# PROFESSIONAL TRAINING REPORT

**at**

**Sathyabama Institute of Science and Technology (Deemed to be University)**

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering

by

**GANTYADA MANI ROHITH**

**(Reg. No. 38110157)**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY JEPPIAAR NAGAR, RAJIV GANDHI SALAI,**

**CHENNAI – 600119, TAMILNADU**

**AUGUST 2020**

**SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY**

### (DEEMED TO BE UNIVERSITY)

**Accredited with Grade “A” by NAAC**

**(Established under Section 3 of UGC Act, 1956)**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119**

[**www.sathyabamauniversity.ac.in**](http://www.sathyabamauniversity.ac.in/)

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **GANTYADA MANI ROHITH (38110157)** who carried out the project entitled “” under my supervision from November 2019 to April 2020.

## Internal Guide

## Name

**Head of the Department**



## Submitted for Viva voce Examination held on

**Internal Examiner External Examiner**

**DECLARATION**

I **GANTYADA MANI ROHITH (Reg. No. 38110157)** hereby declare that the Project Report entitled **“Statistics of Data Analysis using R Programming Language”** done by me under the guidance of Ms. **R.Yogitha M.E.,(Ph.D)** **Department of Computer Science and Engineering** at **Sathyabama Institute of Science and Technology** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

## DATE:

**PLACE: SIGNATURE OF THE CANDIDATE**

**ACKNOWLEDGEMENT**

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

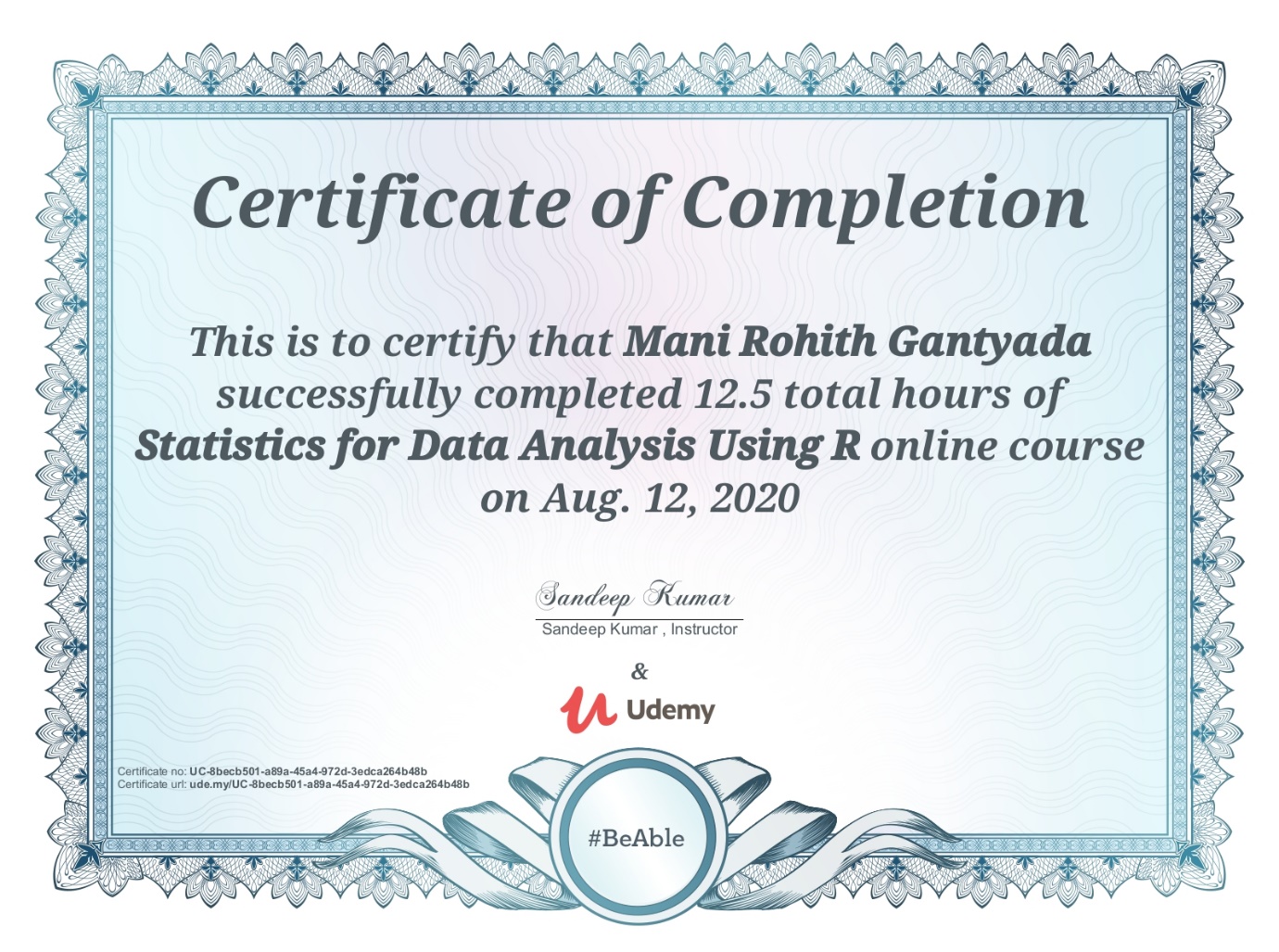
I convey my thanks to **Dr. T.Sasikala M.E., Ph.D.**, **Dean**, School of Computing, **Dr.S.Vigneshwari M.E., Ph.D., and Dr.L.Lakshmanan M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. R.Yogitha M.E., Ph.D.,** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

**TRAINING CERTIFICATE**







**ABSTRACT**

**R** was created by **Ross Ihaka** and **Robert Gentleman** at the University of Auckland, New Zealand, and is developed by the R Development Core Team (of which, as of August 2018, Chambers was a member). R is named partly after the first names of the first two R authors and partly as a play on the name of S. .In this project it is suggested to simplify data and visualize them in a useful way. For Downloading a data. use **Kaggle**, a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges(ex: High Scorer in ICC dataset ..etc) then automatically the file will be saved with **.CSV** (comma-separated values).It is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format. A CSV file typically stores tabular data (numbers and text) in plain text, in which case each line will have the same number of fields**.**  Using the **R studios** import that data into r and using the code available simply that data and visualize them using pie charts, bar charts, line graphs in a way that it can be understood easily. The r is used among statisticians and data miners for developing statistical software and data analysis. Polls, data mining surveys, and studies of scholarly literature databases show substantial increases in popularity. The Many users think of R as a statistics system. We prefer to think of it as an environment within which statistical techniques are implemented. R can be extended (easily) via packages. There are eight packages supplied with the. R is available as free software foundation’s **GNU** general public license in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), windows and macOS.

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE No**

ABSTRACT **vii**

LIST OF FIGURES **x**

LIST OF ABBREVIATIONS **x**i

**1. INTRODUCTION 1**

1.1 What is R 1

1.2 Why R and What is the use of R 2

1.2.1 Why R 2

1.2.2 What is the use for R 2

1.3 Installation of R

1.3.1 Download R 3

1.3.2 Download R Studio 3

**2. AIM AND SCOPE OF THE PRESENT 4**

**INVESTIGATION**

2.1 Aim and Scope 4

2.2 Vectors and assignments 5

2.3 Generating regular sequences 5

2.4 R Variables 7

2.5 Variable assignment 8

2.6 Data type of a variable 9

**3. EXPERIMENTAL OR MATERIALS AND METHODS;**

**ALGORITHMS USED 13**

3.1 Operators 13

3.1.1 Arithmetic Operators 13

3.1.2 Logical Operators 14

3.2 Looping 16

3.2.1 Looping Control Statements 17

3.3 Functions 17

3.3.1 Function Definition 18

3.3.2 Function Components 18

3.3.3 Built-in Function 18

3.3.4 User Defined Function 19

3.3.5 Calling a Function 20

3.4 CSV 22

3.4.1 Input a CSV 22

3.4.2 Analysing the CSV 24

3.5 Packages 26

**4. RESULTS AND DISCUSSSION, PERFORMANCE**

**ANALYSIS 27**

4.1 Importing 27

4.2 Exporting 27

4.3 Bar chart and Histogram in R 28

4.4 Pie Chart 29

4.4.1 Simple Pie Chart 29

4.4.2 3D Pie Chart 31

4.5 Line graph 33

**5. SUMMARY AND CONCLUSIONS 34**

5.1 Summary 34

5.2 Conclusion 34

**REFERENCES 35**

**APPENDIX 36**

A. SCREENSHOTS 36

B. SOURCE CODE 39

# LIST OF FIGURES

**FIGURE NO. FIGURE NAME PAGE NO.**

* 1. Why R 2
  2. R Console 3

1.3 Download R Studio 3

2.1 Data science for bussiness capability rating 6

2.2 Stack overflow traffic to programming languages 7

3.1 Looping 15

3.2 R visualization packages 26

4.1 Importing data into R 27

4.2 Pie chart 29

4.3 Pie chart with percentage 30

4.4 3D pie chart 31

4.5 R-Line Graph 33

# LIST OF TABLES

## TABLE NO. TABLE NAME PAGE NO.

2.1 R variables 8

3.1 Loop type and Description 16

3.2 Control statement &Description 17

# LIST OF ABBREVATIONS

**ABBREVATION EXPANSION**

CSV Comma Seperated Values

SAS Syntacticcally Awsome Style Sheet

**CHAPTER 1**

**INTRODUCTION**

R is a programming language and environment for statistical computing and graphics. It is a GNU Projectwhich is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

* 1. **What is R?**

R is a programming language developed by Ross Ihaka and Robert Gentleman in 1993. R possesses an extensive catalog of statistical and graphical methods. It includes machine learning algorithm, linear regression, time series, statistical inference to name a few. Most of the R libraries are written in R, but for heavy computational task, C, C++ and Fortran codes are preferred.

