Index

Exp No: 1	Install, configure and run Hadoop and R
Date:	

AIM

To install, configure and run Hadoop and R.

PROCEDURE

- Hadoop Installation

Run the following command on ubuntu terminal:

1. Install Hadoop on ubuntu.
   a. sudo apt update
   b. sudo apt install openjdk-8-jdk -y
2. Check if java is installed
   a. java -version
   b. javac -version
3. Install SSH server
   a. sudo apt install openssh-server openssh-client -y
4. Create a new user in ubuntu
   a. sudo adduser hdoop
   b. sudo adduser hdoop sudo
   c. su – hdoop
5. Create rsa file
   a. ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
   b. cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
   c. chmod 0600 ~/.ssh/authorized_keys
6. Open ssh server
   a. ssh localhost
7. Download Hadoop
   a. wget https://downloads.apache.org/hadoop/common/hadoop-3.2.3/hadoop-3.2.3.tar.gz
   b. tar xzf hadoop-3.2.3.tar.gz
8. Edit bashrc
   a. sudo nano .bashrc
   b. Add the following lines at the end of the file
      i. export HADOOP_HOME=/home/hdoop/hadoop-3.2.3
      ii. export HADOOP_INSTALL=$HADOOP_HOME
      iii. export HADOOP_MAPRED_HOME=$HADOOP_HOME
      iv. export HADOOP_COMMON_HOME=$HADOOP_HOME
      v. export HADOOP_HDFS_HOME=$HADOOP_HOME
      vi. export YARN_HOME=$HADOOP_HOME

      vii. export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
      viii. export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
      ix. export HADOOP_OPTS"-Djava.library.path=$HADOOP_HOME/lib/nativ"
   c. source ~/.bashrc

9. Edit JAVA_HOME
   a. sudo nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh
   b. Add the following line at the end of the file
      i. export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

10. Edit core-site
   a. sudo nano $HADOOP_HOME/etc/hadoop/core-site.xml
   b. Add the following lines
      i. <property>
      ii. <name>hadoop.tmp.dir</name>
      iii. <value>/home/hdoop/tmpdata</value>
      iv. <description>A base for other temporary directories.</description>
      v. </property>
      vi. <property>
      vii. <name>fs.default.name</name>
      viii. <value>hdfs://localhost:9000</value>
      ix. <description>The name of the default file system></description>
      x. </property>

11. Edit hdfs-site
   a. sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml
   b. Add the following lines
      i. <property>
      ii. <name>dfs.data.dir</name>
      iii. <value>/home/hdoop/dfsdata/namenode</value>
      iv. </property>
      v. <property>
      vi. <name>dfs.data.dir</name>
      vii. <value>/home/hdoop/dfsdata/datanode</value>
      viii. </property>
      ix. <property>
      x. <name>dfs.replication</name>
      xi. <value>1</value>
      xii. </property>

12. Edit mapred-site
   a. sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
   b. Add the following lines
      i. <property>
      ii. <name>mapreduce.framework.name</name>
      iii. <value>yarn</value>
      iv. </property>

13. Edit yarn-site

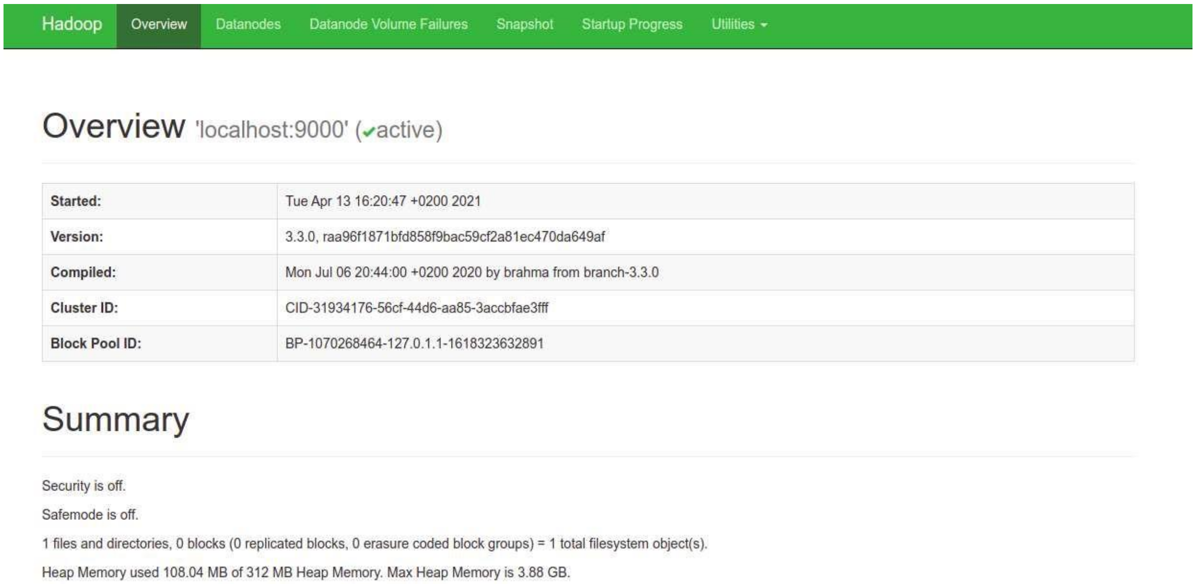
   a. sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml
   b. Add the following lines
      i. <property>
      ii. <name>yarn.nodemanager.aux-services</name>
      iii. <value>mapreduce_shuffle</value>
      iv. </property>
      v. <property>
      vi. <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
      vii. <value>org.apache.hadoop.mapred.ShuffleHandler</value>
      viii. </property>
      ix. <property>
      x. <name>yarn.resourcemanager.hostname</name>
      xi. <value>127.0.0.1</value>
      xii. </property>
      xiii. <property>
      xiv. <name>yarn.acl.enable</name>
      xv. <value>0</value>
      xvi. </property>
      xvii. <property>
      xviii. <name>yarn.nodemanager.env-whitelist</name>
      xix. <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLAS
      xx. SPATH_PERPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
      xxi. </property>

14. Launch Hadoop
    a. hdfs namenode -format
    b. cd $HADOOP_HOME/sbin
    c. start-all.sh
15. Go to browser
    a. localhost:8088
    b. localhost:9870

- Install R on windows

Download and install R for windows from - https://cran.r-project.org/bin/windows/base/

OUTPUT



Overview 'localhost:9000' (✓active)

Started:	Tue Apr 13 16:20:47 +0200 2021
Version:	3.3.0, raa96f1871bfd858f9bac59cf2a81ec470da649af
Compiled:	Mon Jul 06 20:44:00 +0200 2020 by brahma from branch-3.3.0
Cluster ID:	CID-31934176-56cf-44d6-aa85-3accbfae3fff
Block Pool ID:	BP-1070268464-127.0.1.1-1618323632891

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 108.04 MB of 312 MB Heap Memory. Max Heap Memory is 3.88 GB.

RESULT

Thus, the installation, configuration of Hadoop and R has been executed successfully.

Exp No: 2	Implement word count / frequency programs using MapReduce
Date:	

AIM

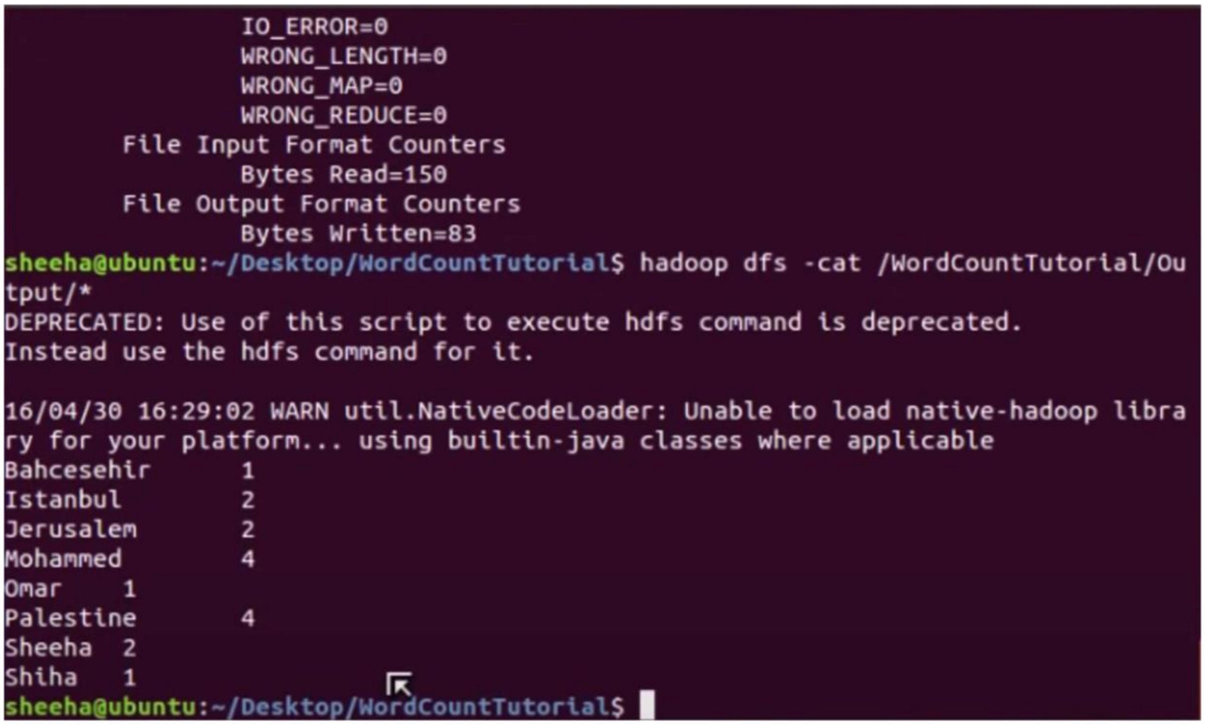
To implement word count program using MapReduce.

