	10	Never-ma	Machine-	Own-chil	White	Male	0	0	40	United-St	<=50K
19	Private	544091	HS-grad	9	Married-/	Adm-cleri	Wife	White	Female	0	0	25	United-St	<=50K
31	Private	84154	Some-coll	10	Married-c	Sales	Husband	White	Male	0	0	38	?	>50K



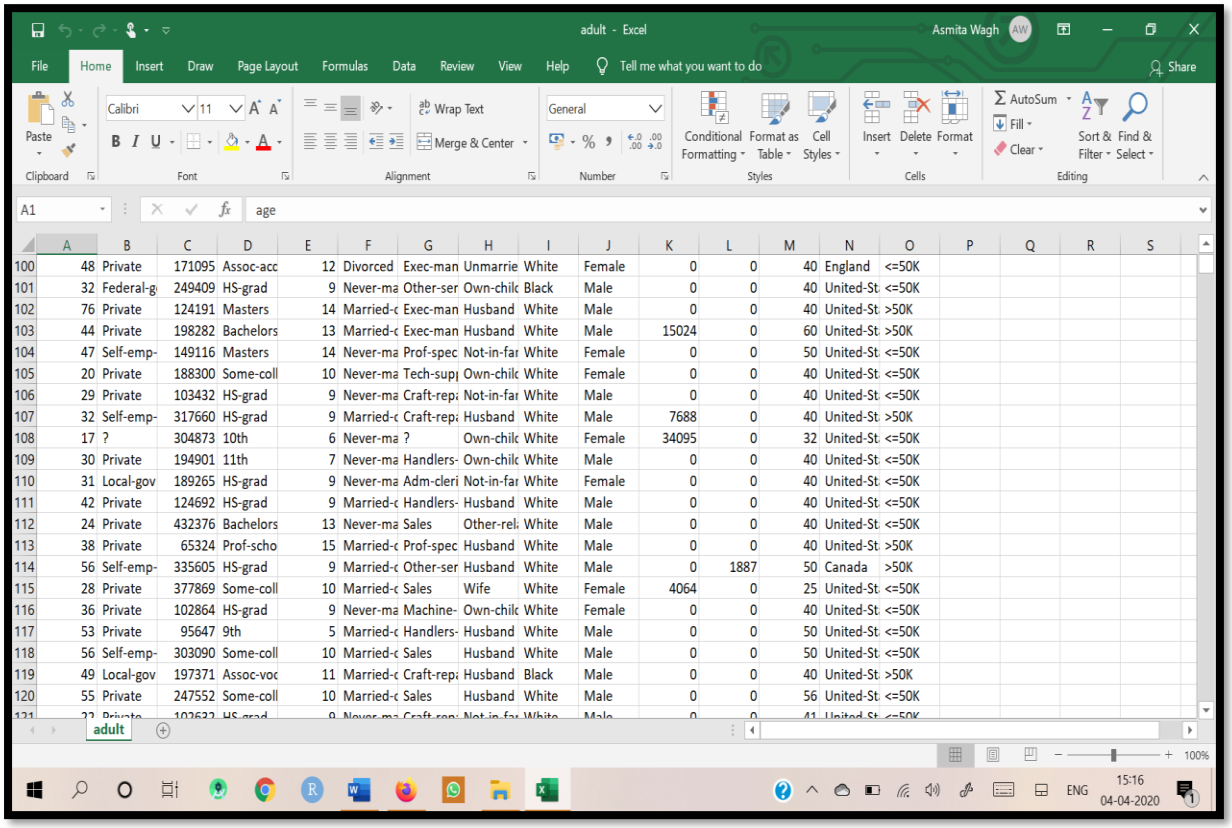
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
41	48	Self-emp-	265477	Assoc-acc	12	Married-c	Prof-spec	Husband	White	Male	0	0	40	United-St	<=50K
42	31	Private	507875	9th	5	Married-c	Machine-	Husband	White	Male	0	0	43	United-St	<=50K
43	53	Self-emp-	88506	Bachelors	13	Married-c	Prof-spec	Husband	White	Male	0	0	40	United-St	<=50K
44	24	Private	172987	Bachelors	13	Married-c	Tech-sup	Husband	White	Male	0	0	50	United-St	<=50K
45	49	Private	94638	HS-grad	9	Separatec	Adm-cleri	Unmarrie	White	Female	0	0	40	United-St	<=50K
46	25	Private	289980	HS-grad	9	Never-ma	Handlers-	Not-in-fa	White	Male	0	0	35	United-St	<=50K
47	57	Federal-g	337895	Bachelors	13	Married-c	Prof-spec	Husband	Black	Male	0	0	40	United-St	>50K
48	53	Private	144361	HS-grad	9	Married-c	Machine-	Husband	White	Male	0	0	38	United-St	<=50K
49	44	Private	128354	Masters	14	Divorced	Exec-man	Unmarrie	White	Female	0	0	40	United-St	<=50K
50	41	State-gov	101603	Assoc-voc	11	Married-c	Craft-rep	Husband	White	Male	0	0	40	United-St	<=50K
51	29	Private	271466	Assoc-voc	11	Never-ma	Prof-spec	Not-in-fa	White	Male	0	0	43	United-St	<=50K
52	25	Private	32275	Some-coll	10	Married-c	Exec-man	Wife	Other	Female	0	0	40	United-St	<=50K
53	18	Private	226956	HS-grad	9	Never-ma	Other-ser	Own-child	White	Female	0	0	30	?	<=50K
54	47	Private	51835	Prof-scho	15	Married-c	Prof-spec	Wife	White	Female	0	1902	60	Honduras	>50K
55	50	Federal-g	251585	Bachelors	13	Divorced	Exec-man	Not-in-fa	White	Male	0	0	55	United-St	>50K
56	47	Self-emp-	109832	HS-grad	9	Divorced	Exec-man	Not-in-fa	White	Male	0	0	60	United-St	<=50K
57	43	Private	237993	Some-coll	10	Married-c	Tech-sup	Husband	White	Male	0	0	40	United-St	>50K
58	46	Private	216666	5th-6th	3	Married-c	Machine-	Husband	White	Male	0	0	40	Mexico	<=50K
59	35	Private	56352	Assoc-voc	11	Married-c	Other-ser	Husband	White	Male	0	0	40	Puerto-Ri	<=50K
60	41	Private	147372	HS-grad	9	Married-c	Adm-cleri	Husband	White	Male	0	0	48	United-St	<=50K
61	30	Private	188146	HS-grad	9	Married-c	Machine-	Husband	White	Male	5013	0	40	United-St	<=50K