R is not only entrusted by academic, but many large companies also use R programming language, including Uber, Google, Airbnb, Facebook and so on.

Data analysis with R is done in a series of steps; programming, transforming, discovering, modeling and communicate the results:

* **Program**: R is a clear and accessible programming tool
* **Transform**: R is made up of a collection of libraries designed specifically for data science
* **Discover**: Investigate the data, refine your hypothesis and analyze them
* **Model**: R provides a wide array of tools to capture the right model for your data

**Communicate**: Integrate codes, graphs, and outputs to a report with R Markdown or build Shiny apps to share with the world.

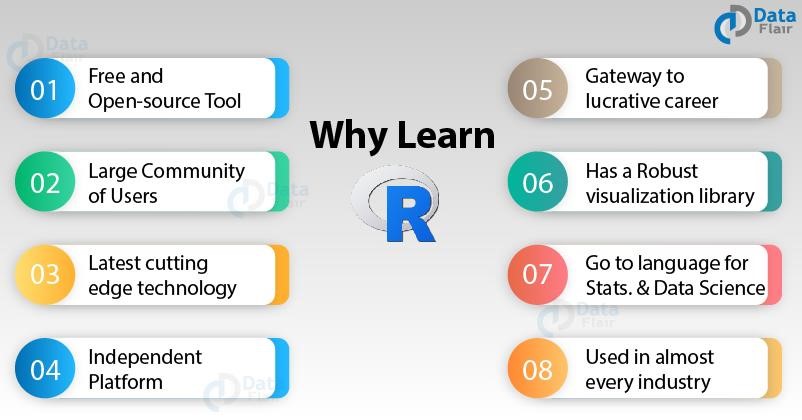
* 1. **Why R & What is the use of R?**

***1.2.1 Why R?***

The questions which arise in the mind of every R aspirant before starting to learn R Programming is – Why Learn R?

R is the most popular language in the world of Data Science.

It is heavily used in analyzing data that is both structured and unstructured. This has made R, the standard language for performing statistical operations. R allows various features that set it apart from other Data Science languages. In this article, we will explain why you must learn R and how it will benefit you in the domain of Data Science.



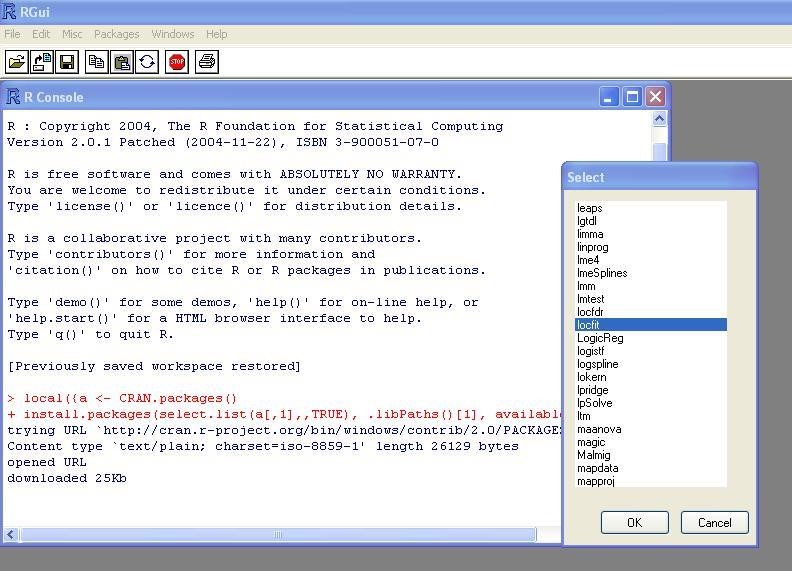
***Fig 1.1***

R is an open-source language. It is maintained by a community of active users and you can avail R for free. You can modify various functions in R and make your own packages. Since R is issued under the **General Public Licence (GNU)**, there are no restrictions on its usage.

***1.2.2 What is the use of R?***

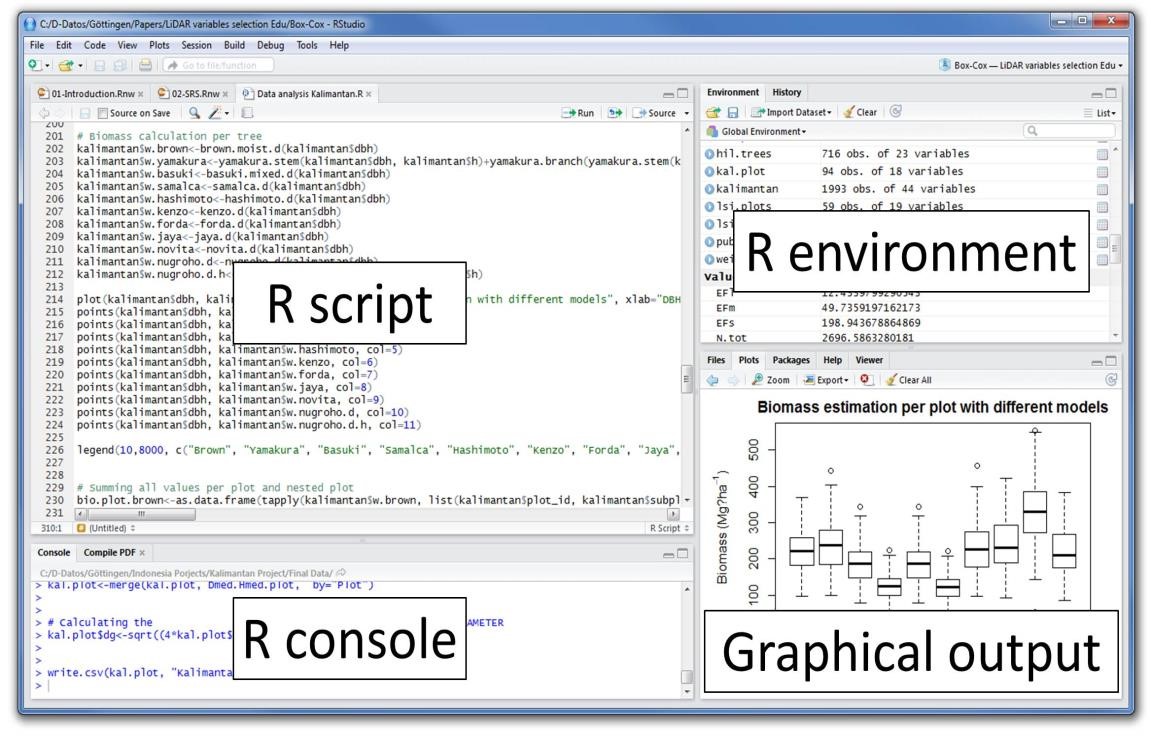
* Statistical inference
* Data analysis
* Machine learning algorithm
  1. **Installation of R?**

***1.3.1 Download R***



***Fig 1.2***

***1.3.2 Download R Studio***



***Fig 1.3***

**CHAPTER 2**

**AIM AND SCOPE OF THE PRESENT INVESTIGATION**

**2.1 Aim and Scope**

To outline all of the key main areas and will explain everything from the basics. It assumes that the reader is not familiar or has a beginner level understanding of the programming language R.

The field of data science and quantitative development requires us to constantly adapt and learn new skills because of its highly dynamic and demanding nature. There comes a time in a data scientist's professional life when it becomes important to learn more than one programming language. Subsequently, I have chosen to learn R.