PROCEDURE

Run the following command on ubuntu terminal.

1. Create a directory on the Desktop named Lab and inside it create two folders; one called “Input” and the other called “tutorial_classes”.
   a. cd Desktop
   b. mkdir Lab
   c. mkdir Lab/Input
   d. mkdir Lab/tutorial_classes
2. Add the file attached with this document “WordCount.java” in the directory Lab
3. Add the file attached with this document “input.txt” in the directory Lab/Input.
4. Type the following command to export the hadoop classpath into bash.
   a. export HADOOP_CLASSPATH=$(hadoop classpath)
5. Make sure it is now exported.
   a. echo $HADOOP_CLASSPATH
6. It is time to create these directories on HDFS rather than locally. Type the following commands.
   a. hadoop fs -mkdir /WordCountTutorial
   b. hadoop fs -mkdir /WordCountTutorial/Input
   c. hadoop fs -put Lab/Input/input.txt /WordCountTutorial/Input
7. Go to localhost:9870 from the browser, Open “Utilities → Browse File System” and you should see the directories and files we placed in the file system.
8. Then, back to local machine where we will compile the WordCount.java file. Assuming we are currently in the Desktop directory.
   a. cd Lab
   b. javac -classpath $HADOOP_CLASSPATH -d tutorial_classes WordCount.javaPut the output files in one jar file (There is a dot at the end)
   c. jar -cvf WordCount.jar -C tutorial_classes .
9. Now, we run the jar file on Hadoop.
   a. hadoop jar WordCount.jar WordCount /WordCountTutorial/Input /WordCountTutorial/Output
10. Output the result:
    a. hadoop dfs -cat /WordCountTutorial/Output/*

OUTPUT



```
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=150
        File Output Format Counters
                Bytes Written=83
sheeha@ubuntu:~/Desktop/WordCountTutorial$ hadoop dfs -cat /WordCountTutorial/Ou
tput/*
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

16/04/30 16:29:02 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Bahcesehir      1
Istanbul        2
Jerusalem       2
Mohammed        4
Omar    1
Palestine       4
Sheeha  2
Shiha   1
sheeha@ubuntu:~/Desktop/WordCountTutorial$ 
```

RESULT

Thus, the implementation of word count using MapReduce has been executed successfully.

Exp No: 3	**Implement an MR program that processes a Weather Dataset**
Date:	

AIM

To implement an MR program that processes a weather dataset.

PROCEDURE

Run the following commands on ubuntu terminal.

1. Download dataset from - ftp://ftp.ncdc.noaa.gov/pub/data/uscrn/products/daily01
2. Create a java class as MyMaxMin in eclipse IDE

MyMaxMin.java

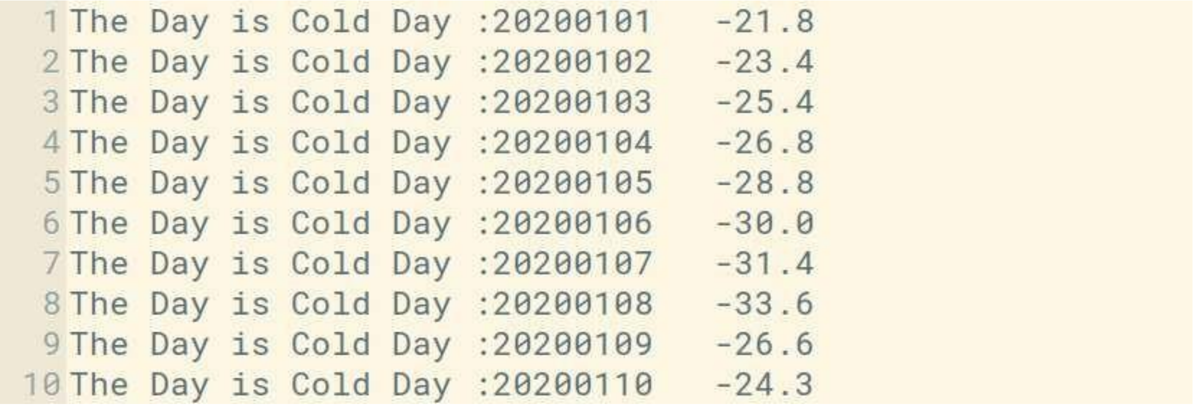
```
import java.io.IOException;
import java.util.Iterator;
import  org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
public class MyMaxMin {
    public static class MaxTemperatureMapper extends
                Mapper<LongWritable, Text, Text, Text> {
    public static final int MISSING = 9999;

    @Override
        public void map(LongWritable arg0, Text Value, Context context)
                    throws IOException, InterruptedException {
        String line = Value.toString();
            if (!(line.length() == 0)) {
                String date = line.substring(6, 14);
                float temp_Max = Float.parseFloat(line.substring(39, 45).trim());
                float temp_Min = Float.parseFloat(line.substring(47, 53).trim()); if
                (temp_Max > 30.0) {
                    context.write(new Text("The Day is Hot Day :" + date),
                                         new
Text(String.valueOf(temp_Max)));
```

```
                }

                if (temp_Min < 15) {
                    context.write(new Text("The Day is Cold Day :" + date),
                            new Text(String.valueOf(temp_Min)));
                }
            }
        }

    }
    public static class MaxTemperatureReducer extends
            Reducer<Text, Text, Text, Text> {
        public void reduce(Text Key, Iterator<Text> Values, Context context)
                throws IOException, InterruptedException {
            String temperature = Values.next().toString();
            context.write(Key, new Text(temperature));
        }

    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = new Job(conf, "weather example");
        job.setJarByClass(MyMaxMin.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);
        job.setMapperClass(MaxTemperatureMapper.class);
        job.setReducerClass(MaxTemperatureReducer.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        Path OutputPath = new Path(args[1]);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        OutputPath.getFileSystem(conf).delete(OutputPath);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

3. Now we add these external jars to our MyProject. Right Click on MyProject -> then select Build Path-> Click on Configure Build Path and select Add External jars…. and add jars from it’s download location then click -> Apply and Close.
4. Now export the project as jar file. Right-click on MyProject choose Export.. and go to Java -> JAR file click -> Next and choose your export destination then click -> Next.
5. Choose Main Class as MyMaxMin by clicking -> Browse and then click -> Finish -> Ok.

6. Start Hadoop
    a. start-all.sh
7. Move dataset to Hadoop HDFS
    a. hdfs dfs -put /file_path /destination
    b. hdfs dfs -put /home/hadoop/Downloads/CRND0103-2020-AK_Fairbanks_11_NE.txt /
    c. hdfs dfs -ls /
8. Now Run your Jar File with below command and produce the output in MyOutput File.
    a. hadoop jar /home/hadoop/Documents/Project.jar /CRND0103-2020-AK_Fairbanks_11_NE.txt /MyOutput
9. Go to browser – localhost:9870

OUTPUT