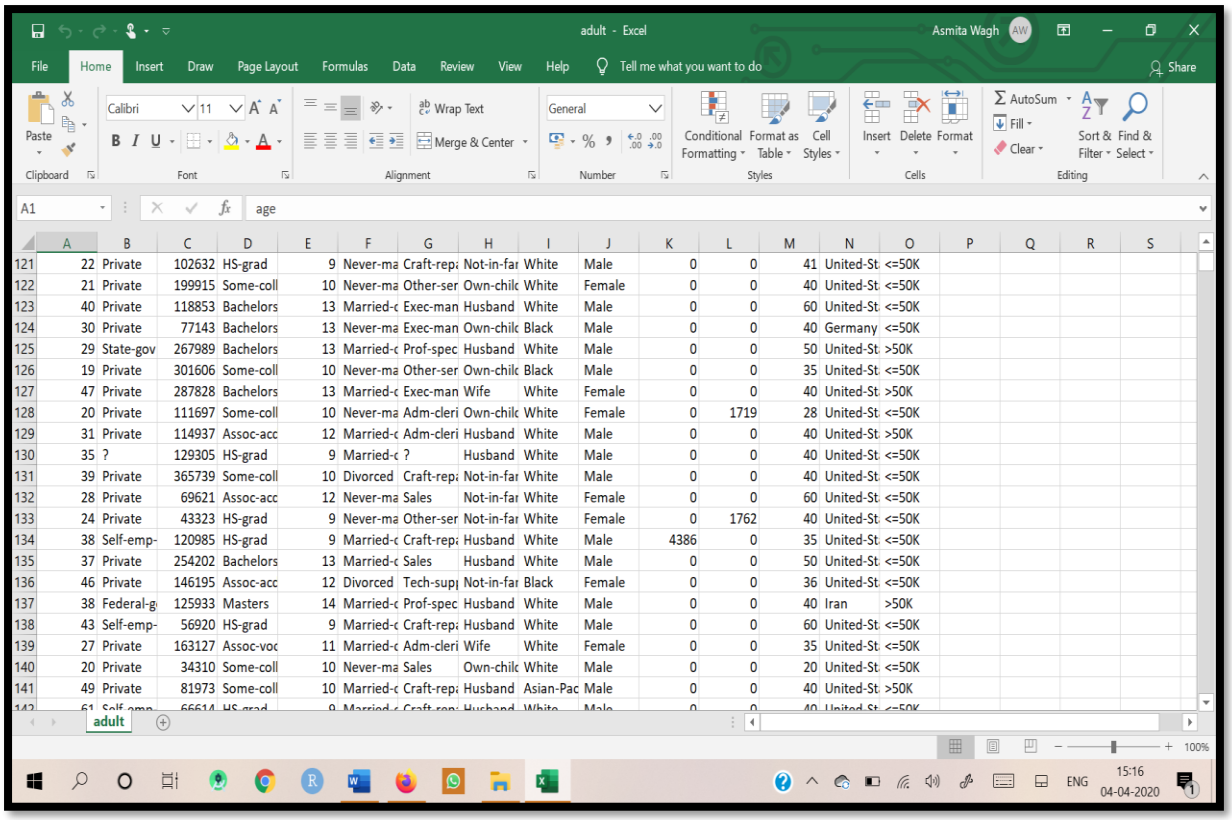
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
60	41	Private	147372	HS-grad	9	Married-c	Adm-cleri	Husband	White	Male	0	0	48	United-St	<=50K
61	30	Private	188146	HS-grad	9	Married-c	Machine-	Husband	White	Male	5013	0	40	United-St	<=50K
62	30	Private	59496	Bachelors	13	Married-c	Sales	Husband	White	Male	2407	0	40	United-St	<=50K
63	32	?	293936	7th-8th	4	Married-s	?	Not-in-fa	White	Male	0	0	40	?	<=50K
64	48	Private	149640	HS-grad	9	Married-c	Transport	Husband	White	Male	0	0	40	United-St	<=50K
65	42	Private	116632	Doctorate	16	Married-c	Prof-spec	Husband	White	Male	0	0	45	United-St	>50K
66	29	Private	105598	Some-coll	10	Divorced	Tech-sup	Not-in-fa	White	Male	0	0	58	United-St	<=50K
67	36	Private	155537	HS-grad	9	Married-c	Craft-rep	Husband	White	Male	0	0	40	United-St	<=50K
68	28	Private	183175	Some-coll	10	Divorced	Adm-cleri	Not-in-fa	White	Female	0	0	40	United-St	<=50K
69	53	Private	169846	HS-grad	9	Married-c	Adm-cleri	Wife	White	Female	0	0	40	United-St	>50K
70	49	Self-emp-	191681	Some-coll	10	Married-c	Exec-man	Husband	White	Male	0	0	50	United-St	>50K
71	25	?	200681	Some-coll	10	Never-ma	?	Own-child	White	Male	0	0	40	United-St	<=50K
72	19	Private	101509	Some-coll	10	Never-ma	Prof-spec	Own-child	White	Male	0	0	32	United-St	<=50K
73	31	Private	309974	Bachelors	13	Separatec	Sales	Own-child	Black	Female	0	0	40	United-St	<=50K
74	29	Self-emp-	162298	Bachelors	13	Married-c	Sales	Husband	White	Male	0	0	70	United-St	>50K
75	23	Private	211678	Some-coll	10	Never-ma	Machine-	Not-in-fa	White	Male	0	0	40	United-St	<=50K
76	79	Private	124744	Some-coll	10	Married-c	Prof-spec	Other-rel	White	Male	0	0	20	United-St	<=50K
77	27	Private	213921	HS-grad	9	Never-ma	Other-ser	Own-child	White	Male	0	0	40	Mexico	<=50K
78	40	Private	32214	Assoc-acc	12	Married-c	Adm-cleri	Husband	White	Male	0	0	40	United-St	<=50K
79	67	?	212759	10th	6	Married-c	?	Husband	White	Male	0	0	2	United-St	<=50K
80	18	Private	309634	11th	7	Never-ma	Other-ser	Own-child	White	Female	0	0	22	United-St	<=50K