**2.2 Vectors and assignment**

R operates on named data structures. The simplest such structure is the numeric vector, which is a single entity consisting of an ordered collection of numbers. To set up a vector named x, say, consisting of five numbers, namely 10.4, 5.6, 3.1, 6.4 and 21.7, use the R command > x <- c(10.4, 5.6, 3.1, 6.4, 21.7) This is an assignment statement using the function c() which in this context can take an arbitrary number of vector arguments and whose value is a vector got by concatenating its arguments end to end.1 A number occurring by itself in an expression is taken as a vector of length one. Notice that the assignment operator („<-‟), which consists of the two characters „ assign("x", c(10.4, 5.6, 3.1, 6.4, 21.7)) The usual operator, <-, can be thought of as a syntactic short-cut to this. Assignments can also be made in the other direction, using the obvious change in the assignment operator. So the same assignment could be made using > c(10.4, 5.6, 3.1, 6.4, 21.7) -> x If an expression is used as a complete command, the value is printed and lost 2 . So now if we were to use the command > 1/x the reciprocals of the five values would be printed at the terminal (and the value of x, of course, unchanged). The further assignment > y <- c(x, 0, x) would create a vector y with 11 entries consisting of two copies of x with a zero in the middle place.

Example program:

x=20

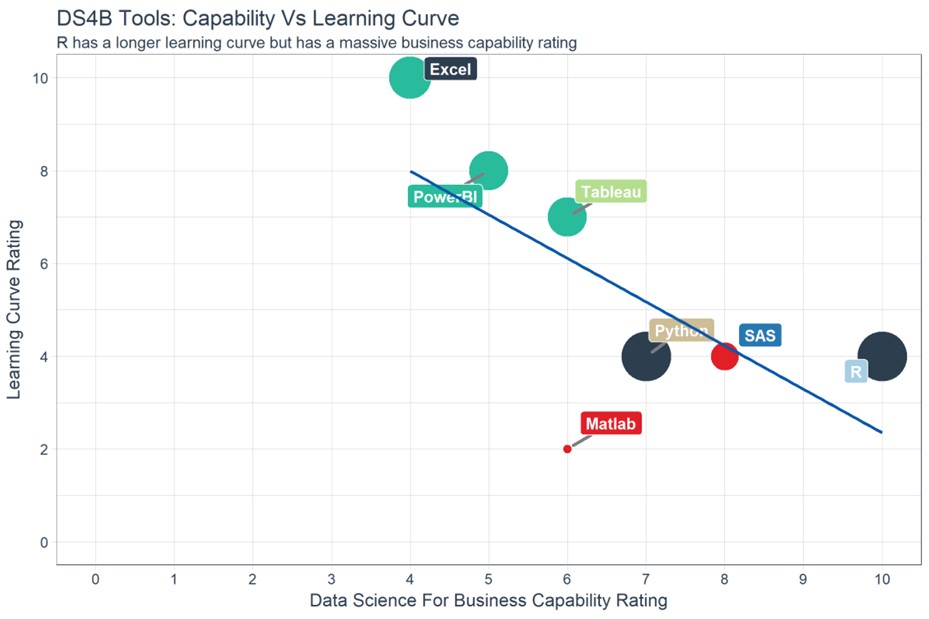
y<-"Artificial" "Intelligence"->z cat("x is ",var.1,"\n") cat("y is ",var.2,"\n") cat("z is ",var.3,"\n")

*Output :* x is 20

y is Artificial z is Intelligence

**2.3 Generating regular sequences**

R has a number of facilities for generating commonly used sequences of numbers. For example 1:30 is the vector c(1, 2, ..., 29, 30). The colon operator has high priority within an expression, so, for example 2\*1:15 is the vector c(2, 4, ..., 28, 30). Put n <- 10 and compare the sequences 1:n-1 and 1:(n-1). The construction 30:1 may be used to generate a sequence backwards. The function seq() is a more general facility for generating sequences. It has five arguments, only some of which may be specified in any one call. The first two arguments, if given, specify the beginning and end of the sequence, and if these are the only two arguments given the result is the same as the colon operator. That is seq(2,10) is the same vector as 2:10. Arguments to seq(), and to many other R functions, can also be given in named form, in which case the order in which they appear is irrelevant. The first two arguments may be named from=value and to=value; thus seq(1,30), seq(from=1, to=30) and seq(to=30, from=1) are all the same as 1:30. The next two arguments to seq() may be named by=value and length=value, which specify a step size and a length for the sequence respectively. If neither of these is given, the default by=1 is assumed. For example > seq(-5, 5, by=.2) -> s3 generates in s3 the vector c(-5.0, -4.8, -4.6, ..., 4.6, 4.8, 5.0). Similarly > s4 <- seq(length=51, from=-5, by=.2)

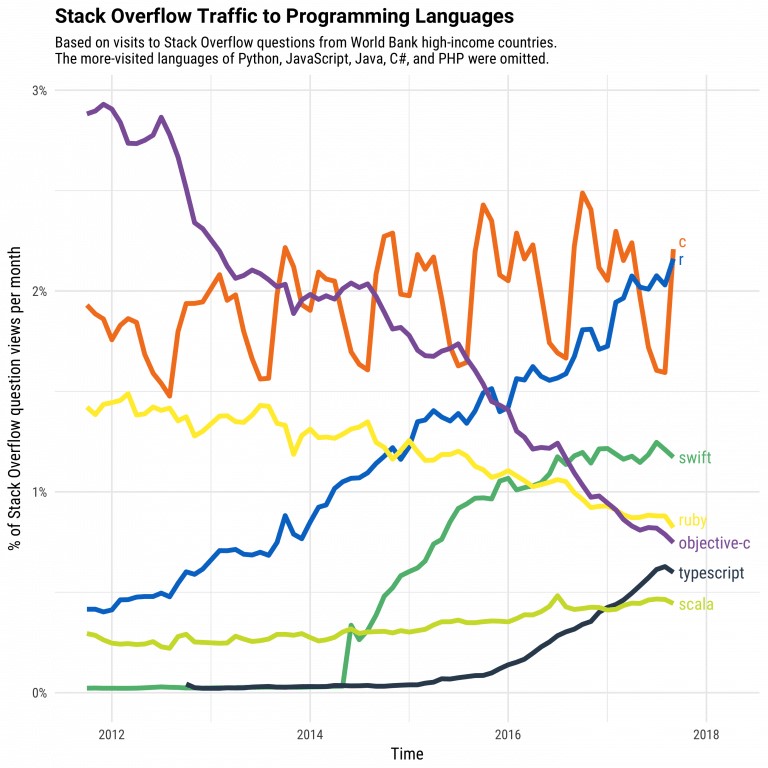


***Fig 2.1***

On the top left of the graph, you can see Excel and PowerBI.These two tools are simple to learn but don't offer outstanding business capability, especially in term of modeling. In the middle, you can see Python and SAS. SAS is a dedicated tool to run a statistical analysis for business, but it is not free. SAS is a click and run software. Python, however, is a language with a monotonous learning curve. Python is a fantastic tool to deploy Machine Learning and AI but lacks communication features. With an identical learning curve, R is a good trade-off between implementation and data analysis.

When it comes to data visualization (DataViz), you'd probably heard about Tableau. Tableau is, without a doubt, a great tool to discover patterns through graphs and charts. Besides, learning Tableau is not time-consuming. One big problem with data visualization is you might end up never finding a pattern or just create plenty of useless charts. Tableau is a good tool for quick visualization of the data or Business Intelligence. When it comes to statistics and decision-making tool, R is more appropriate.

Stack Overflow is a big community for programming languages. If you have a coding issue or need to understand a model, Stack Overflow is here to help. Over the year, the percentage of question-views has increased sharply for R compared to the other languages. This trend is of course highly correlated with the booming age of data science but, it reflects the demand of R language for data science.



***Fig 2.2***

In data science, there are two tools competing with each other. R and Python are probably the programming language that defines data science.

**2.4** **R Variables**

A variable provides us with named storage that our programs can manipulate. A variable in R can store an atomic vector, group of atomic vectors or a combination of many Robjects. A valid variable name consists of letters, numbers and the dot or underline characters. The variable name starts with a letter or the dot not followed by a number.

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Validity** | **Reason** |
| var\_name2. | valid | Has letters, numbers, dot and underscore |
| var\_name% | Invalid | Has the character '%'. Only dot(.) and underscore allowed. |
| 2var\_name | invalid | Starts with a number |
| .var\_name, var.name | valid | Can start with a dot(.) but the dot(.)should not be followed by a number. |
| .2var\_name | invalid | The starting dot is followed by a number making it invalid. |
| \_var\_name | invalid | Starts with \_ which is not valid |

***Table 2.1***

**2.5 Variable Assignment**

The variables can be assigned values using leftward, rightward and equal to operator. The values of the variables can be printed using **print()** or **cat()** function. The **cat()** function combines multiple items into a continuous print output.

|  |
| --- |
| # Assignment using equal operator. var.1 = c(0,1,2,3)    # Assignment using leftward operator.  var.2 <- c("learn","R")    # Assignment using rightward operator.  c(TRUE,1) -> var.3    print(var.1)  cat ("var.1 is ", var.1 ,"\n") cat ("var.2 is ", var.2 ,"\n") cat ("var.3 is ", var.3 ,"\n") |

When we execute the above code, it produces the following result-

[1] 0 1 2 3 var.1 is 0 1 2 3 var.2 is learn R var.3 is 1 1

Note − The vector c(TRUE,1) has a mix of logical and numeric class. So logical class is coerced to numeric class making TRUE as 1.