```
The Day is Cold Day :20200101   -21.8
The Day is Cold Day :20200102   -23.4
The Day is Cold Day :20200103   -25.4
The Day is Cold Day :20200104   -26.8
The Day is Cold Day :20200105   -28.8
The Day is Cold Day :20200106   -30.0
The Day is Cold Day :20200107   -31.4
The Day is Cold Day :20200108   -33.6
The Day is Cold Day :20200109   -26.6
The Day is Cold Day :20200110   -24.3
```

RESULT

Thus, the implementation of MR program that processes a weather dataset has been executed successfully.

Exp No: 4 a	R – Programming Introduction and Basics
Date:	

AIM

To learn and execute R – Programming basics.

PROCEDURE

- WHAT IS R?

R is a programming language developed by Ross Ihaka and Robert Gentleman in 1993. R possesses an extensive catalogue 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, modelling 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 What is R used for?
- Statistical inference
- Data analysis
- Machine learning algorithm

- R DATA TYPES & OPERATOR

Basic data types

- R works with numerous data types, including
- Scalars
- Vectors (numerical, character, logical)
- Matrices
- Data frames
- Lists

Basics types

- 4.5 is a decimal value called numerics.
- 4 is a natural value called integers. Integers are also numerics.
- TRUE or FALSE is a Boolean value called logical.

- The value inside " " or ' ' are text (string). They are called characters.
- We can check the type of a variable with the class function

Data Types	**R Code**	**Output**
# Numeric	x <- 28 class(x)	## [1] "numeric"
# String	y <- "R is Fantastic" class(y)	## [1] "character"

# Boolean	z <- TRUE 	
	class(z)
		## [1] "logical"

- Variables

Variables store values and are an important component in programming, especially for a data scientist. A variable can store a number, an object, a statistical result, vector, dataset, a model prediction basically anything R outputs. We can use that variable later simply by calling the name of the variable. To declare a variable, we need to assign a variable name. The name should not have space. We can use _ to connect to words. To add a value to the variable, use <- or =.

- SYNTAX

SYNTAX	**RCODE**	**OUTPUT**

# First way to declare a variable: use the `<-` 	
name_of_variable <- value
# Second way to declare a variable: use the `=` 	
name_of_variable = value
	# Print variable x x <- 42
	x
		## [1] 42

	y <- 10 y	## [1] 10
	# We call x and y and apply a subtraction x-y	## [1] 32

- Vectors

A vector is a one-dimensional array. We can create a vector with all the basic data type we learnt before. The simplest way to build a vector in R, is to use the c command.

Vectors	**Type – Rcode**	**OUTPUT**
# Numerical	vec_num <- c(1, 10, 49) vec_num	## [1] 1 10 49

# Character	vec_chr <- c("a", "b", "c") vec_chr	## [1] "a" "b" "c"
# Boolean	vec_bool <- c(TRUE, FALSE, TRUE) vec_bool	##[1] TRUE FALSE TRUE

# Create the vectors	vect_1 <- c(1, 3, 5) 	
	vect_2 <- c(2, 4, 6)
	# Take the sum of A_vector and B_vector
	sum_vect <- vect_1 + vect_2 # Print out total_vector sum_vect
		[1] 3 7 11

# Slice the first 5 rows of the vector	slice_vector <- c(1,2,3,4,5,6,7,8,9,10) slice_vector[1:5]	## [1] 1 2 3 4 5
# Faster way to create adjacent values	c(1:10)	## [1] 1 2 3 4 5 6 7 8 9 10

Operator Description	**RCode**	**OUTPUT**
Arithmetic + Addition - Subtraction * Multiplication / Division ^ or ** Exponentiation	 3 + 4 4-2 3*5 (5+5)/2 2^5	 ## [1] 7 ## [1] 2 ## [1] 15 ## [1] 5 ## [1] 32
Logical	logical_vector <- c(1:10) logical_vector>5 logical_vector[(logical_vector>5)]	## [1]FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE ## [1] 6 7 8 9 10
	# Print 5 and 6 logical_vector <- c(1:10) logical_vector[(logical_vector>4) & (logical_vector<7)]	## [1] 5 6

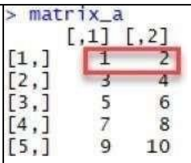
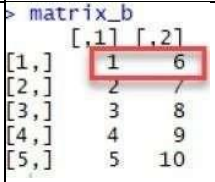
- R MATRIX TUTORIAL: CREATE, PRINT, ADD COLUMN, SLICE

What is a Matrix?

A matrix is a 2-dimensional array that has m number of rows and n number of columns. In other words, matrix is a combination of two or more vectors with the same data type. Note: It is possible to create more than two dimensions arrays with R.

SYNTAX: matrix(data, nrow, ncol, byrow = FALSE) Arguments:

- - data: The collection of elements that R will arrange into the rows and columns of the matrix
- - nrow: Number of rows
- - ncol: Number of columns
- - byrow: The rows are filled from the left to the right. We use `byrow = FALSE` (default values), if we want the matrix to be filled by the columns i.e. the values are filled top to bottom.

# Construct a matrix with 5 rows that contain the numbers 1 up to 10 and byrow = TRUE	matrix_a <-matrix(1:10, byrow = TRUE, nrow = 5) matrix_a	 > matrix_a [,1] [,2] [1,] 1 2 [2,] 3 4 [3,] 5 6 [4,] 7 8 [5,] 9 10
# Print dimension of the matrix	dim(matrix_a)	## [1] 5 2
# Construct a matrix with 5 rows that contain the numbers 1 up to 10 and byrow = FALSE	matrix_b <-matrix(1:10, byrow = FALSE, nrow = 5) matrix_b	 > matrix_b [,1] [,2] [1,] 1 6 [2,] 2 7 [3,] 3 8 [4,] 4 9 [5,] 5 10
	matrix_c <-matrix(1:12, byrow = FALSE, ncol = 3) matrix_c	## [,1] [,2] [,3] ## [1,] 1 5 9 ## [2,] 2 6 10 ## [3,] 3 7 11 ## [4,] 4 8 12
# concatenate c(1:5) to the matrix_a	matrix_a1 <- cbind(matrix_a, c(1:5)) # Check the dimension dim(matrix_a1)	## [1] 5 3

- Slice a Matrix

We can select elements one or many elements from a matrix by using the square brackets [ ]. This is where slicing comes into the picture.

For example:

- matrix_c[1,2] selects the element at the first row and second column.
- matrix_c[1:3,2:3] results in a matrix with the data on the rows 1, 2, 3 and columns 2, 3.
- matrix_c[,1] selects all elements of the first column.
- matrix_c[1,] selects all elements of the first row.

- WHAT IS FACTOR IN R? CATEGORICAL & CONTINUOUS

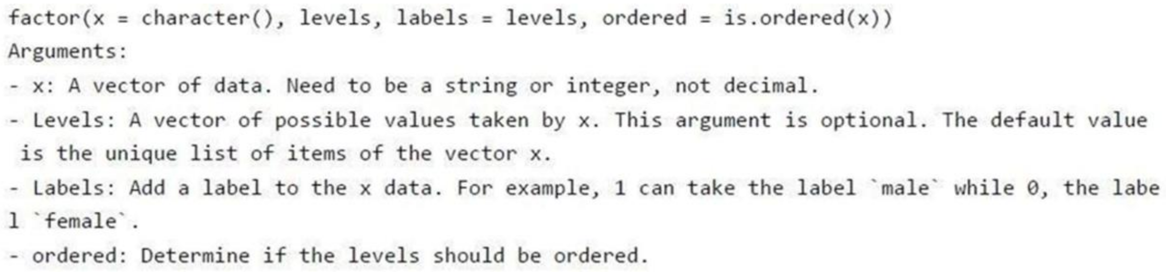
What is Factor in R?

Factors are variables in R which take on a limited number of different values; such variables are often referred to as categorical variables. In a dataset, we can distinguish two types of variables: categorical and continuous. In a categorical variable, the value is limited and usually based on a particular finite group. For example, a categorical variable can be countries, year, gender, occupation. A continuous variable, however, can take any values, from integer to decimal. For example, we can have the revenue, price of a share, etc..

- Categorical variables

R stores categorical variables into a factor. Let's check the code below to convert a character variable into a factor variable. Characters are not supported in machine learning algorithm, and the only way is to convert a string to an integer.

- SYNTAX



```
factor(x = character(), levels, labels = levels, ordered = is.ordered(x))
Arguments:
- x: A vector of data. Need to be a string or integer, not decimal.
- Levels: A vector of possible values taken by x. This argument is optional. The default value
 is the unique list of items of the vector x.
- Labels: Add a label to the x data. For example, 1 can take the label `male` while 0, the labe
l `female`.
- ordered: Determine if the levels should be ordered.
```

- RCODE

# Create gender vector

gender_vector <- c("Male", "Female", "Female", "Male", "Male") class(gender_vector)
# Convert gender_vector to a factor
factor_gender_vector <-factor(gender_vector) class(factor_gender_vector)

- OUTPUT:

## [1] "character"
## [1] "factor"

- Nominal categorical variable

A categorical variable has several values but the order does not matter. For instance, male or female categorical variable do not have ordering.

- RCODE:

```
# Create a color vector

color_vector <- c('blue', 'red', 'green', 'white', 'black', 'yellow') # Convert the vector to factor
factor_color <- factor(color_vector) factor_color
```

- OUTPUT:

```
## [1] blue red green white black yellow
## Levels: black blue green red white yellow
```

- Ordinal categorical variable

Ordinal categorical variables do have a natural ordering. We can specify the order, from the lowest to the highest with order = TRUE and highest to lowest with order = FALSE. We can use summary to count the values for each factor.

- RCODE:

```
# Create Ordinal categorical vector
day_vector <- c('evening', 'morning', 'afternoon', 'midday', 'midnight', 'evening') # Convert
`day_vector` to a factor with ordered level


factor_day <- factor(day_vector, order = TRUE, levels =c('morning', 'midday', 'afternoon',
'evening', 'midnight'))
# Print the new variable factor_day
```

- OUTPUT:

```
## [1]  evening  morning  afternoon  midday midnight evening
## Levels: morning < midday < afternoon < evening < midnight # Append the line to above code
# Count the number of occurence of each level summary(factor_day)
## Morning midday afternoon evening midnight ## 1 1 1 2 1
```

- Continuous variables

Continuous class variables are the default value in R. They are stored as numeric or integer. We can see it from the dataset below. mtcars is a built-in dataset. It gathers information on different types of car. We can import it by using mtcars and check the class of the variable mpg, mile per gallon. It returns a numeric value, indicating a continuous variable.

- RCODE:

```
dataset <- mtcars class(dataset)
```

- OUTPUT:

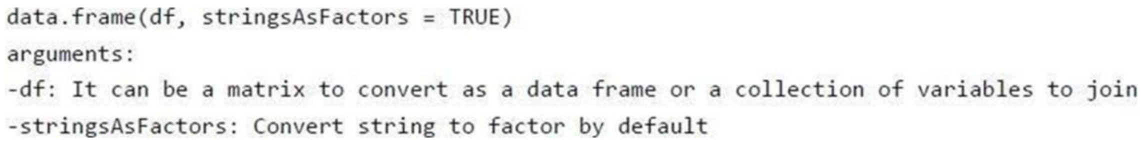
```
## [1] "numeric"
```

- R DATA FRAMES: CREATE, APPEND, SELECT, SUBSET

What is a Data Frame?

A data frame is a list of vectors which are of equal length. A matrix contains only one type of data, while a data frame accepts different data types (numeric, character, factor, etc.).

- Syntax



```
data.frame(df, stringsAsFactors = TRUE)
arguments:
-df: It can be a matrix to convert as a data frame or a collection of variables to join
-stringsAsFactors: Convert string to factor by default
```

- Create a data frame RCODE:

```
# Create a, b, c, d variables a <- c(10,20,30,40)
b  <-   c('book',  'pen',   'textbook',  'pencil_case') c <- c(TRUE,FALSE,TRUE,FALSE)
d <- c(2.5, 8, 10, 7)
# Join the variables to create a data frame df <- data.frame(a,b,c,d)
df
```

- OUTPUT:

```
##    a    b    c d
##    1    1    book TRUE 2.5
##    2    2    pen TRUE 8.0
##    3    3    textbook TRUE 10.0
##    4    4    pencil_case FALSE  7.0
```

```
# Name the data frame
names(df) <-  c('ID',  'items',  'store',  'price') df
```

- OUTPUT:

```
## ID items store price

##    1    10    book TRUE    2.5
##    2    20    pen FALSE    8.0
##    3    30   textbook TRUE 10.0
##       4         40       pencil_case FALSE 7.0
# Print the structure str(df)
```

- OUTPUT:

```
## 'data.frame': 4 obs. of 4 variables:
## $ ID : num 10 20 30 40
## $ items: Factor w/ 4 levels "book","pen","pencil_case",..: 1 2 4 3 ## $ store: logi TRUE
FALSE TRUE FALSE
## $ price: num 2.5 8 10 7
```

- Slice Data Frame

It is possible to SLICE values of a Data Frame. We select the rows and columns to return into bracket precede by the name of the data frame.
RCODE:

```
## Select Rows 1 to 3 and columns 3 to 4
df[1:3, 3:4]
```

- OUTPUT:

```
## store price ## 1 TRUE 2.5
## 2 FALSE 8.0
## 3 TRUE 10.0
```

- Append a Column to Data Frame

You can also append a column to a Data Frame. You need to use the symbol $ to append a new Variable.

- RCODE:

```
# Create a new vector quantity <- c(10, 35, 40, 5)
# Add `quantity` to  the  `df`  data  frame df$quantity <- quantity
df
```

- OUTPUT:

```
## ID items store price quantity ## 1 10 book TRUE 2.5 10
## 2 20 pen FALSE 8.0 35
## 3 30 textbook TRUE 10.0 40
## 4 40 pencil_case FALSE 7.0 5
```

Note: The number of elements in the vector has to be equal to the no of elements in data frame. Executing the following statement

- RCODE

quantity <- c(10, 35, 40)

# Add `quantity` to the `df` data frame df$quantity <- quantity

- Select a column of a data frame

Sometimes, we need to store a column of a data frame for future use or perform operation on a column. We can use the $ sign to select the column from a data frame.

- RCODE:

# Select the column ID df$ID

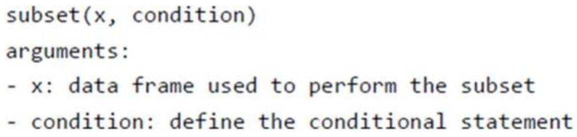
- OUTPUT:

## [1] 1 2 3 4

- Subset a data frame

In the previous section, we selected an entire column without condition. It is possible to subset based on whether or not a certain condition was true.
We use the subset() function.

- SYNTAX



```
subset(x, condition)
arguments:
- x: data frame used to perform the subset
- condition: define the conditional statement
```

We want to return only the items with price above 10, we can do

- RCODE:

# Select price above 5 subset(df, subset = price >

- OUTPUT:

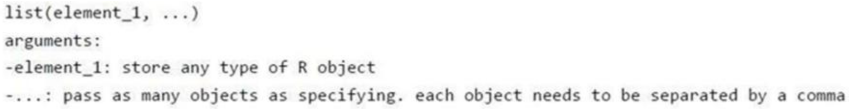
ID items store price
20 pen FALSE 8
30 textbook TRUE 10
40 pencil_case FALSE 7

- LISTS IN R: CREATE, SELECT [EXAMPLE]

- What is a List?

A list is a great tool to store many kinds of object in the order expected. We can include matrices, vectors data frames or lists. We can imagine a list as a bag in which we want to put many different items. When we need to use an item, we open the bag and use it. A list is similar; we can store a collection of objects and use them when we need them.
We can use list() function to create a list.

- SYNTAX

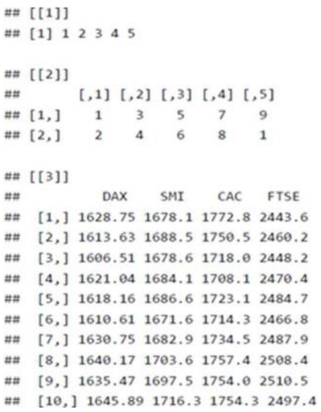


```
list(element_1, ...)
arguments:
-element_1: store any type of R object
-...: pass as many objects as specifying. each object needs to be separated by a comma
```

- RCODE:

```
# Vector with numeric from 1 up to 5 vect <- 1:5
# A 2x 5 matrix
mat <- matrix(1:9, ncol = 5) dim(mat)
# select the 10th row of the built-in R data set EuStockMarkets df <- EuStockMarkets[1:10,]
# Construct list with these vec, mat, and df: my_list <- list(vect, mat, df)
my_list
```

- OUTPUT



```
## [[1]]
## [1] 1 2 3 4 5

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

## [[3]]
##           DAX    SMI    CAC   FTSE
##  [1,] 1628.75 1678.1 1772.8 2443.6
##  [2,] 1613.63 1688.5 1750.5 2460.2
##  [3,] 1606.51 1678.6 1718.0 2448.2
##  [4,] 1621.04 1684.1 1708.1 2470.4
##  [5,] 1618.16 1686.6 1723.1 2484.7
##  [6,] 1610.61 1671.6 1714.3 2466.8
##  [7,] 1630.75 1682.9 1734.5 2487.9
##  [8,] 1640.17 1703.6 1757.4 2508.4
##  [9,] 1635.47 1697.5 1754.0 2510.5
##  [10,] 1645.89 1716.3 1754.3 2497.4
```

- IF, ELSE, ELIF STATEMENT IN R

The if, else, ELIF statement
An if-else statement is a great tool for the developer trying to return an output based on a condition.

- SYNTAX:

```
if (condition1) { expr1
} else it (condition2) { expr2
} else if (condition3) { expr3
} else { expr4
}
```

VAT has different rate according to the product purchased. Imagine we have three different kind of products with different VAT applied:

Categories	Products	VAT
A	Book, magazine, newspaper, etc..	8%
B	Vegetable, meat, beverage, etc..	10%
C	Tee-shirt, jean, pant, etc..	20%

We can write a chain to apply the correct VAT rate to the product a customer bought.

- RCODE:

```
category <- 'A' price <- 10
if (category =='A'){
cat('A vat rate of 8% is applied.','The total price is',price *1.08)
} else if (category =='B'){
cat('A vat rate of 10% is applied.','The total price is',price *1.10)
} else {
cat('A vat rate of 20% is applied.','The total price is',price *1.20)
}
```

- OUTPUT:

```
# A vat rate of 8% is applied. The total price is 10.8
```

- FOR LOOP SYNTAX AND EXAMPLES

```
# Create fruit vector
fruit <-  c('Apple',      'Orange',        'Passion         fruit', 'Banana') # Create the for statement
for ( i in fruit){ print(i)
}
```

- OUTPUT:

```
##      [1]      "Apple"
##      [1]      "Orange"
##      [1]      "Passion fruit"
##      [1]      "Banana"
```

- For Loop over a matrix

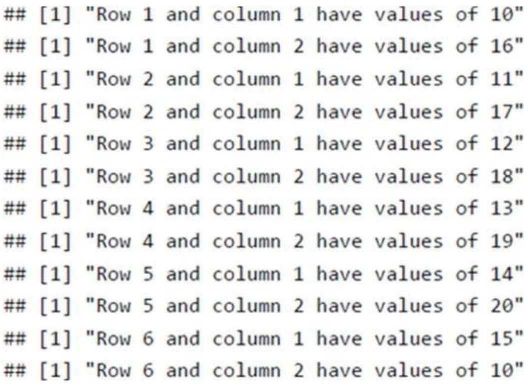
A matrix has 2-dimension, rows and columns. To iterate over a matrix, we have to define two for loop, namely one for the rows and another for the column.

- SYNTAX:

# Create a matrix
mat <- matrix(data = seq(10, 20, by=1), nrow = 6, ncol =2) # Create the loop with r and c to iterate over the matrix

for (r in 1:nrow(mat)) for (c in 1:ncol(mat))
print(paste("Row", r, "and column",c, "have values of", mat[r,c]))

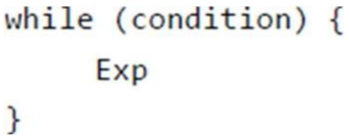
- OUTPUT