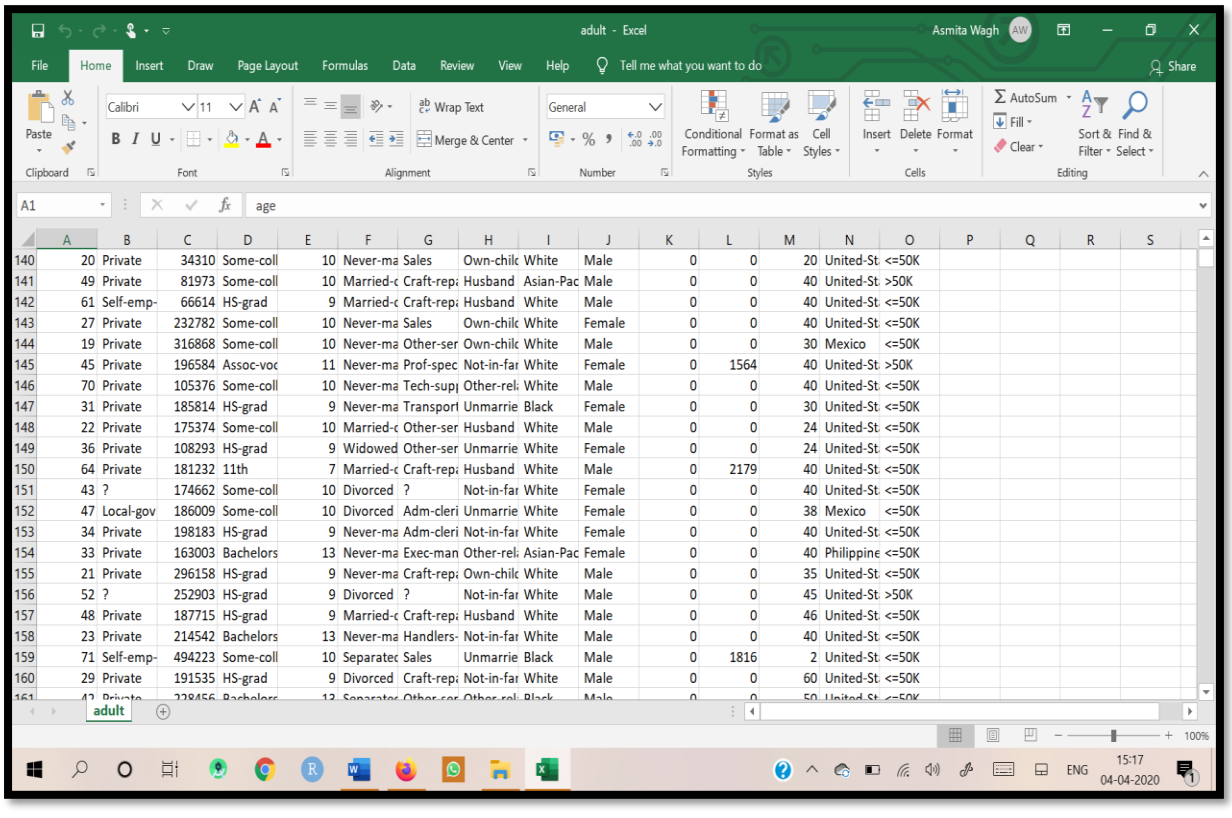
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	18	Private	309634	11th	7	Never-ma	Other-ser	Own-chil	White	Female	0	0	22	United-St	<=50K
81	31	Local-gov	125927	7th-8th	4	Married-c	Farming-f	Husband	White	Male	0	0	40	United-St	<=50K
82	18	Private	446839	HS-grad	9	Never-ma	Sales	Not-in-fa	White	Male	0	0	30	United-St	<=50K
83	52	Private	276515	Bachelors	13	Married-c	Other-ser	Husband	White	Male	0	0	40	Cuba	<=50K
84	46	Private	51618	HS-grad	9	Married-c	Other-ser	Wife	White	Female	0	0	40	United-St	<=50K
85	59	Private	159937	HS-grad	9	Married-c	Sales	Husband	White	Male	0	0	48	United-St	<=50K
86	44	Private	343591	HS-grad	9	Divorced	Craft-repa	Not-in-fa	White	Female	14344	0	40	United-St	>50K
87	53	Private	346253	HS-grad	9	Divorced	Sales	Own-chil	White	Female	0	0	35	United-St	<=50K
88	49	Local-gov	268234	HS-grad	9	Married-c	Protective	Husband	White	Male	0	0	40	United-St	>50K
89	33	Private	202051	Masters	14	Married-c	Prof-spec	Husband	White	Male	0	0	50	United-St	<=50K
90	30	Private	54334	9th	5	Never-ma	Sales	Not-in-fa	White	Male	0	0	40	United-St	<=50K
91	43	Federal-g	410867	Doctorate	16	Never-ma	Prof-spec	Not-in-fa	White	Female	0	0	50	United-St	>50K
92	57	Private	249977	Assoc-voc	11	Married-c	Prof-spec	Husband	White	Male	0	0	40	United-St	<=50K
93	37	Private	286730	Some-coll	10	Divorced	Craft-repa	Unmarrie	White	Female	0	0	40	United-St	<=50K
94	28	Private	212563	Some-coll	10	Divorced	Machine-	Unmarrie	Black	Female	0	0	25	United-St	<=50K
95	30	Private	117747	HS-grad	9	Married-c	Sales	Wife	Asian-Pac	Female	0	1573	35	?	<=50K
96	34	Local-gov	226296	Bachelors	13	Married-c	Protective	Husband	White	Male	0	0	40	United-St	>50K
97	29	Local-gov	115585	Some-coll	10	Never-ma	Handlers-	Not-in-fa	White	Male	0	0	50	United-St	<=50K
98	48	Self-emp-	191277	Doctorate	16	Married-c	Prof-spec	Husband	White	Male	0	1902	60	United-St	>50K
99	37	Private	202683	Some-coll	10	Married-c	Sales	Husband	White	Male	0	0	48	United-St	>50K
100	48	Private	171095	Assoc-acc	12	Divorced	Exec-man	Unmarrie	White	Female	0	0	40	England	<=50K