**2.6 Data Type of a Variable**

In R, a variable itself is not declared of any data type, rather it gets the data type of the R - object assigned to it. So R is called a dynamically typed language, which means that we can change a variable‟s data type of the same variable again and again when using it in a program.

var\_x <- "Hello"

cat("The class of var\_x is ",class(var\_x),"\n")

var\_x <- 34.5

cat(" Now the class of var\_x is ",class(var\_x),"\n")

var\_x <- 27L

cat(" Next the class of var\_x becomes ",class(var\_x),"\n")

When we execute the above code, it produces the following result-

The class of var\_x is character

Now the class of var\_x is numeric

Next the class of var\_x becomes integer

Finding Variables:

To know all the variables currently available in the workspace we use the **ls()** function. Also the ls() function can use patterns to match the variable names.

print(ls())

When we execute the above code, it produces the following result-

1. "my var"
2. "my\_new\_var"
3. "my\_var"
4. "var.1"
5. "var.2"
6. "var.3"
7. "var.name"
8. "var\_name2."
9. "var\_x"
10. "varname"

**Note** − It is a sample output depending on what variables are declared in your environment.

The ls() function can use patterns to match the variable names.

# List the variables starting with the pattern "var". print(ls(pattern = "var"))

When we execute the above code, it produces the following result-

1. "my var"
2. "my\_new\_var"
3. "my\_var"
4. "var.1"
5. "var.2"
6. "var.3"
7. "var.name"
8. "var\_name2."
9. "var\_x"
10. "varname"

The variables starting with **dot(.)** are hidden, they can be listed using "all.names = TRUE" argument to ls() function.

print(ls(all.name = TRUE))

When we execute the above code, it produces the following result-

1. ".cars"
2. ".Random.seed"
3. ".var\_name"
4. ".varname"
5. ".varname2"
6. "my var"
7. "my\_new\_var"
8. "my\_var"
9. "var.1"
10. "var.2"
11. "var.3"
12. "var.name"
13. "var\_name2."
14. "var\_x"

**CHAPTER 3**

**EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED**

**3.1 Operators**

R's binary and logical operators will look very familiar to programmers. Note that binary operators work on vectors and matrices as well as scalars.

***3.1.1 Arithmetic Operators***

|  |  |
| --- | --- |
| **Operator** | Description |
| **+** | Addition |
| **-** | Subtraction |
| **\*** | Multiplication |
| **/** | Division |
| **^ or \*\*** | Exponentiation |
| **x %% y** | modulus (x mod y) 5%%2 is 1 |
| **x %/% y** | integer division 5%/%2 is 2 |

***3.1.2 Logical Operators***

|  |  |
| --- | --- |
| **Operator** | **Description** |
| **<** | less than |
| **<=** | less than or equal to |
| **>** | greater than |
| **>=** | greater than or equal to |
| **==** | exactly equal to |
| **!=** | not equal to |
| **!x** | Not x |
| **x | y** | x OR y |
| **x & y** | x AND y |
| **isTRUE(x)** | test if X is TRUE |

Example

# An example

x <

-

c(1:10

)

<5)]

x[(x>8) | (x

# yields 1 2

3

4

9

10

# How it w

orks

x <

-

c(1:10

)

x

0

1

2 3 4 5 6 7 8 9

1

x > 8

F F

F F F F F F T T

x < 5

T T T T F F F F F F

x > 8 | x < 5

T T T T F F

F F T T

x[c(T,T,T,T,

F,F,F,F,T,T)]

1

2 3 4 9

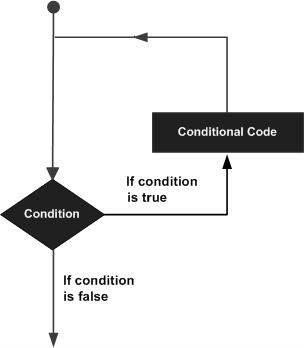
1

0

**3.2 LOOPING**

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times and the following is the general form of a loop statement in most of the programming languages −



***Fig 3.1***

R programming language provides the following kinds of loop to handle looping requirements. Click the following links to check their detail.

|  |  |
| --- | --- |
| **Sr.No.** | **Loop Type & Description** |
| 1 | [repeat loop](https://www.tutorialspoint.com/r/r_repeat_loop.htm)  Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable. |
| 2 | [while loop](https://www.tutorialspoint.com/r/r_while_loop.htm)  Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body. |
| 3 | [for loop](https://www.tutorialspoint.com/r/r_for_loop.htm)  Like a while statement, except that it tests the condition at the end of the loop body. |

***Table 3.1***

***3.2.1 Loop Control Statements***

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

R supports the following control statements. Click the following links to check their detail.

|  |  |
| --- | --- |
| **Sr.No.** | **Control Statement & Description** |
| 1 | [break statement](https://www.tutorialspoint.com/r/r_break_statement.htm)  Terminates the **loop** statement and transfers execution to the statement immediately following the loop. |
| 2 | [Next statement](https://www.tutorialspoint.com/r/r_next_statement.htm)  The **next** statement simulates the behavior of R switch. |

***Table 3.2***

**3.3 FUNCTIONS**

**A** function is a set of statements organized together to perform a specific task. R has a large number of in-built functions and the user can create their own functions.

In R, a function is an object so the R interpreter is able to pass control to the function, along with arguments that may be necessary for the function to accomplish the actions.

The function in turn performs its task and returns control to the interpreter as well as any result which may be stored in other objects.

***3.3.1 Function Definition***

An R function is created by using the keyword **function**. The basic syntax of an R function definition is as follows −

function\_name <- function(arg\_1, arg\_2, ...) {

Function body

}

***3.3.2 Function Components***

The different parts of a function are −

* **Function Name** − This is the actual name of the function. It is stored in R environment as an object with this name.
* **Arguments** − An argument is a placeholder. When a function is invoked, you pass a value to the argument. Arguments are optional; that is, a function may contain no arguments. Also arguments can have default values.
* **Function Body** − The function body contains a collection of statements that defines what the function does.
* **Return Value** − The return value of a function is the last expression in the function body to be evaluated.

R has many **in-built** functions which can be directly called in the program without defining them first. We can also create and use our own functions referred as **user defined** functions.

***3.3.3 Built-in Function***

Simple examples of in-built functions are **seq()**, **mean()**, **max()**, **sum(x)** and **paste(...)** etc. They are directly called by user written programs. You can refer [most widely used R functions.](https://cran.r-project.org/doc/contrib/Short-refcard.pdf)

# Create a sequence of numbers from 32 to 44.

print(seq(32,44))

# Find mean of numbers from 25 to 82. print(mean(25:82))

# Find sum of numbers frm 41 to 68. print(sum(41:68))

When we execute the above code, it produces the following result-

[1] 32 33 34 35 36 37 38 39 40 41 42 43 44

[1] 53.5

[1] 1526

***3.3.4 User-defined Function***

We can create user-defined functions in R. They are specific to what a user wants and once created they can be used like the built-in functions. Below is an example of how a function is created and used.

|  |
| --- |
| # Create a function to print squares of numbers in sequence.  new.function <- function(a) { for(i in 1:a) { b <- i^2 print(b)  } |

}

***3.3.5 Calling a Function***

|  |
| --- |
| # Create a function to print squares of numbers in sequence. new.function <- function(a) { for(i in 1:a) { b <- i^2 print(b)  }  }  # Call the function new.function supplying 6 as an argument. new.function(6) |

When we execute the above code, it produces the following result-

[1] 1

[1] 4

[1] 9

[1] 16

[1] 25

[1] 36

Calling a Function without an Argument :

|  |
| --- |
| # Create a function without an argument.  new.function <- function() { for(i in 1:5) { print(i^2)  }  }    # Call the function without supplying an argument.  new.function() |

When we execute the above code, it produces the following result-

[1] 1

[1] 4

[1] 9

[1] 16

[1] 25

Calling a Function with Argument Values :

The arguments to a function call can be supplied in the same sequence as defined in the function or they can be supplied in a different sequence but assigned to the names of the arguments.

|  |
| --- |
| # Create a function with arguments.  new.function <- function(a,b,c) {  result <- a \* b + c print(result)  }  # Call the function by position of arguments. new.function(5,3,11)    # Call the function by names of the arguments. new.function(a = 11, b = 5, c = 3) |

When we execute the above code, it produces the following result-

[1] 26

[1] 58

Calling a Function with Default Argument :

We can define the value of the arguments in the function definition and call the function without supplying any argument to get the default result. But we can also call such functions by supplying new values of the argument and get non default result.

# Create a function with arguments. new.function <- function(a = 3, b = 6) {

result <- a \* b print(result)

}

# Call the function without giving any argument. new.function()

# Call the function with giving new values of the argument. new.function(9,5)

When we execute the above code, it produces the following result-

[1] 18

[1] 45

**3.4 CSV**

***3.4.1 Input as CSV File***

The csv file is a text file in which the values in the columns are separated by a comma. Let's consider the following data present in the file named **input.csv**.