```
## [1] "Row 1 and column 1 have values of 10"
## [1] "Row 1 and column 2 have values of 16"
## [1] "Row 2 and column 1 have values of 11"
## [1] "Row 2 and column 2 have values of 17"
## [1] "Row 3 and column 1 have values of 12"
## [1] "Row 3 and column 2 have values of 18"
## [1] "Row 4 and column 1 have values of 13"
## [1] "Row 4 and column 2 have values of 19"
## [1] "Row 5 and column 1 have values of 14"
## [1] "Row 5 and column 2 have values of 20"
## [1] "Row 6 and column 1 have values of 15"
## [1] "Row 6 and column 2 have values of 10"
```

- WHILE LOOP IN R WITH EXAMPLE

A loop is a statement that keeps running until a condition is satisfied. The syntax for a while loop is the following:



```
while (condition) {
     Exp
}
```

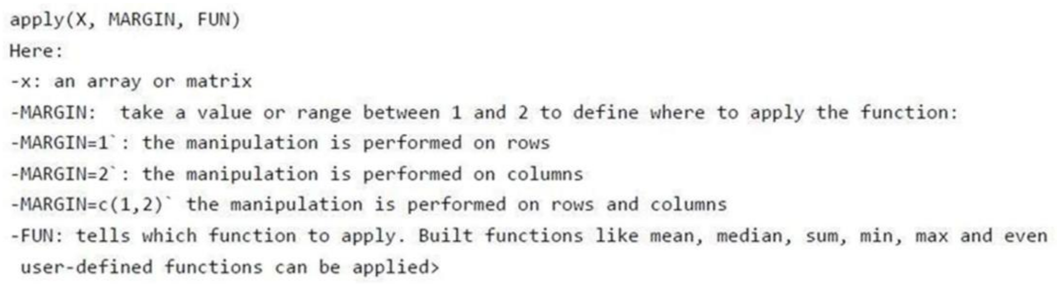
- SYNTAX:

#Create a variable with value 1 begin <- 1
#Create the loop while (begin <= 10){ #See which we are
cat('This is loop number',begin)
#add 1 to the variable begin after each loop begin <- begin+1
print(begin)
}

- APPLY(), SAPPLY(), TAPPLY() IN R WITH EXAMPLES

apply() function
We use apply() over a matrice. This function takes 5 arguments:

- SYNTAX:



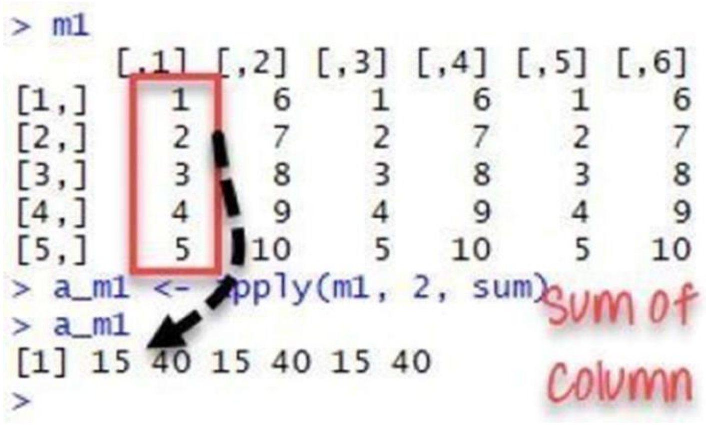
```
apply(X, MARGIN, FUN)
Here:
-x: an array or matrix
-MARGIN:  take a value or range between 1 and 2 to define where to apply the function:
-MARGIN=1`: the manipulation is performed on rows
-MARGIN=2`: the manipulation is performed on columns
-MARGIN=c(1,2)` the manipulation is performed on rows and columns
-FUN: tells which function to apply. Built functions like mean, median, sum, min, max and even
 user-defined functions can be applied>
```

The simplest example is to sum a matrices over all the columns. The code apply(m1, 2, sum) will apply the sum function to the matrix 5x6 and return the sum of each column accessible in the dataset.

- RCODE:

m1 <- matrix(C<-(1:10),nrow=5, ncol=6) m1
a_m1 <- apply(m1, 2, sum) a_m1

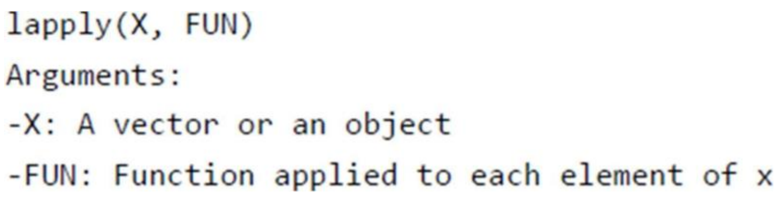
- OUTPUT:



```
> m1
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    6    1    6    1    6
[2,]    2    7    2    7    2    7
[3,]    3    8    3    8    3    8
[4,]    4    9    4    9    4    9
[5,]    5   10    5   10    5   10
> a_m1 <- apply(m1, 2, sum)
> a_m1
[1] 15 40 15 40 15 40
>
```

- lapply() function

- SYNTAX



```
lapply(X, FUN)
Arguments:
-X: A vector or an object
-FUN: Function applied to each element of x
```

l in lapply() stands for list. The difference between lapply() and apply() lies between the output return. The output of lapply() is a list. lapply() can be used for other objects like data frames and lists. lapply() function does not need MARGIN.

A very easy example can be to change the string value of a matrix to lower case with to lower function. We construct a matrix with the name of the famous movies. The name is in upper case format.

- RCODE:

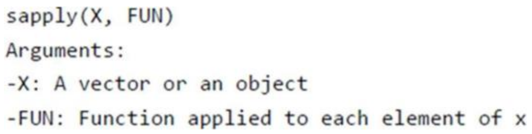
```
movies   <-    c("SPYDERMAN","BATMAN","VERTIGO","CHINATOWN")
movies_lower <-lapply(movies, tolower) str(movies_lower)
We can use unlist() to convert the list into a vector. movies_lower <-
unlist(lapply(movies,tolower)) str(movies_lower)
```

- OUTPUT:

```
## List of 4
## $:chr"spyderman"
##       $:chr"batman"
##       $:chr"vertigo"
## $:chr"chinatown"
## chr [1:4]  "spyderman" "batman"  "vertigo"  "chinatown"
```

- sapply() function

- SYNTAX



```
sapply(X, FUN)
Arguments:
-X: A vector or an object
-FUN: Function applied to each element of x
```

- RCODE:

```
dt <- cars
lmn_cars <- lapply(dt, min) smn_cars <- sapply(dt, min) lmn_cars
smn_cars
lmxcars <- lapply(dt, max) smxcars <- sapply(dt, max) lmxcars
smxcars
```

We can use a user built-in function into lapply() or sapply(). We create a function named avg to compute the average of the minimum and maximum of the Vector.

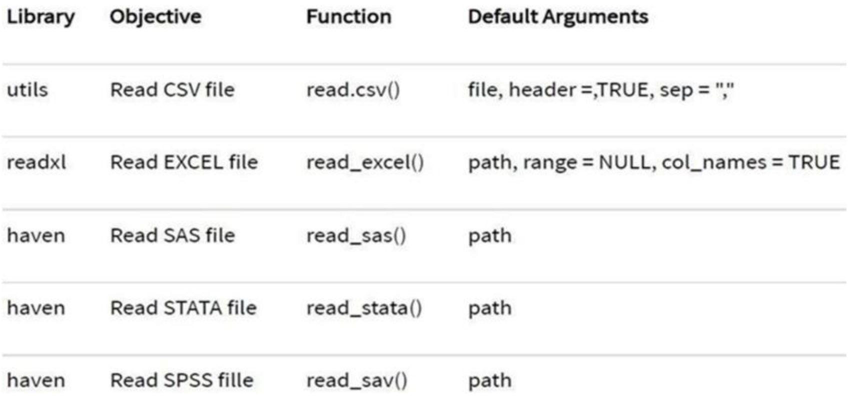
```
avg <- function(x) {
( min(x) + max(x) ) / 2} fcars <- sapply(dt, avg) fcars
```

- OUTPUT:

```
## $speed
## [1] 4
## $dist
## [1] 2
## speed dist
## 4 2
## $speed
## [1] 25
##  $dist
## [1] 120
## speed dist
## 25 120
```

Function	Arguments	Objective	Input	Output
apply	apply(x, MARGIN, FUN)	Apply a function to the rows or columns or both	Data frame or matrix	vector, list, array
lapply	lapply(X, FUN)	Apply a function to all the elements of the input	List, vector or data frame	list
sapply	sappy(X FUN)	Apply a function to all the elements of the input	List, vector or data frame	vector or matrix

- IMPORT DATA INTO R: READ CSV, EXCEL, SPSS, STATA, SAS FILES

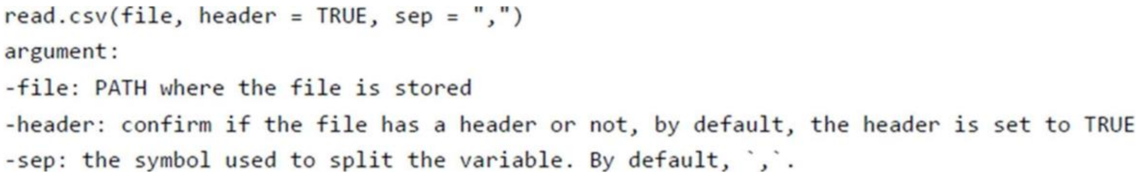


Library	Objective	Function	Default Arguments
utils	Read CSV file	read.csv()	file, header =,TRUE, sep = ","
readxl	Read EXCEL file	read_excel()	path, range = NULL, col_names = TRUE
haven	Read SAS file	read_sas()	path
haven	Read STATA file	read_stata()	path
haven	Read SPSS fille	read_sav()	path

- Read CSV

One of the most widely data store is the .csv (comma-separated values) file formats. R loads an array of libraries during the start-up, including the utils package. This package is convenient to open csv files combined with the reading.csv () function.

- SYNTAX:



```
read.csv(file, header = TRUE, sep = ",")
argument:
-file: PATH where the file is stored
-header: confirm if the file has a header or not, by default, the header is set to TRUE
-sep: the symbol used to split the variable. By default, `,`.
```

- RCODE:

```
PATH <-
'https://raw.githubusercontent.com/vincentarelbundock/Rdatasets/master/csv/datasets/m tc
ars.csv'
df <- read.csv(PATH, header = TRUE, sep = ',', stringsAsFactors =FALSE) length(df)
class(df$X)
```

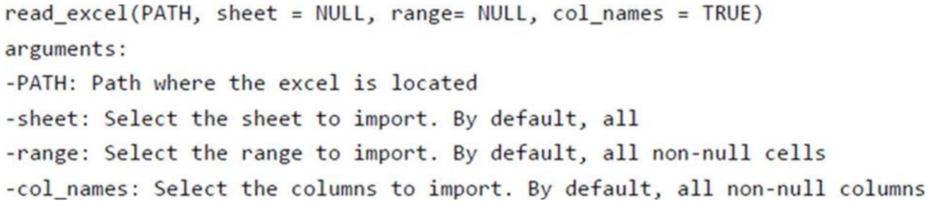
- OUTPUT:

```
## [1] 12
##  [1] "factor"
```

- Read Excel files

Excel files are very popular among data analysts. Spreadsheets are easy to work with and flexible. R is equipped with a library readxl to import Excel spreadsheet.

- SYNTAX



```
read_excel(PATH, sheet = NULL, range= NULL, col_names = TRUE)
arguments:
-PATH: Path where the excel is located
-sheet: Select the sheet to import. By default, all
-range: Select the range to import. By default, all non-null cells
-col_names: Select the columns to import. By default, all non-null columns
```

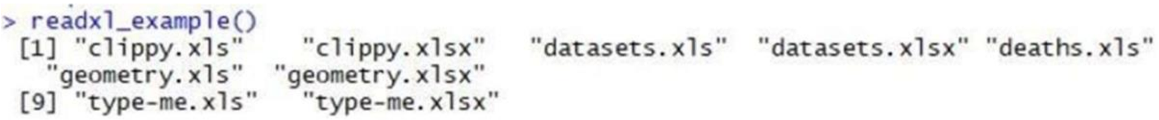
- RCODE:

```
require(readxl)
library(readxl)
readxl_example()
readxl_example("geometry.xls")
```

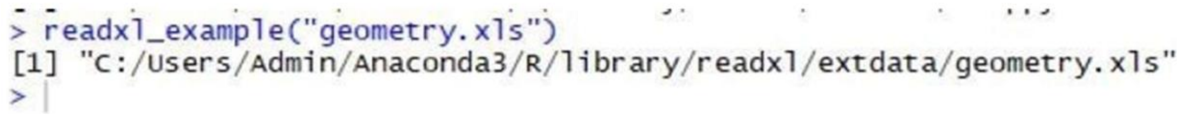
We can import the spreadsheets from the readxl library and count the number of columns in the first sheet. # Store the path of `datasets.xlsx`

example <- readxl_example("datasets.xlsx") #
Import the spreadsheet
df <- read_excel(example)
# Count the number of columns length(df)

- OUTPUT



```
> readxl_example()
 [1] "clippy.xls"    "clippy.xlsx"   "datasets.xls"  "datasets.xlsx" "deaths.xls"
   "geometry.xls"  "geometry.xlsx"
 [9] "type-me.xls"   "type-me.xlsx"
```



```
> readxl_example("geometry.xls")
[1] "C:/Users/Admin/Anaconda3/R/library/readxl/extdata/geometry.xls"
> |
```

- Read excel_sheets()

The file datasets.xlsx is composed of 4 sheets. We can find out which sheets are available in the workbook by using excel_sheets() function

- RCODE:

example <- readxl_example("datasets.xlsx")
excel_sheets(example)

If a worksheet includes many sheets, it is easy to select a particular sheet by using the sheet arguments. We can specify the name of the sheet or the sheet index. We can verify if both function returns the same output with identical().

example <- readxl_example("datasets.xlsx") quake <- read_excel(example, sheet = "quakes") quake_1 <-read_excel(example, sheet = 4) identical(quake, quake_1)

- OUTPUT:

[1] "iris" "mtcars" "chickwts" "quakes"
## [1] TRUE

- R EXPORTING DATA TO CSV, EXCEL, SAS, STATA, AND TEXT FILE

Export to Hard drive

To begin with, you can save the data directly into the working directory. The following code prints the path of your working directory:

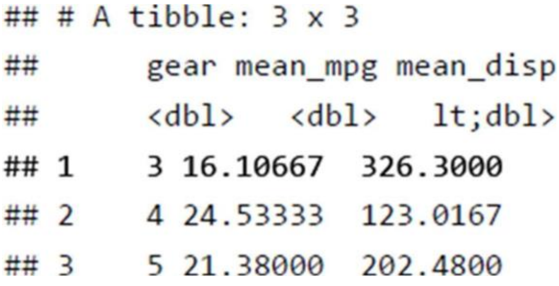
- RCODE:

directory <-getwd() directory OUTPUT:
## [1] "/Users/15_Export_to_do"
Create data frame
First of all, let's import the mtcars dataset and get the mean of mpg and disp grouped by
Gear

- RCODE:

library(dplyr)
df <-mtcars % > % select(mpg, disp, gear) % > % group_by(gear) % > %
summarize(mean_mpg = mean(mpg), mean_disp = mean(disp)) df

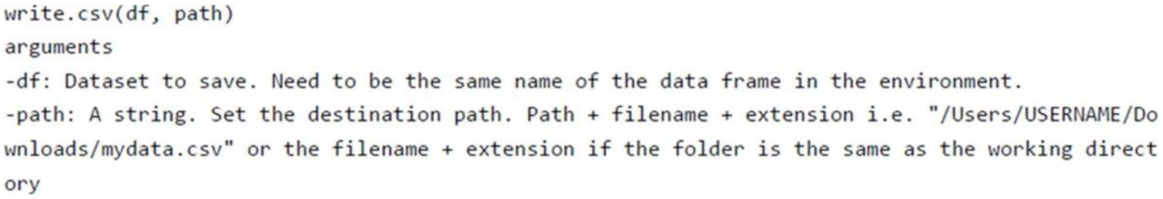
- OUTPUT



```
## # A tibble: 3 x 3
##      gear mean_mpg mean_disp
##      <dbl>   <dbl>   lt;dbl>
## 1    3 16.10667  326.3000
## 2    4 24.53333  123.0167
## 3    5 21.38000  202.4800
```

The table contains three rows and three columns. You can create a CSV file with the function write.csv().
Export CSV

- SYNTAX