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
100	48	Private	171095	Assoc-acc	12	Divorced	Exec-man	Unmarrie	White	Female	0	0	40	England	<=50K
101	32	Federal-g	249409	HS-grad	9	Never-ma	Other-ser	Own-chil	Black	Male	0	0	40	United-St	<=50K
102	76	Private	124191	Masters	14	Married-c	Exec-man	Husband	White	Male	0	0	40	United-St	>50K
103	44	Private	198282	Bachelors	13	Married-c	Exec-man	Husband	White	Male	15024	0	60	United-St	>50K
104	47	Self-emp-	149116	Masters	14	Never-ma	Prof-spec	Not-in-fa	White	Female	0	0	50	United-St	<=50K
105	20	Private	188300	Some-coll	10	Never-ma	Tech-sup	Own-chil	White	Female	0	0	40	United-St	<=50K
106	29	Private	103432	HS-grad	9	Never-ma	Craft-repa	Not-in-fa	White	Male	0	0	40	United-St	<=50K
107	32	Self-emp-	317660	HS-grad	9	Married-c	Craft-repa	Husband	White	Male	7688	0	40	United-St	>50K
108	17	?	304873	10th	6	Never-ma	?	Own-chil	White	Female	34095	0	32	United-St	<=50K
109	30	Private	194901	11th	7	Never-ma	Handlers-	Own-chil	White	Male	0	0	40	United-St	<=50K
110	31	Local-gov	189265	HS-grad	9	Never-ma	Adm-cleri	Not-in-fa	White	Female	0	0	40	United-St	<=50K
111	42	Private	124692	HS-grad	9	Married-c	Handlers-	Husband	White	Male	0	0	40	United-St	<=50K
112	24	Private	432376	Bachelors	13	Never-ma	Sales	Other-rel	White	Male	0	0	40	United-St	<=50K
113	38	Private	65324	Prof-scho	15	Married-c	Prof-spec	Husband	White	Male	0	0	40	United-St	>50K
114	56	Self-emp-	335605	HS-grad	9	Married-c	Other-ser	Husband	White	Male	0	1887	50	Canada	>50K
115	28	Private	377869	Some-coll	10	Married-c	Sales	Wife	White	Female	4064	0	25	United-St	<=50K
116	36	Private	102864	HS-grad	9	Never-ma	Machine-	Own-chil	White	Female	0	0	40	United-St	<=50K
117	53	Private	95647	9th	5	Married-c	Handlers-	Husband	White	Male	0	0	50	United-St	<=50K
118	56	Self-emp-	303090	Some-coll	10	Married-c	Sales	Husband	White	Male	0	0	50	United-St	<=50K
119	49	Local-gov	197371	Assoc-voc	11	Married-c	Craft-repa	Husband	Black	Male	0	0	40	United-St	>50K
120	55	Private	247552	Some-coll	10	Married-c	Sales	Husband	White	Male	0	0	56	United-St	<=50K



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
121	22	Private	102632	HS-grad	9	Never-ma	Craft-repa	Not-in-fa	White	Male	0	0	41	United-St	<=50K
122	21	Private	199915	Some-coll	10	Never-ma	Other-ser	Own-child	White	Female	0	0	40	United-St	<=50K
123	40	Private	118853	Bachelors	13	Married-c	Exec-man	Husband	White	Male	0	0	60	United-St	<=50K
124	30	Private	77143	Bachelors	13	Never-ma	Exec-man	Own-child	Black	Male	0	0	40	Germany	<=50K
125	29	State-gov	267989	Bachelors	13	Married-c	Prof-spec	Husband	White	Male	0	0	50	United-St	>50K
126	19	Private	301606	Some-coll	10	Never-ma	Other-ser	Own-child	Black	Male	0	0	35	United-St	<=50K
127	47	Private	287828	Bachelors	13	Married-c	Exec-man	Wife	White	Female	0	0	40	United-St	>50K
128	20	Private	111697	Some-coll	10	Never-ma	Adm-cleri	Own-child	White	Female	0	1719	28	United-St	<=50K
129	31	Private	114937	Assoc-acc	12	Married-c	Adm-cleri	Husband	White	Male	0	0	40	United-St	>50K
130	35	?	129305	HS-grad	9	Married-c	?	Husband	White	Male	0	0	40	United-St	<=50K
131	39	Private	365739	Some-coll	10	Divorced	Craft-repa	Not-in-fa	White	Male	0	0	40	United-St	<=50K
132	28	Private	69621	Assoc-acc	12	Never-ma	Sales	Not-in-fa	White	Female	0	0	60	United-St	<=50K
133	24	Private	43323	HS-grad	9	Never-ma	Other-ser	Not-in-fa	White	Female	0	1762	40	United-St	<=50K
134	38	Self-emp-	120985	HS-grad	9	Married-c	Craft-repa	Husband	White	Male	4386	0	35	United-St	<=50K
135	37	Private	254202	Bachelors	13	Married-c	Sales	Husband	White	Male	0	0	50	United-St	<=50K
136	46	Private	146195	Assoc-acc	12	Divorced	Tech-supp	Not-in-fa	Black	Female	0	0	36	United-St	<=50K
137	38	Federal-g	125933	Masters	14	Married-c	Prof-spec	Husband	White	Male	0	0	40	Iran	>50K
138	43	Self-emp-	56920	HS-grad	9	Married-c	Craft-repa	Husband	White	Male	0	0	60	United-St	<=50K
139	27	Private	163127	Assoc-voc	11	Married-c	Adm-cleri	Wife	White	Female	0	0	35	United-St	<=50K
140	20	Private	34310	Some-coll	10	Never-ma	Sales	Own-child	White	Male	0	0	20	United-St	<=50K
141	49	Private	81973	Some-coll	10	Married-c	Craft-repa	Husband	Asian-Pac	Male	0	0	40	United-St	>50K