You can create this file using windows notepad by copying and pasting this data. Save the file as **input.csv** using the save As All files(\*.\*) option in notepad.

id,name,salary,start\_date,dept

1,Rick,623.3,2012-01-01,IT

2,Dan,515.2,2013-09-23,Operations

3,Michelle,611,2014-11-15,IT

4,Ryan,729,2014-05-11,HR

5,Gary,843.25,2015-03-27,Finance

6,Nina,578,2013-05-21,IT

7,Simon,632.8,2013-07-30,Operations

8,Guru,722.5,2014-06-17,Finance

Reading a CSV File

Following is a simple example of **read.csv()** function to read a CSV file available in your current working directory –

data <- read.csv("input.csv")

print(data)

When we execute the above code, it produces the following result- id, name, salary, start\_date, dept

1. 1 Rick 623.30 2012-01-01 IT
2. 2 Dan 515.20 2013-09-23 Operations
3. 3 Michelle 611.00 2014-11-15 IT
4. 4 Ryan 729.00 2014-05-11 HR
5. NA Gary 843.25 2015-03-27 Finance
6. 6 Nina 578.00 2013-05-21 IT
7. 7 Simon 632.80 2013-07-30 Operations
8. 8 Guru 722.50 2014-06-17 Finance

***3.4.2 Analyzing the CSV File***

By default the **read.csv()** function gives the output as a data frame. This can be easily checked as follows. Also we can check the number of columns and rows.

data <- read.csv("input.csv") print(is.data.frame(data)) print(ncol(data)) print(nrow(data))

When we execute the above code, it produces the following result-

[1] TRUE

[1] 5

[1] 8

Once we read data in a data frame, we can apply all the functions applicable to data frames as explained in subsequent section.

**Example :** Get the maximum salary

|  |
| --- |
| # Create a data frame. data <- read.csv("input.csv")    # Get the max salary from data frame.  sal <- max(data$salary)  print(sal) |

When we execute the above code, it produces the following result-

[1] 843.25

**Example :** Get the details of the person with max salary

We can fetch rows meeting specific filter criteria similar to a SQL where clause.

# Create a data frame.

data <- read.csv("input.csv")

# Get the max salary from data frame.

sal <- max(data$salary)

# Get the person detail having max salary. retval <- subset(data, salary == max(salary)) print(retval)

When we execute the above code, it produces the following result-

Id name salary start\_date dept

5 NA Gary 843.2 2015-03-27 Finance

What is R Data Visualization ?

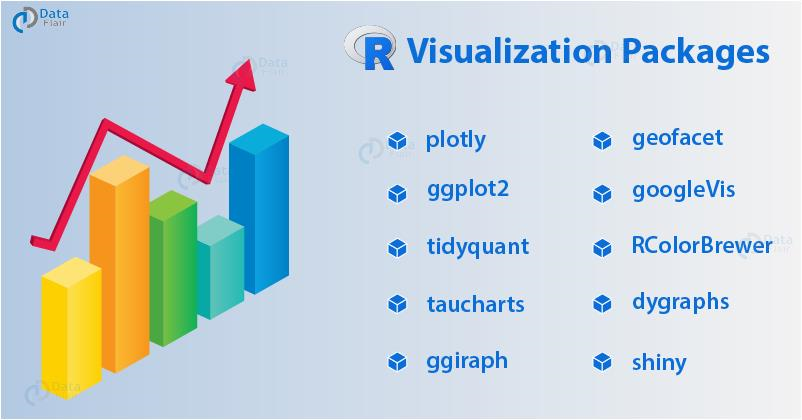
Using the diverse functionalities provided by R, one can create visually appealing data visualizations with only a few lines of code. Data visualization is an efficient technique of gaining insights about data through a visual medium.

* With the help of visualization techniques, humans can easily gain insights about the hidden patterns in data which might otherwise be neglected.
* Using data visualization, one can work with large datasets to efficiently obtain key insights about it.

**3.5 PACKAGES**

**R Visualization Packages :**

Following are some of the essential visualization packages in R Programming:



***Fig 3.2***

**Use of R Programming :**

For most of our work in R Programming, we will use the environment RStudio.

RStudio of R has four panels:

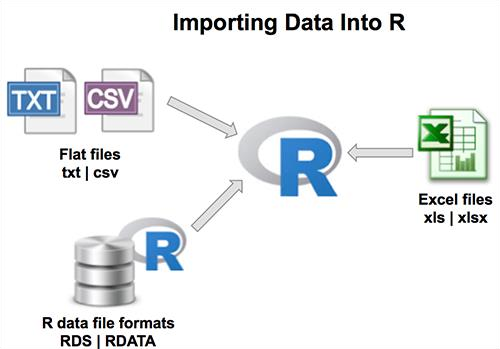
* **Console –** This is the actual R window, you can enter R commands here. And, thus execute them by pressing enter.
* **Source –** This is where we can edit scripts. It is where you should always be working. Control-enter sends selected codes to console.
* **Plots/Help –** Here plots and help pages will be shown.
* **Workspace –** Shows which objects you currently have.

Anything following a **# symbol** is treated as a comment.

**CHAPTER 4**

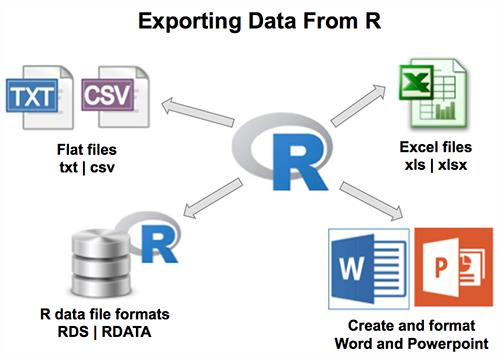
**RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS**

**4.1 IMPORTING**



**Fig 4.1**

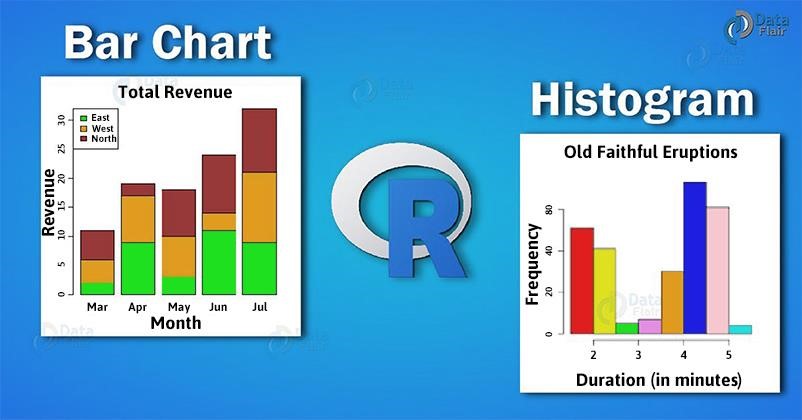
**4.2 EXPORTING**



**Fig 4.2**

**4.3 Bar Chart and Histogram in R**

The most popular way of representing the data, that is, in the form of a bar chart and histogram. First, we will explore the concept of data visualization and data set, then learn about the central tendency measures and also understand the types of bar charts with examples.



***Fig 4.1***

**Data Set in R :**

A data set is defined as the field in the following data. Thus, it provides a number of sample data sets as there are thousands of data sets present. We can use them to practice and get better at machine learning.

Although, in *U.C.I Machine Learning Repository*, most of data sets are hosted for free. These data sets are very useful as they are small, well behaved and well understood.

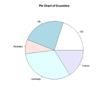
* It can download easily and fast.
* One can fit them into memory easily.
* You can run algorithms on them quickly.

**4.4 Pie Charts**

Pie charts are not recommended in the R documentation, and their features are somewhat limited. The authors recommend bar or dot plots over pie charts because people are able to judge length more accurately than volume. Pie charts are created with the function **pie(***x***, labels=)** where *x* is a non-negative numeric vector indicating the area of each slice and labels= notes a character vector of names for the slices.