```
write.csv(df, path)
arguments
-df: Dataset to save. Need to be the same name of the data frame in the environment.
-path: A string. Set the destination path. Path + filename + extension i.e. "/Users/USERNAME/Do
wnloads/mydata.csv" or the filename + extension if the folder is the same as the working direct
ory
```

- RCODE:

write.csv(df, "table_car.csv")

Code Explanation

write.csv(df, "table_car.csv"): Create a CSV file in the hard drive: df: name of the data frame in the environment
"table_car.csv": Name the file table_car and store it as csv

Note: You can use the function write.csv2() to separate the rows with a semicolon.
write.csv2(df, "table_car.csv")

- Export to Excel file

Export data to Excel is trivial for Windows users and trickier for Mac OS user. Both users will use the library xlsx to create an Excel file. The slight difference comes from the installation of the library. Indeed, the library xlsx uses Java to create the file. Java needs to be installed if not present in your machine.

- Windows users

If you are a Windows user, you can install the library directly with conda:
conda install -c r r-xlsx library(xlsx)
write.xlsx(df, "table_car.xlsx") library(haven)
write_sav(df, "table_car.sav" ## spss file write_sas(df, "table_car.sas7bdat") write_dta(df, "table_car.dta") ## STATA File save(df, file ='table_car.RData')

- R SELECT(), FILTER(), ARRANGE()

Verb	Objective	Code	Explanation
glimpse	check the structure of a df	glimpse(df)	Identical to str()

select()	Select/exclude the variables	select(df, A, B,C)
		select(df, A:C)
		select(df, -C)
			Select the variables A, B and C
			Select all variables from A to C
			Exclude C

arrange()	Sort the dataset with oneor many variables	arrange(A)
		arrange(desc (A), B)
			Descending sort of variable A and ascending sort of B

RESULT

Thus, basics programs of R – programming has been executed successfully.

Exp No: 4 b	**Implement Linear and Logistic Regression**
Date:	

AIM

To implement linear and logistic regression.

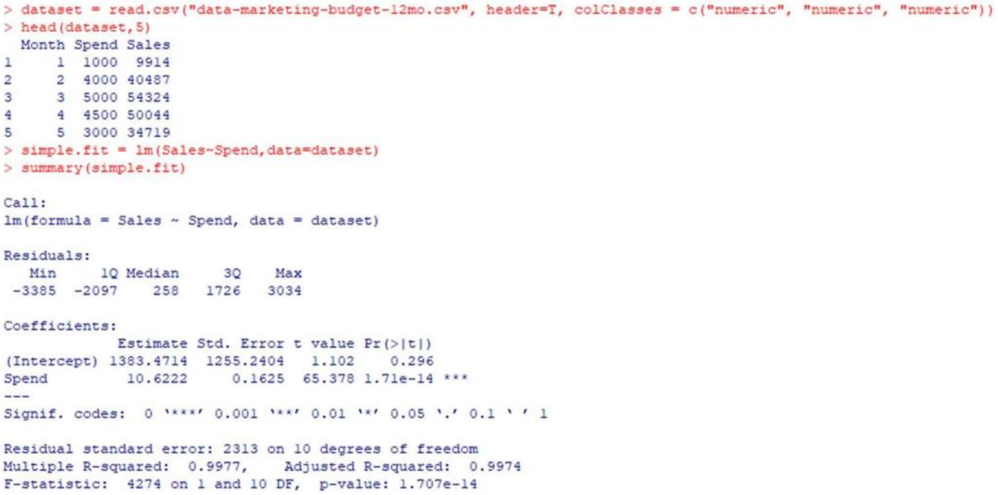
PROCEDURE

1. Open R on windows.
2. Create a new workspace.
3. Create a new script file.
4. Type the code in the script file.
5. Run the script file.
6. Close R.

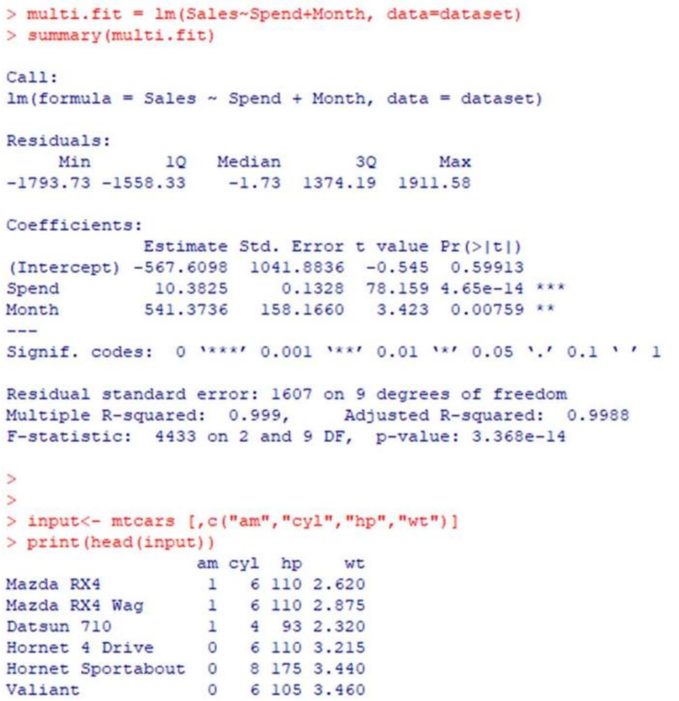
PROGRAM

```
>   dataset = read.csv("data-marketing-budget-12mo.csv", header=T, colClasses = c("numeric",
    "numeric", "numeric"))
>   head(dataset,5)
>   simple.fit = lm(Sales~Spend,data=dataset)
>   summary(simple.fit)
>   multi.fit = lm(Sales~Spend+Month, data=dataset)
>   summary(multi.fit)
>   input<- mtcars [,c("am","cyl","hp","wt")]
>   print(head(input))
>   am.data =glm(formula = am ~ cyl+hp+wt,data = input,family = binomial)
>   print(summary(am.data))
```

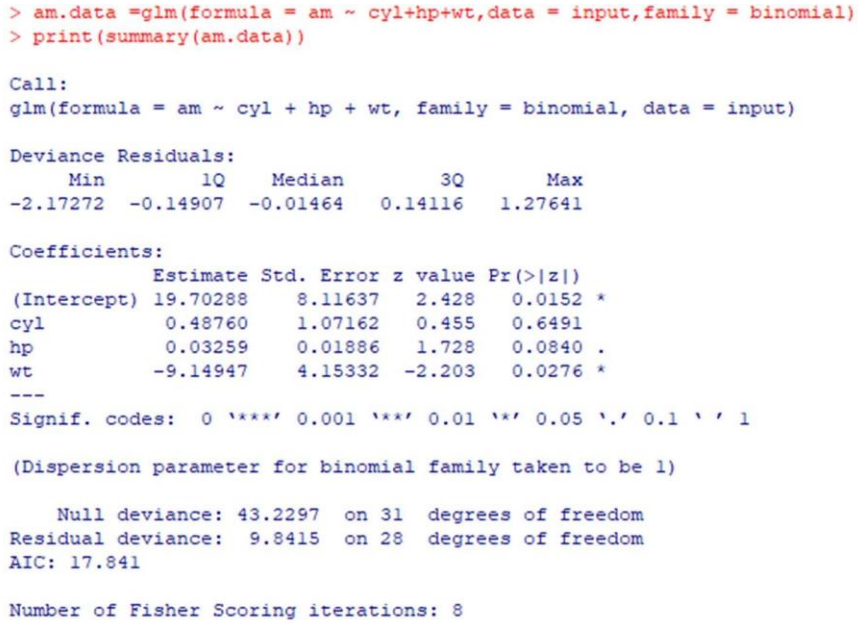
OUTPUT



```
> dataset = read.csv("data-marketing-budget-12mo.csv", header=T, colClasses = c("numeric", "numeric", "numeric"))
> head(dataset,5)
  Month Spend Sales
1     1  1000  9914
2     2  4000 40487
3     3  5000 54324
4     4  4500 50044
5     5  3000 34719
> simple.fit = lm(Sales~Spend,data=dataset)
> summary(simple.fit)

Call:
lm(formula = Sales ~ Spend, data = dataset)

Residuals:
   Min     1Q Median     3Q    Max 
 -3385  -2097    258   1726   3034 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1383.4714  1255.2404   1.102    0.296    
Spend         10.6222     0.1625  65.378 1.71e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2313 on 10 degrees of freedom
Multiple R-squared:  0.9977,	Adjusted R-squared:  0.9974 
F-statistic:  4274 on 1 and 10 DF,  p-value: 1.707e-14
```



```
> multi.fit = lm(Sales~Spend+Month, data=dataset)
> summary(multi.fit)

Call:
lm(formula = Sales ~ Spend + Month, data = dataset)

Residuals:
     Min       1Q   Median       3Q      Max
-1793.73 -1558.33    -1.73  1374.19  1911.58

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -567.6098  1041.8836  -0.545  0.59913
Spend         10.3825     0.1328  78.159 4.65e-14 ***
Month        541.3736   158.1660   3.423  0.00759 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1607 on 9 degrees of freedom
Multiple R-squared:  0.999,     Adjusted R-squared:  0.9988
F-statistic:  4433 on 2 and 9 DF,  p-value: 3.368e-14

>
>
> input<- mtcars [,c("am","cyl","hp","wt")]
> print(head(input))
                  am cyl  hp    wt
Mazda RX4          1   6 110 2.620
Mazda RX4 Wag      1   6 110 2.875
Datsun 710         1   4  93 2.320
Hornet 4 Drive     0   6 110 3.215
Hornet Sportabout  0   8 175 3.440
Valiant            0   6 105 3.460
```



```
> am.data =glm(formula = am ~ cyl+hp+wt,data = input,family = binomial)
> print(summary(am.data))

Call:
glm(formula = am ~ cyl + hp + wt, family = binomial, data = input)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.17272  -0.14907  -0.01464   0.14116   1.27641

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 19.70288    8.11637   2.428   0.0152 *
cyl          0.48760    1.07162   0.455   0.6491
hp           0.03259    0.01886   1.728   0.0840 .
wt          -9.14947    4.15332  -2.203   0.0276 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 43.2297  on 31  degrees of freedom
Residual deviance:  9.8415  on 28  degrees of freedom
AIC: 17.841