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
140	20	Private	34310	Some-coll	10	Never-ma	Sales	Own-child	White	Male	0	0	20	United-St	<=50K
141	49	Private	81973	Some-coll	10	Married-c	Craft-repa	Husband	Asian-Pac	Male	0	0	40	United-St	>50K
142	61	Self-emp-	66614	HS-grad	9	Married-c	Craft-repa	Husband	White	Male	0	0	40	United-St	<=50K
143	27	Private	232782	Some-coll	10	Never-ma	Sales	Own-child	White	Female	0	0	40	United-St	<=50K
144	19	Private	316868	Some-coll	10	Never-ma	Other-ser	Own-child	White	Male	0	0	30	Mexico	<=50K
145	45	Private	196584	Assoc-voc	11	Never-ma	Prof-spec	Not-in-fa	White	Female	0	1564	40	United-St	>50K
146	70	Private	105376	Some-coll	10	Never-ma	Tech-supp	Other-rel	White	Male	0	0	40	United-St	<=50K
147	31	Private	185814	HS-grad	9	Never-ma	Transport	Unmarrie	Black	Female	0	0	30	United-St	<=50K
148	22	Private	175374	Some-coll	10	Married-c	Other-ser	Husband	White	Male	0	0	24	United-St	<=50K
149	36	Private	108293	HS-grad	9	Widowed	Other-ser	Unmarrie	White	Female	0	0	24	United-St	<=50K
150	64	Private	181232	11th	7	Married-c	Craft-repa	Husband	White	Male	0	2179	40	United-St	<=50K
151	43	?	174662	Some-coll	10	Divorced	?	Not-in-fa	White	Female	0	0	40	United-St	<=50K
152	47	Local-gov	186009	Some-coll	10	Divorced	Adm-cleri	Unmarrie	White	Female	0	0	38	Mexico	<=50K
153	34	Private	198183	HS-grad	9	Never-ma	Adm-cleri	Not-in-fa	White	Female	0	0	40	United-St	<=50K
154	33	Private	163003	Bachelors	13	Never-ma	Exec-man	Other-rel	Asian-Pac	Female	0	0	40	Philippine	<=50K
155	21	Private	296158	HS-grad	9	Never-ma	Craft-repa	Own-child	White	Male	0	0	35	United-St	<=50K
156	52	?	252903	HS-grad	9	Divorced	?	Not-in-fa	White	Male	0	0	45	United-St	>50K
157	48	Private	187715	HS-grad	9	Married-c	Craft-repa	Husband	White	Male	0	0	46	United-St	<=50K
158	23	Private	214542	Bachelors	13	Never-ma	Handlers-	Not-in-fa	White	Male	0	0	40	United-St	<=50K
159	71	Self-emp-	494223	Some-coll	10	Separatec	Sales	Unmarrie	Black	Male	0	1816	2	United-St	<=50K
160	29	Private	191535	HS-grad	9	Divorced	Craft-repa	Not-in-fa	White	Male	0	0	60	United-St	<=50K

Theory:

The data was extracted from 1994 Census bureau database. Please view the adult.csv file for the dataset, this dataset is what is being used in the entire study of ours.

income <- read.csv ('adult.csv', na. strings = c ('', '?')) …… To assign the csv data into variable income, the na. strings are basically used to specify the content that is unknown in the dataset or left blank.

supply (income, function(x) sum(is.na(x))) ……. basically, counts the data that is unavailable.

After finding it out, I realized that numbers are a bit annoying to comprehend and select the reliable points, henceforth we went ahead and plotted the graphs of it to see how they are.

```
library(Amelia)
missmap(income, main = "Missing values vs observed")
table (complete.cases (income))`
```

This basically allows us to see the plot of missing and observed data.

Check Rplots.pdf for the plot

Since the dataset has already been cleaned running it up again, will not show any false values.

```
library(Amelia)
Loading required package: foreign
##
## Amelia II: Multiple Imputation
## (Version 1.6.4, built: 2012-12-17)
## Copyright (C) 2005-2020 James Honaker, Gary King and Matthew Blackwell
## Refer to http://gking.harvard.edu/amelia/ for more information
##
> missmap(income, main = "Missing values vs observed")
> table (complete.cases (income))

 TRUE
32561
```

So, we go ahead and make a boxplot of everything with respect to income level.

```
library(gridExtra)
p1 <- ggplot(aes(x=income, y=age), data = income) + geom_boxplot() +
  ggtitle('Age vs. Income Level')
p2 <- ggplot(aes(x=income, y=education.num), data = income) + geom_boxplot() +
  ggtitle('Years of Education vs. Income Level')
str(income)
p3 <- ggplot(aes(x=income, y=house.per.week), data = income) + geom_boxplot() + ggtitle('Hours Per week vs. Income Level
p4 <- ggplot(aes(x=income, y=capital.gain), data=income) + geom_boxplot() +
  ggtitle('Capital Gain vs. Income Level')
p5 <- ggplot(aes(x=income, y=capital.loss), data=income) + geom_boxplot() +
  ggtitle('Capital Loss vs. Income Level')
p6 <- ggplot(aes(x=income, y=fnlwgt), data=income) + geom_boxplot() +
  ggtitle('Final Weight vs. Income Level')
grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)
income$fnlwgt <- NULL
```