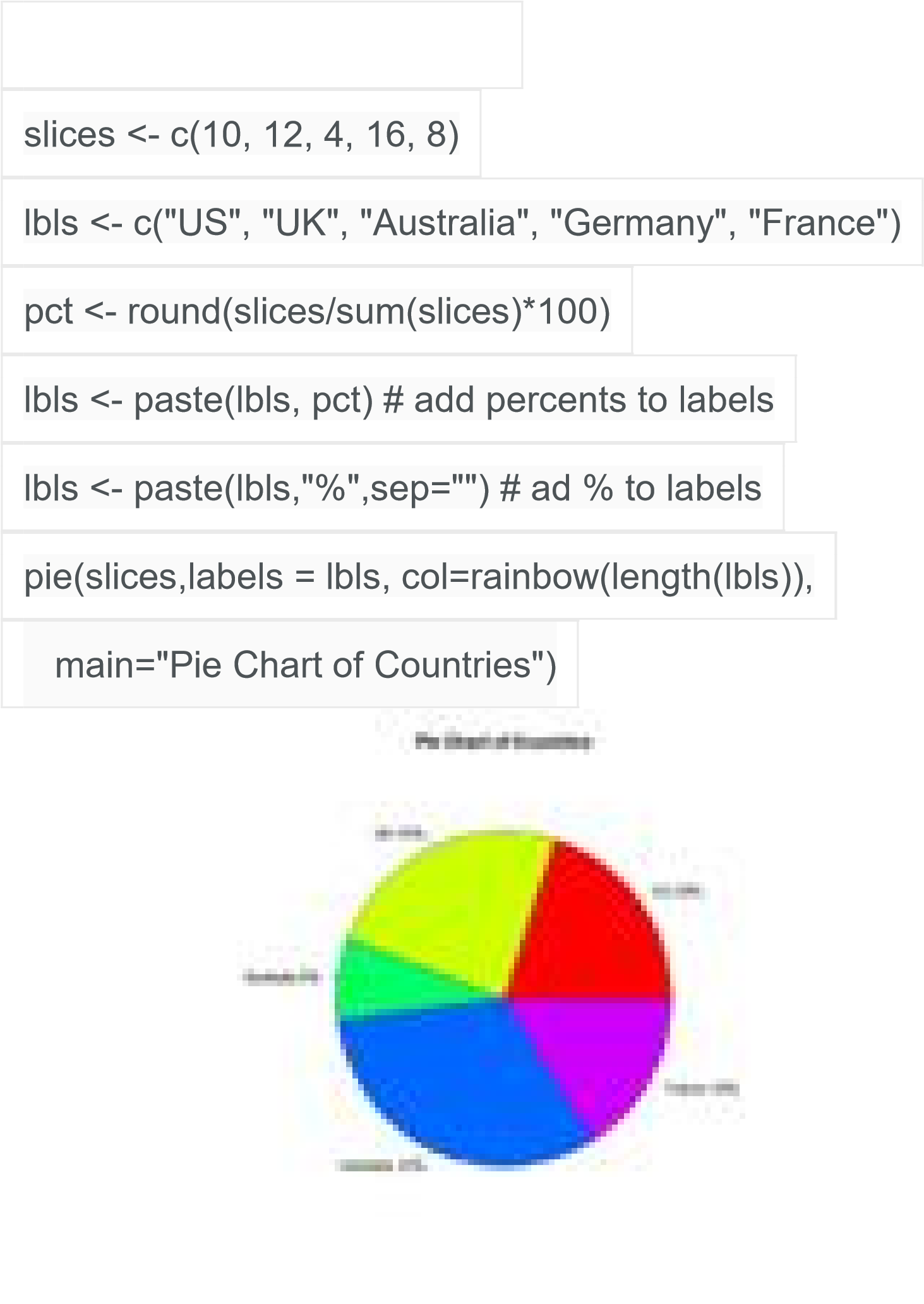
***4.4.1 Simple Pie Chart :***

# Simple Pie Chart slices <- c(10, 12,4, 16, 8) lbls <- c("US", "UK", "Australia", "Germany", "France") pie(slices, labels = lbls, main="Pie Chart of Countries")



***Fig 4.2***

***Pie Chart with Annotated Percentages :***



***Fig 4.3***

# Pie Chart with Percentages

***4.4.2 3D Pie Chart***

The **pie3D( )** function in the [plotrix](http://cran.r-project.org/web/packages/plotrix/index.html) package provides 3D exploded pie charts.

#

3

D

Exploded

Pie

Chart

library(plotrix)

slices

<

-

c(10,

12

,

4

,

16

,

8)

lbls

<

-

c("US",

"UK",

"Australia",

"Germany",

"France")

pie3D(slices,labels=lbls,explode=0.1,

main="Pie

Chart

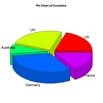
of

Count

ries

"

)



**Fig 4.4**

**4.5 R-LINE GRAPH**

A line chart is a graph that connects a series of points by drawing line segments between them. These points are ordered in one of their coordinate (usually the x-coordinate) value. Line charts are usually used in identifying the trends in data.

The **plot()** function in R is used to create the line graph.

**Syntax :**

The basic syntax to create a line chart in R is

plot(v,type,col,xlab,ylab)

Following is the description of the parameters used −  **v** is a vector containing the numeric values.

* **type** takes the value "p" to draw only the points, "l" to draw only the lines and "o" to draw both points and lines.
* **xlab** is the label for x axis.
* **ylab** is the label for y axis.
* **main** is the Title of the chart.
* **col** is used to give colors to both the points and lines.

**Example :**

A simple line chart is created using the input vector and the type parameter as "O". The below script will create and save a line chart in the current R working directory.

# Create the data for the chart.

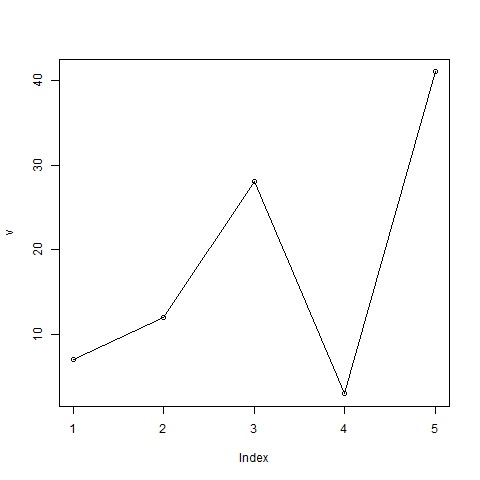
v <- c(7,12,28,3,41)

# Give the chart file a name. png(file = "line\_chart.jpg")

# Plot the bar chart. plot(v,type = "o")

# Save the file. dev.off()

When we execute the above code, it produces the following result-



***Fig 4.5***

**CHAPTER 5**

**SUMMARY AND CONCLUSIONS**

**5.1 SUMMARY**

R is a language and environment for statistical computing and graphics. It is a GNU Projectwhich is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

**5.2 CONCLUSIONS**

Programming your statistical analyses leads to a flexible, reproducible and time-saving workflow, in comparison to more traditional point-and-click focused applications. R is probably the best programming language around for applied statistics, because it has a large user base and many user-contributed packages that make your life easier. While it may take an hour or so to get acquainted with R, after initial difficulty it is easy to use, and provides a fast and reliable platform for data wrangling, visualization, modeling, and statistical testing.

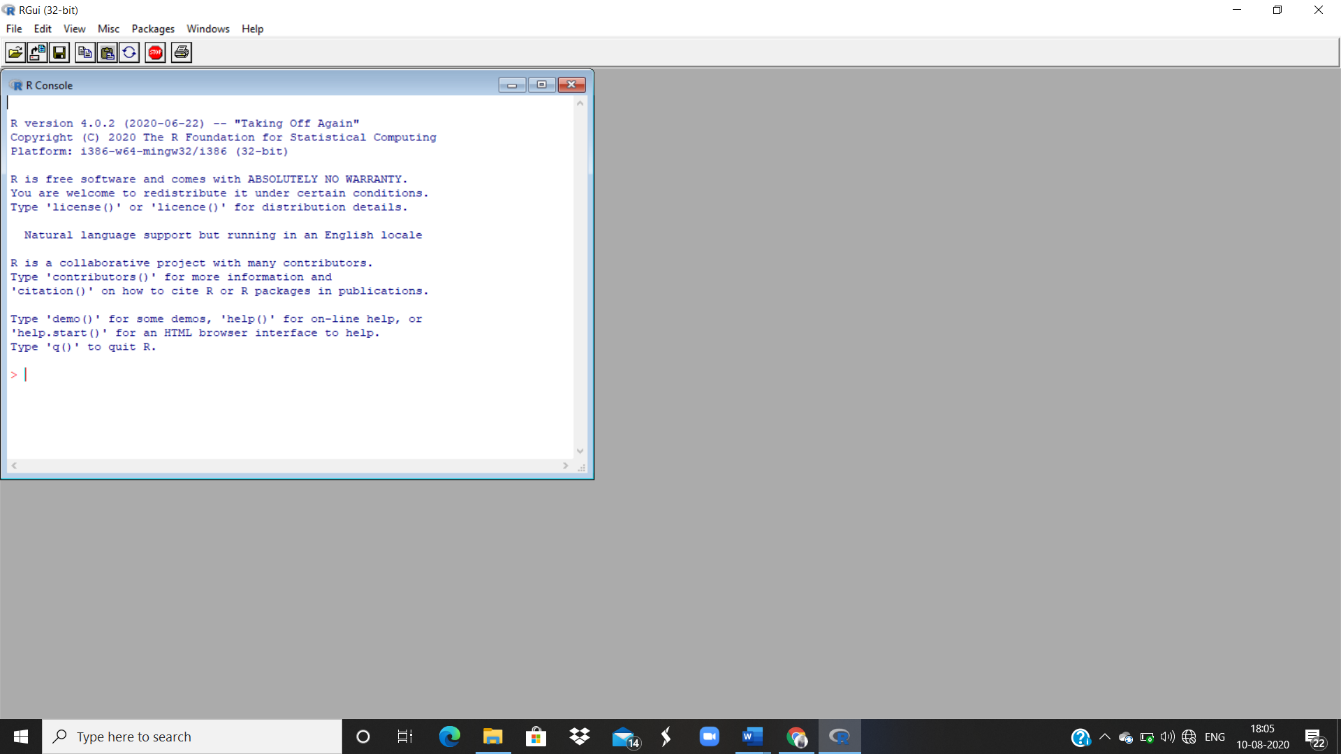
Finally, learning to code is not about having a superhuman memory for function names, but instead it is about developing a programmer’s mindset: Think your problem through and decompose it to small chunks, then ask a computer to do those chunks for you. Do that a couple of times and you will magically have memorized, as a byproduct, the names of a few common functions. You learn to code not by reading and memorizing a tutorial, but by writing it out, examining the output, changing the input and figuring out what changed in the output. Even better, you’ll learn the most once you use code to examine your own data, data that you know and care about. Hopefully, you’ll be now able to begin doing just that.