“Age”, “Years of education” and “hours per week” all show significant variations with income level. Therefore, they are kept for the regression analysis. “Final Weight” does not show any variation with income level, therefore, it has been excluded from the analysis. It’s hard to see whether “Capital gain” and “Capital loss” have variation with Income level from the above plot, so we shall keep them for now.

```
library(dplyr)
by_workclass <- income %>% group_by(workclass, income) %>% summarise(n=n())
by_education <- income %>% group_by(education, income) %>% summarise(n=n())
by_education$education <- ordered(by_education$education,
                                  levels = c('Preschool', '1st-4th', '5th-6th', '7th-8th', '9th', '10th', '11th', '12th
by_marital <- income %>% group_by(marital.status, income) %>% summarise(n=n())
by_occupation <- income %>% group_by(occupation, income) %>% summarise(n=n())
by_relationship <- income %>% group_by(relationship, income) %>% summarise(n=n())
by_race <- income %>% group_by(race, income) %>% summarise(n=n())
by_sex <- income %>% group_by(sex, income) %>% summarise(n=n())
by_country <- income %>% group_by(native.country, income) %>% summarise(n=n())
p7 <- ggplot(aes(x=workclass, y=n, fill=income), data=by_workclass) + geom_bar(stat = 'identity', position = position_do
p8 <- ggplot(aes(x=education, y=n, fill=income), data=by_education) + geom_bar(stat = 'identity', position = position_do
p9 <- ggplot(aes(x=marital.status, y=n, fill=income), data=by_marital) + geom_bar(stat = 'identity', position=position_d
p10 <- ggplot(aes(x=occupation, y=n, fill=income), data=by_occupation) + geom_bar(stat = 'identity', position=position_d
p11 <- ggplot(aes(x=relationship, y=n, fill=income), data=by_relationship) + geom_bar(stat = 'identity', position=positi
p12 <- ggplot(aes(x=race, y=n, fill=income), data=by_race) + geom_bar(stat = 'identity', position = position_dodge()) +
p13 <- ggplot(aes(x=sex, y=n, fill=income), data=by_sex) + geom_bar(stat = 'identity', position = position_dodge()) + gg
p14 <- ggplot(aes(x=native.country, y=n, fill=income), data=by_country) + geom_bar(stat = 'identity', position = positio
grid.arrange(p7, p8, p9, p10, ncol=2)
grid.arrange(p11,p12,p13, ncol=2)
```

Most of the data is collected from the United States, so variable “native country” does not have effect on our analysis, we shall exclude it from regression model.

And all the other categorial variables seem to have reasonable variation, so they will be kept.

income$income = as.factor(ifelse(income$income==income$income[1],0,1))
basically if >50k it will be set to 1 else to 0

```
train <- income[1:22793,]
test <- income[22793:32561,]
model <-glm(income ~.,family=binomial(link='logit'),data=train)
summary(model)
```

So as to fit the model and view the details.

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.9275  -0.5171  -0.1885  -0.0327   3.2694

Coefficients: (2 not defined because of singularities)
                                    Estimate Std. Error z value Pr(>|z|)
(Intercept)                       -8.325e+00  4.908e-01 -16.963  < 2e-16 ***
age                                2.505e-02  1.932e-03  12.961  < 2e-16 ***
workclass Federal-gov              1.100e+00  1.818e-01   6.055 1.41e-09 ***
workclass Local-gov                4.509e-01  1.660e-01   2.716 0.006599 **
workclass Never-worked            -9.225e+00  6.808e+02  -0.014 0.989189
workclass Private                  6.245e-01  1.481e-01   4.216 2.49e-05 ***
workclass Self-emp-inc             8.046e-01  1.766e-01   4.557 5.20e-06 ***
workclass Self-emp-not-inc         1.674e-01  1.623e-01   1.031 0.302392
workclass State-gov                2.851e-01  1.804e-01   1.580 0.114139
workclass Without-pay             -1.279e+01  4.165e+02  -0.031 0.975497
education 11th                     3.831e-02  2.421e-01   0.158 0.874287
education 12th                     3.946e-01  3.182e-01   1.240 0.214856
education 1st-4th                 -5.194e-01  5.324e-01  -0.976 0.329244
education 5th-6th                 -4.645e-01  3.784e-01  -1.228 0.219579
education 7th-8th                 -4.160e-01  2.683e-01  -1.550 0.121113
education 9th                     -4.346e-01  3.154e-01  -1.378 0.168233
education Assoc-acdm               1.312e+00  2.054e-01   6.388 1.68e-10 ***
education Assoc-voc                1.307e+00  1.972e-01   6.627 3.43e-11 ***
education Bachelors                1.885e+00  1.821e-01  10.351  < 2e-16 ***
education Doctorate                2.840e+00  2.522e-01  11.261  < 2e-16 ***
education HS-grad                  7.689e-01  1.773e-01   4.336 1.45e-05 ***
education Masters                  2.181e+00  1.947e-01  11.204  < 2e-16 ***

education Preschool               -1.996e+01  2.167e+02  -0.092 0.926598
education Prof-school              2.614e+00  2.353e-01  11.109  < 2e-16 ***
education Some-college             1.062e+00  1.802e-01   5.897 3.71e-09 ***
education.num                             NA         NA      NA       NA
marital.status Married-AF-spouse   2.687e+00  6.768e-01   3.970 7.19e-05 ***
marital.status Married-civ-spouse  2.246e+00  3.188e-01   7.043 1.88e-12 ***
marital.status Married-spouse-absent -2.308e-02  2.629e-01  -0.088 0.930043
marital.status Never-married      -3.864e-01  1.039e-01  -3.718 0.000201 ***
marital.status Separated           5.553e-02  1.931e-01   0.288 0.773704
marital.status Widowed             1.271e-01  1.917e-01   0.663 0.507395
occupation Adm-clerical            1.184e-02  1.163e-01   0.102 0.918946
occupation Armed-Forces           -1.325e+01  5.210e+02  -0.025 0.979717
occupation Craft-repair            1.318e-01  9.983e-02   1.320 0.186672
occupation Exec-managerial         8.385e-01  1.027e-01   8.166 3.19e-16 ***
occupation Farming-fishing        -9.949e-01  1.696e-01  -5.867 4.45e-09 ***
occupation Handlers-cleaners      -6.268e-01  1.732e-01  -3.620 0.000295 ***
occupation Machine-op-inspct      -3.041e-01  1.256e-01  -2.420 0.015510 *
occupation Other-service          -7.233e-01  1.435e-01  -5.041 4.63e-07 ***
occupation Priv-house-serv        -1.205e+01  1.174e+02  -0.103 0.918289
occupation Prof-specialty          6.148e-01  1.103e-01   5.573 2.50e-08 ***
occupation Protective-serv         5.988e-01  1.526e-01   3.924 8.70e-05 ***
occupation Sales                   2.675e-01  1.058e-01   2.527 0.011507 *
occupation Tech-support            7.188e-01  1.403e-01   5.125 2.98e-07 ***
occupation Transport-moving               NA         NA      NA       NA
relationship Not-in-family         5.715e-01  3.148e-01   1.815 0.069470 .
relationship Other-relative       -5.502e-01  2.934e-01  -1.876 0.060718 .
relationship Own-child            -6.178e-01  3.113e-01  -1.984 0.047209 *
relationship Unmarried             3.774e-01  3.342e-01   1.129 0.258751
relationship Wife                  1.357e+00  1.213e-01  11.193  < 2e-16 ***
```

```
relationship Wife                    1.357e+00  1.213e-01  11.193  < 2e-16 ***
race Asian-Pac-Islander              2.078e-01  2.824e-01   0.736 0.461944
race Black                           3.286e-01  2.690e-01   1.222 0.221864
race Other                          -5.340e-01  4.306e-01  -1.240 0.214975
race White                           4.402e-01  2.565e-01   1.716 0.086104 .
sex Male                             8.320e-01  9.420e-02   8.832  < 2e-16 ***
capital.gain                         3.039e-04  1.187e-05  25.598  < 2e-16 ***
capital.loss                         6.484e-04  4.445e-05  14.587  < 2e-16 ***
house.per.week                       2.919e-02  1.934e-03  15.094  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 25044  on 22792  degrees of freedom
Residual deviance: 14581  on 22736  degrees of freedom
AIC: 14695

Number of Fisher Scoring iterations: 14

Analysis of Deviance Table

Model: binomial, link: logit

Response: income

Terms added sequentially (first to last)
```

```
              Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                          22792      25044
age            1   1152.1     22791      23892 < 2.2e-16 ***
workclass      8    558.3     22783      23334 < 2.2e-16 ***
education     15   2551.2     22768      20783 < 2.2e-16 ***
education.num  0      0.0     22768      20783
marital.status 6   3687.4     22762      17096 < 2.2e-16 ***
occupation    13    552.5     22749      16543 < 2.2e-16 ***
relationship   5    165.6     22744      16377 < 2.2e-16 ***
race           4     19.9     22740      16358 0.0005324 ***
sex            1    105.8     22739      16252 < 2.2e-16 ***
capital.gain   1   1210.5     22738      15041 < 2.2e-16 ***
capital.loss   1    227.1     22737      14814 < 2.2e-16 ***
house.per.week 1    233.4     22736      14581 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "Accuracy : 0.851776026205343"
```

Which gives an accuracy of upto 85%.

Code:

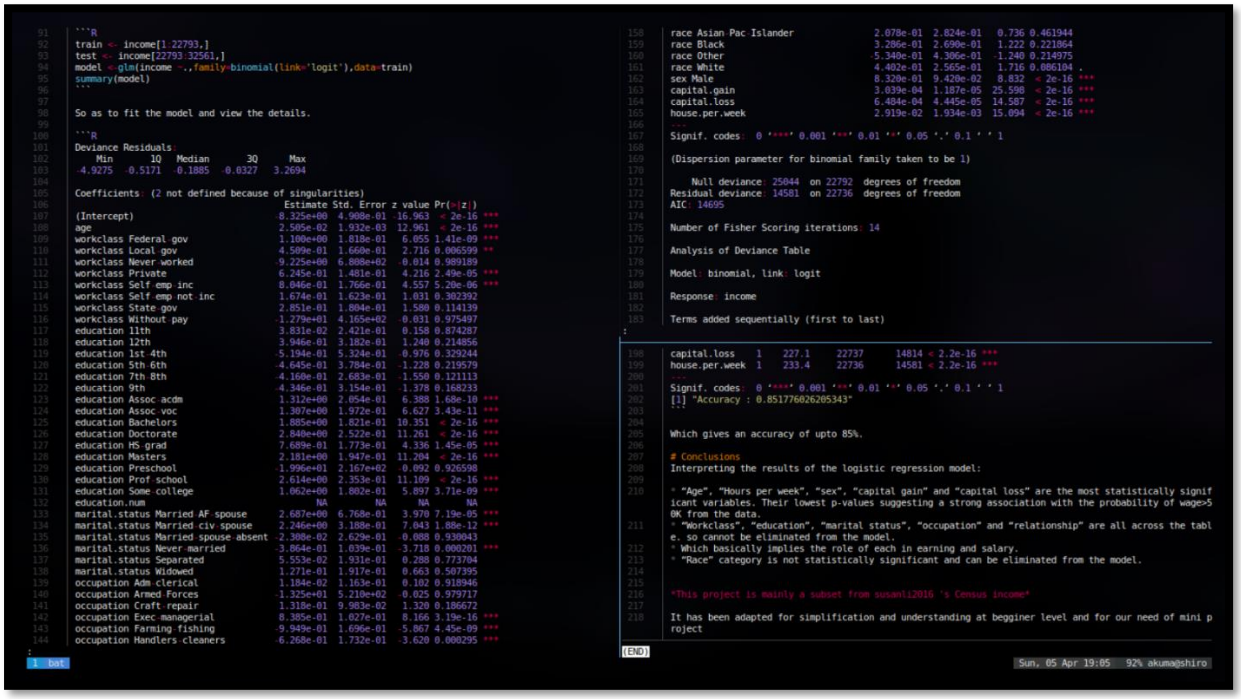
```
1  q()
2  income <- read.csv('adult.csv',na.strings = c('','?'))
```