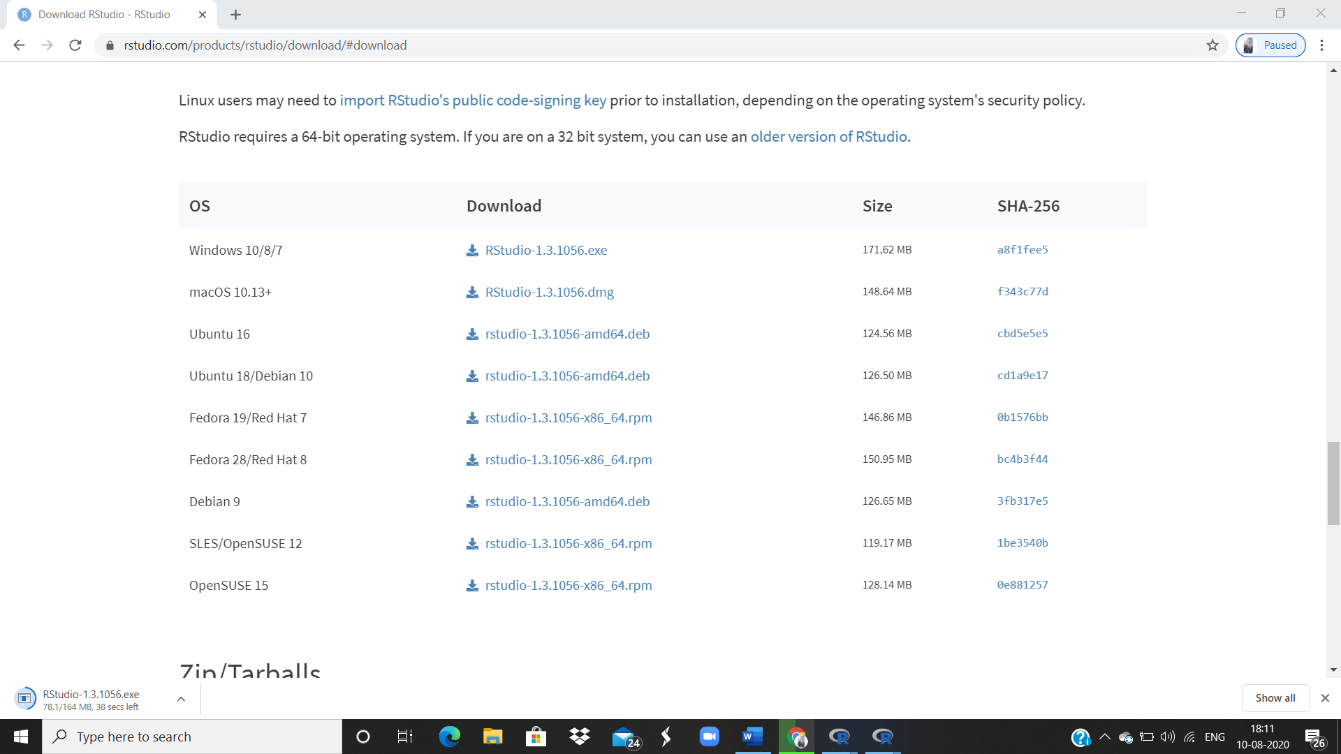
**REFERENCE**

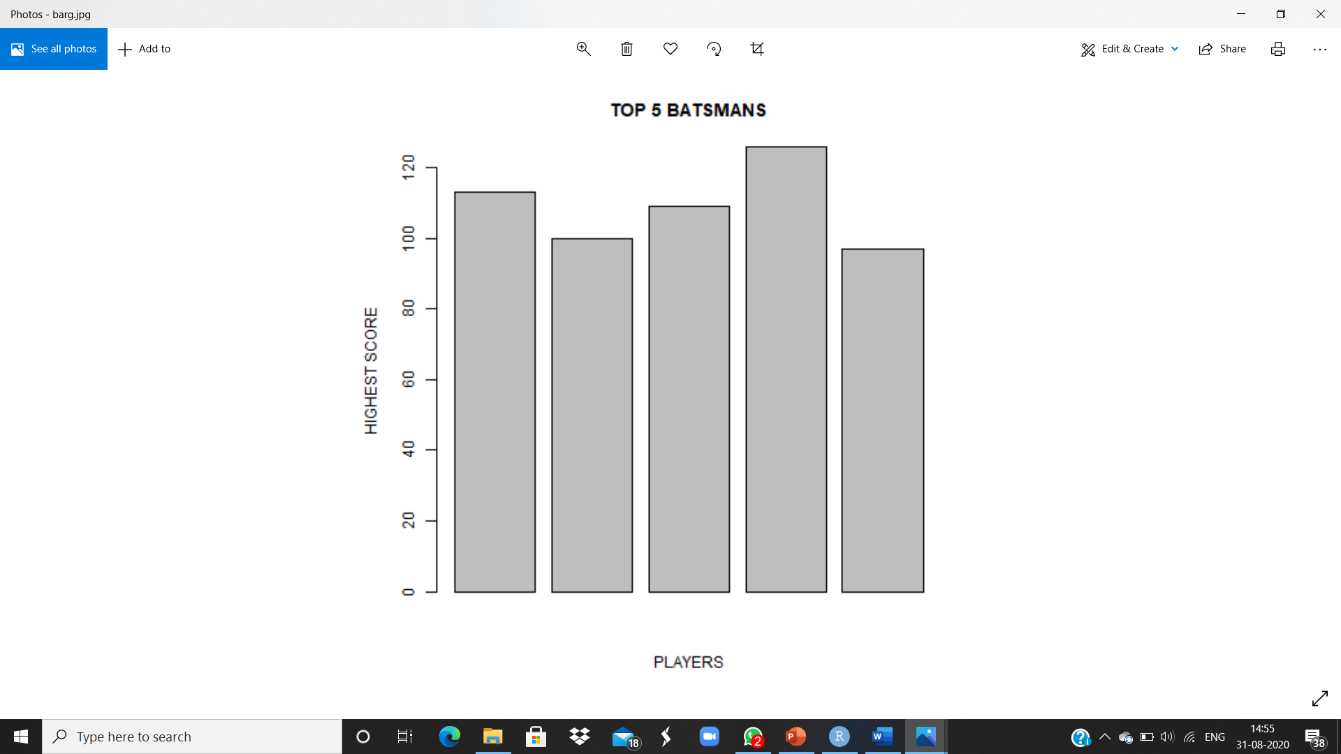
1. Buerkner, P.-C. (2016). *Brms: Bayesian regression models using stan*. Retrieved from [http://CRAN.R-project.org/package=brms](http://cran.r-project.org/package=brms)
2. Fox, J. (2010). Introduction to statistical computing in r. Retrieved November 23, 2016, from <http://socserv.socsci.mcmaster.ca/jfox/Courses/R-course/index.html>
3. Phillips, N. (2016). Yarrr: A companion to the e-book “YaRrr!: The pirate’s guide to r”. Retrieved from [https://CRAN.R-project.org/package=yarrr](https://cran.r-project.org/package=yarrr)
4. Revelle, W. (2016a). Psych: Procedures for psychological, psychometric, and personality research. Evanston, Illinois: Northwestern University. Retrieved from [https://CRAN.R-project.org/package=psych](https://cran.r-project.org/package=psych)
5. Wickham, H. (2016). Tidyverse: Easily install and load ’tidyverse’ packages. Retrieved from [https://CRAN.R-project.org/package=tidyverse](https://cran.r-project.org/package=tidyverse)