```
3   str(income0
4   str(income)
5   summary(income)
6   sapply(income,function(x) sum(is.na(x)))
7   sapply(income, function(x) length(unique(x)))
8   library(Amelia)
9   missmap(income, main = "Missing values vs observed")
10  table (complete.cases (income))
11  income <- income[complete.cases(income),]
12  library(ggplot2)
13  library(gridExtra)
14  p1 <- ggplot(aes(x=income, y=age), data = income) + geom_boxplot() +
15      ggtitle('Age vs. Income Level')
16  p2 <- ggplot(aes(x=income, y=education.num), data = income) +
    geom_boxplot() +
17  ggtitle('Years of Education vs. Income Level')
18  str(incomeo)
19  str(income)
20  p3 <- ggplot(aes(x=income, y=hours.per.week), data = income) +
    geom_boxplot() +
21  p3 <- ggplot(aes(x=income, y=house.per.week), data = income) +
    geom_boxplot() +
22  p3 <- ggplot(aes(x=income, y=house.per.week), data = income) +
    geom_boxplot() + ggtitle('Hours Per week vs. Income Level')
23  p4 <- ggplot(aes(x=income, y=capital.gain), data=income) +
    geom_boxplot()+
24  ggtitle('Capital Gain vs. Income Level')
25  p5 <- ggplot(aes(x=income, y=capital.loss), data=income) +
    geom_boxplot()+
26  ggtitle('Capital Loss vs. Income Level')
27  p6 <- ggplot(aes(x=income, y=fnlwgt), data=income) + geom_boxplot() +
28  ggtitle('Final Weight vs. Income Level')
29  grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)
30  income$fnlwgt <- NULL
31  #cuz final weight shows no variation wrt income
32  library(dplyr)
33  by_workclass <- income %>% group_by(workclass, income) %>%
    summarise(n=n())
34  by_education <- income %>% group_by(education, income) %>%
    summarise(n=n())
```

```
by_education$education <- ordered(by_education$education,
levels = c('Preschool', '1st-4th', '5th-6th', '7th-8th', '9th', '10th', '11th',
'12th', 'HS-grad', 'Prof-school', 'Assoc-acdm', 'Assoc-voc', 'Some-college',
'Bachelors', 'Masters', 'Doctorate'))
by_marital <- income %>% group_by(marital.status, income) %>%
summarise(n=n())
by_occupation <- income %>% group_by(occupation, income) %>%
summarise(n=n())
by_relationship <- income %>% group_by(relationship, income) %>%
summarise(n=n())
by_race <- income %>% group_by(race, income) %>% summarise(n=n())
by_sex <- income %>% group_by(sex, income) %>% summarise(n=n())
by_country <- income %>% group_by(native.country, income) %>%
summarise(n=n())
p7 <- ggplot(aes(x=workclass, y=n, fill=income), data=by_workclass) +
geom_bar(stat = 'identity', position = position_dodge()) + ggtitle('Workclass
with Income Level') + theme(axis.text.x = element_text(angle = 45, hjust =
1))
p8 <- ggplot(aes(x=education, y=n, fill=income), data=by_education) +
geom_bar(stat = 'identity', position = position_dodge()) + ggtitle('Education
vs. Income Level') + coord_flip()
p9 <- ggplot(aes(x=marital.status, y=n, fill=income), data=by_marital) +
geom_bar(stat = 'identity', position=position_dodge()) + ggtitle('Marital
Status vs. Income Level') + theme(axis.text.x = element_text(angle = 45,
hjust = 1))
p10 <- ggplot(aes(x=occupation, y=n, fill=income), data=by_occupation) +
geom_bar(stat = 'identity', position=position_dodge()) + ggtitle('Occupation
vs. Income Level') + coord_flip()
p11 <- ggplot(aes(x=relationship, y=n, fill=income), data=by_relationship) +
geom_bar(stat = 'identity', position=position_dodge()) +
ggtitle('Relationship vs. Income Level') + coord_flip()
p12 <- ggplot(aes(x=race, y=n, fill=income), data=by_race) + geom_bar(stat
= 'identity', position = position_dodge()) + ggtitle('Race vs. Income Level') +
coord_flip()
p13 <- ggplot(aes(x=sex, y=n, fill=income), data=by_sex) + geom_bar(stat =
'identity', position = position_dodge()) + ggtitle('Sex vs. Income Level')
p14 <- ggplot(aes(x=native.country, y=n, fill=income), data=by_country) +
geom_bar(stat = 'identity', position = position_dodge()) + ggtitle('Native
Country vs. Income Level') + coord_flip()
grid.arrange(p7, p8, p9, p10, ncol=2)
```

```
#categorical variable exploration and plotting
income$native.country <- NULL
#for simplification and saving myself from headache I prefer to not conduct my study as per the countries
income$income = as.factor(ifelse(income$income==income$income[1],0,1))
#basically if >50k it will be set to 1 else to 0
summary(income)
str(income)
len(income)
count(income)
train <- income(22793)
train <- income[1:22793,]
test <- income[22793:32561,]
model <-glm(income ~.,family=binomial(link='logit'),data=train)
summary(model)
anova(model,test="Chisq")
fitted.results <- predict(model,newdata=test,type='response')
fitted.results <- ifelse(fitted.results > 0.5,1,0)
misClasificError <- mean(fitted.results != test$income)
print(paste('Accuracy :',1-misClasificError))
library(ROCR)
install.packages(ROCR)
R CMD INSTALL ROCR_1.0-1.tar.gz
install.packages(gplot)
q()
getwd()
ls()
savehistory(file="SalaryPrediction")
```

Output:



Conclusion:

Interpreting the results of the logistic regression model:

- “Age”, “Hours per week”, “sex”, “capital gain” and “capital loss” are the most statistically significant variables. Their lowest p-values suggesting a strong association with the probability of wage>50K from the data.
- “Workclass”, “education”, “marital status”, “occupation” and “relationship” are all across the table. so, cannot be eliminated from the model.
- Which basically implies the role of each in earning and salary.
- “Race” category is not statistically significant and can be eliminated from the model.