**APPENDIX**

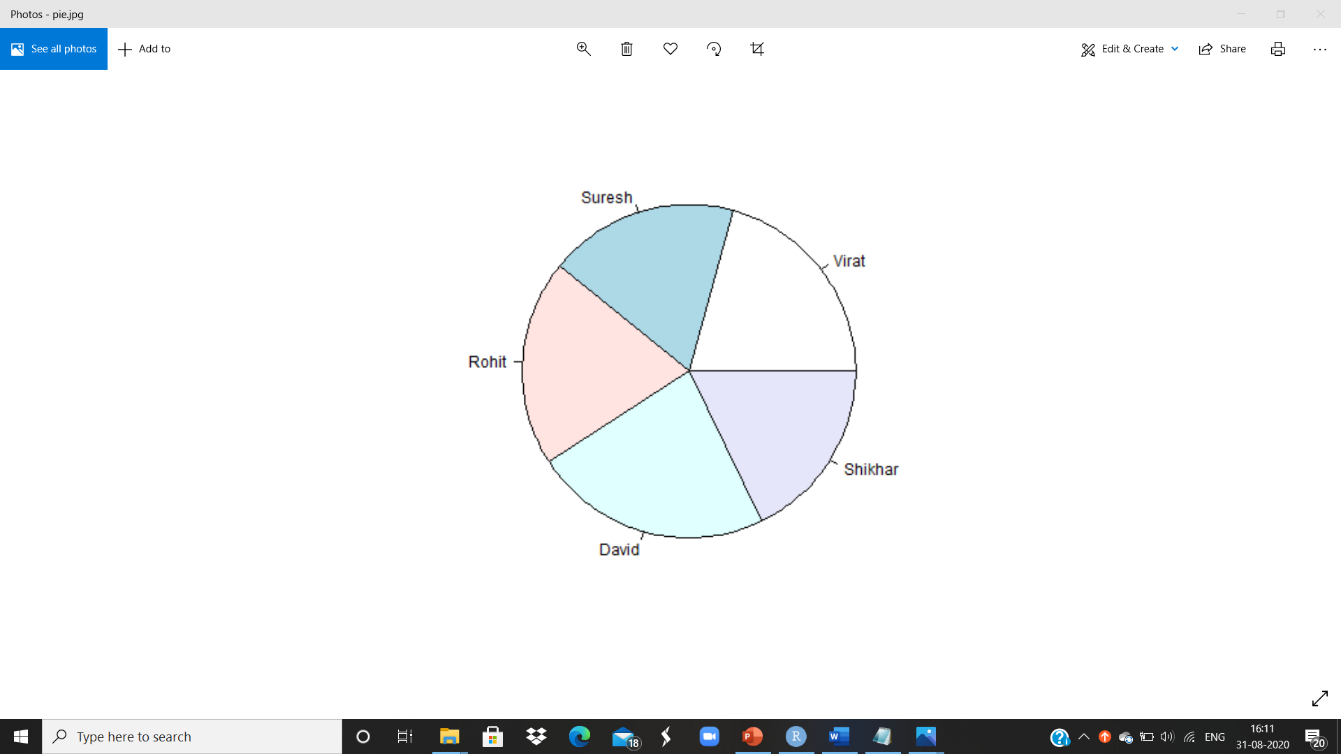
1. **SCREENSHOT**

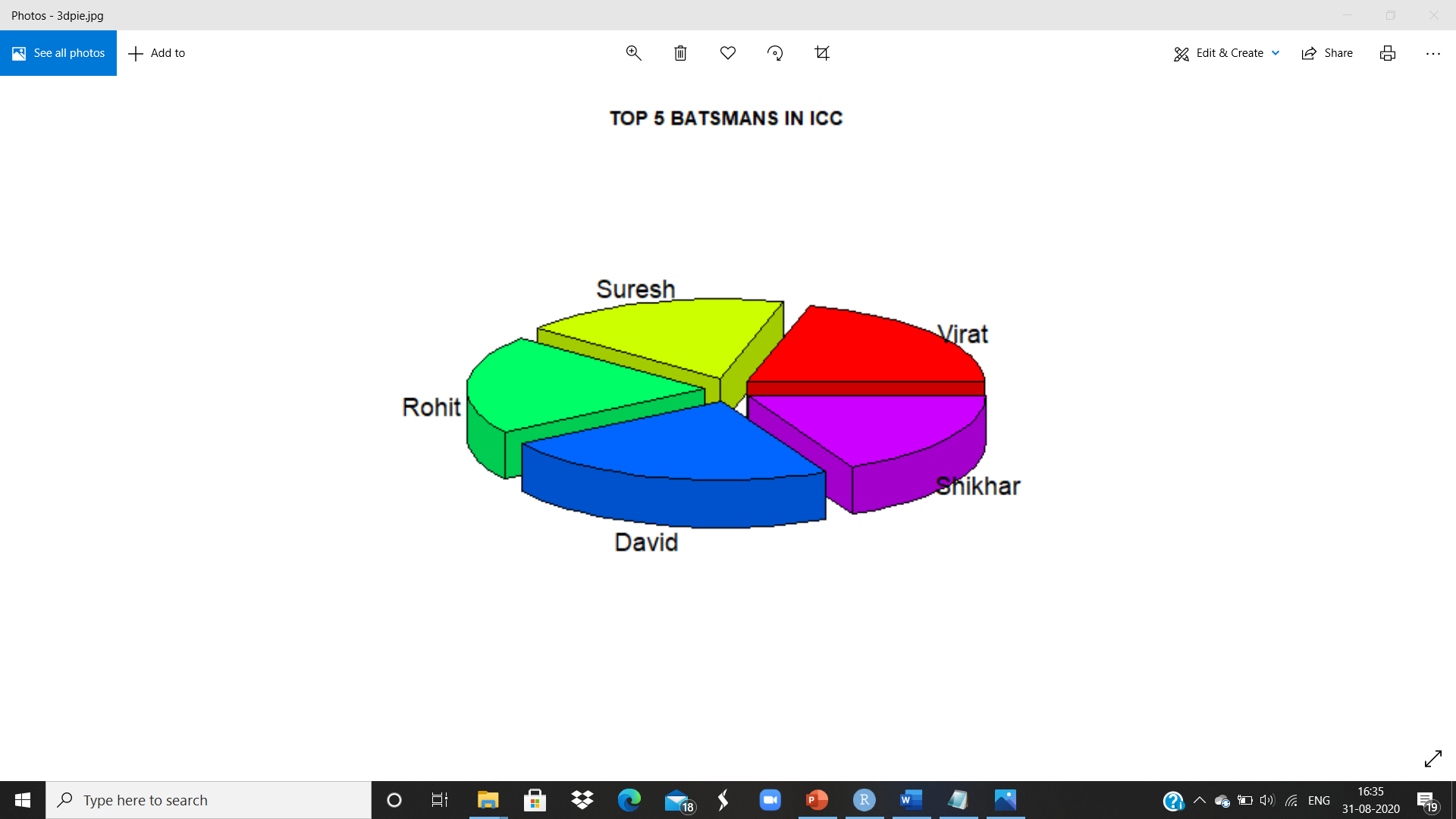
****

****

****

****

****



**SOURCE CODE**

**“TOP 100 BATSMANS IN ICC”**

* Download data from [www.kaggle.com](http://www.kaggle.com)
* Save the file using csv extension in the following path
* Import data in RStudio [Environment 🡪 Import Data Set 🡪 Browse 🡪 Import]

#get\_data

print(getwd())

#read\_data

data<-read.csv(“icc.csv”)

#print\_data

print(data)

#print\_data\_on\_particular\_column

Team<-(data$PLAYERS)

print(Team)

#no\_of\_columns

ncol(data)

#no\_of\_rows

nrow(data)

#substring

substr(Team,1,3)

#mean

mean(data$HS)

#median

median(data$Team)

#mode

mode(data$Avg)

#summary

summary(data)

#barchart

a<-c(113,100,109,126,97)

b<-c("Vir","Sur","Roh","Dav","Shi")

png(file=”barg.jpg”)

barplot(a,main="TOP 5 BATSMANS",xlab = "PLAYERS",ylab = "HIGHEST SCORE")

dev.off()

#histogram

a<-c(113,100,109,126,97)

png(file=”histogram.png”)

hist(a,main="TOP 5 BATSMANS",xlab = "HIGHEST SCORE",ylab = "PLAYERS")

dev.off()

#linechart

a<-c(data$HS)

b<-c(data$Team)

png(file=”linechart.jpg”)

plot(a,xlab=”PLAYERS”,ylab=”HS”,col=”green”,border=”red”,main=”Score Board”)

dev.off()

#piechart

a<-c(113,100,109,126,97)

b<-c("Vir","Sur","Roh","Dav","Shi")

png(file=”pie.jpg”)

pie(a,b)

dev.off()

#3D\_piechart

Install.packages(“plotrix”)

Library(plotrix)

a<-c(113,100,109,126,97)

b<-c("Vir","Sur","Roh","Dav","Shi")

png(file=”3dpie.jpg”)

pie3D(a,labels=b,explode=0.1,main="TOP 5 BATSMANS")

dev.off()