Number of Fisher Scoring iterations: 8
```

RESULT

Thus, the implementation of linear and logistic regression has been executed successfully.

Exp No: 5 a	**Implement SVM Classification Techniques**
Date:	

AIM

To implement SVM Classification technique.

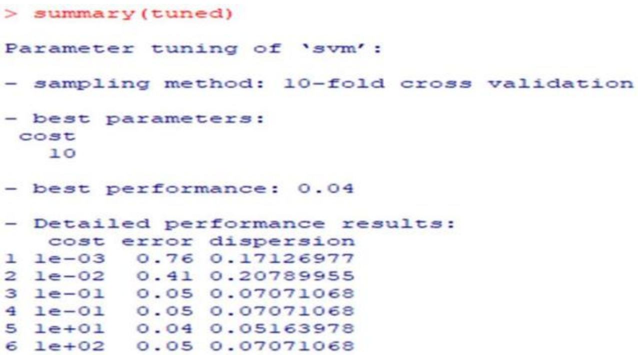
PROCEDURE

1. Open R on windows.
2. Create a new workspace.
3. Create a new script file.
4. Type the code in the script file.
5. Run the script file.
6. Close R.

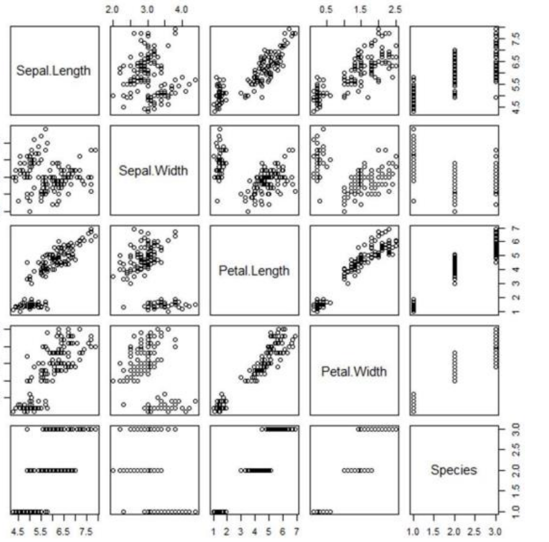
PROGRAM

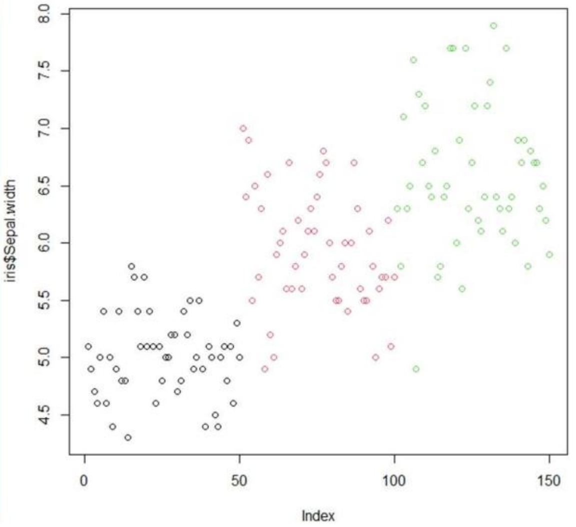
```
>   library(e1071)
>   plot(iris)
>   plot(iris$Sepal.Length, iris$Sepal.width, col=iris$Species)
>   plot(iris$Petal.Length, iris$Petal.width, col=iris$Species)
>   s<-sample(150,100)
>   col<- c("Petal.Length", "Petal.Width", "Species")
>   iris_train<- iris[s,col]
>   iris_test<- iris[-s,col]
>   svmfit<- svm(Species ~., data = iris_train, kernel = "linear", cost = .1, scale = FALSE)
>   print(svmfit)
>   plot(svmfit, iris_train[,col])
>   tuned <- tune(svm, Species~., data = iris_train, kernel = "linear", ranges=
    list(cost=c(0.001,0.01,.1,.1,10,100)))
>   summary(tuned)
>   p<-predict(svmfit, iris_test[,col], type="class")
>   plot(p)
>   table(p,iris_test[,3] )
>   mean(p== iris_test[,3])
```

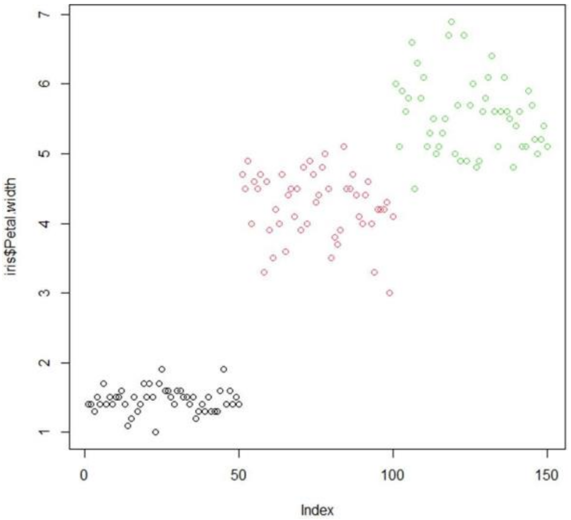
OUTPUT

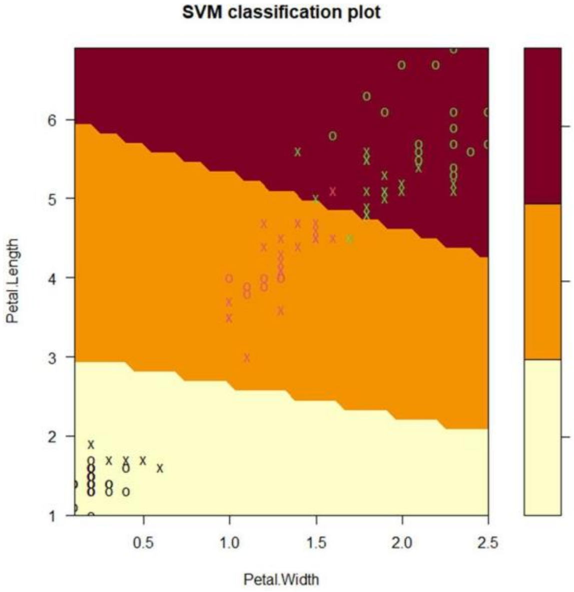


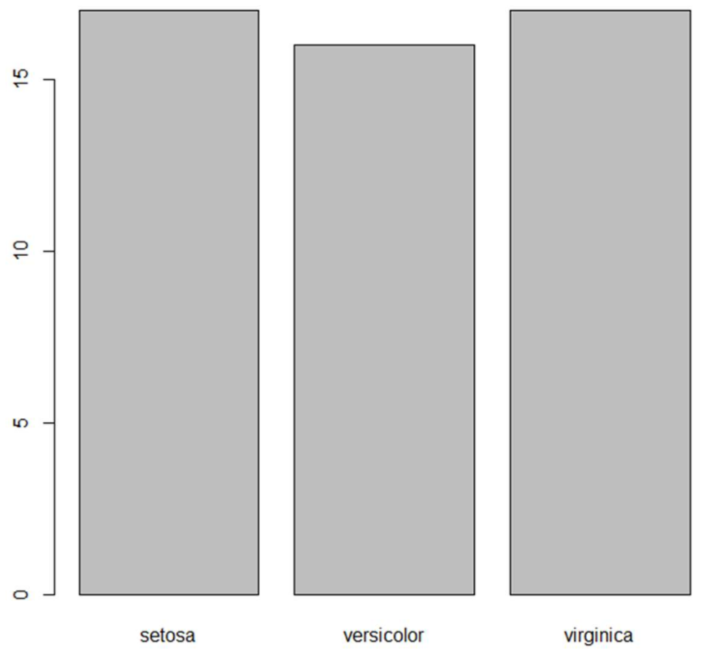
```
> summary(tuned)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
   10

- best performance: 0.04

- Detailed performance results:
   cost error dispersion
1 1e-03  0.76 0.17126977
2 1e-02  0.41 0.20789955
3 1e-01  0.05 0.07071068
4 1e-01  0.05 0.07071068
5 1e+01  0.04 0.05163978
6 1e+02  0.05 0.07071068
```


Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Species
2.0 3.0 4.0
0.5 1.5 2.5
4.5 5.5 6.5 7.5
1 2 3 4 5 6 7
1.0 1.5 2.0 2.5 3.0


iris$Sepal.width
Index
0 50 100 150
4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0


iris$Petal.width
Index
0 50 100 150
1 2 3 4 5 6 7


SVM classification plot
Petal.Length
Petal.Width
0.5 1.0 1.5 2.0 2.5
1 2 3 4 5 6
virginica
versicolor
setosa



RESULT

Thus, the implementation of SVM Classification technique has been executed successfully.

Exp No: 5 b	**Implement Decision Tree Classification Techniques**
Date:	

AIM

To implement decision tree classification technique.

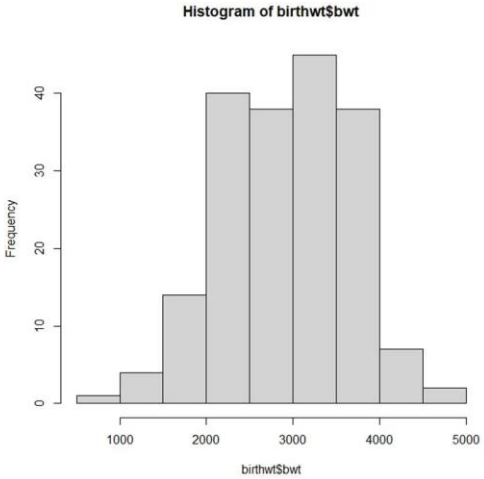
PROCEDURE

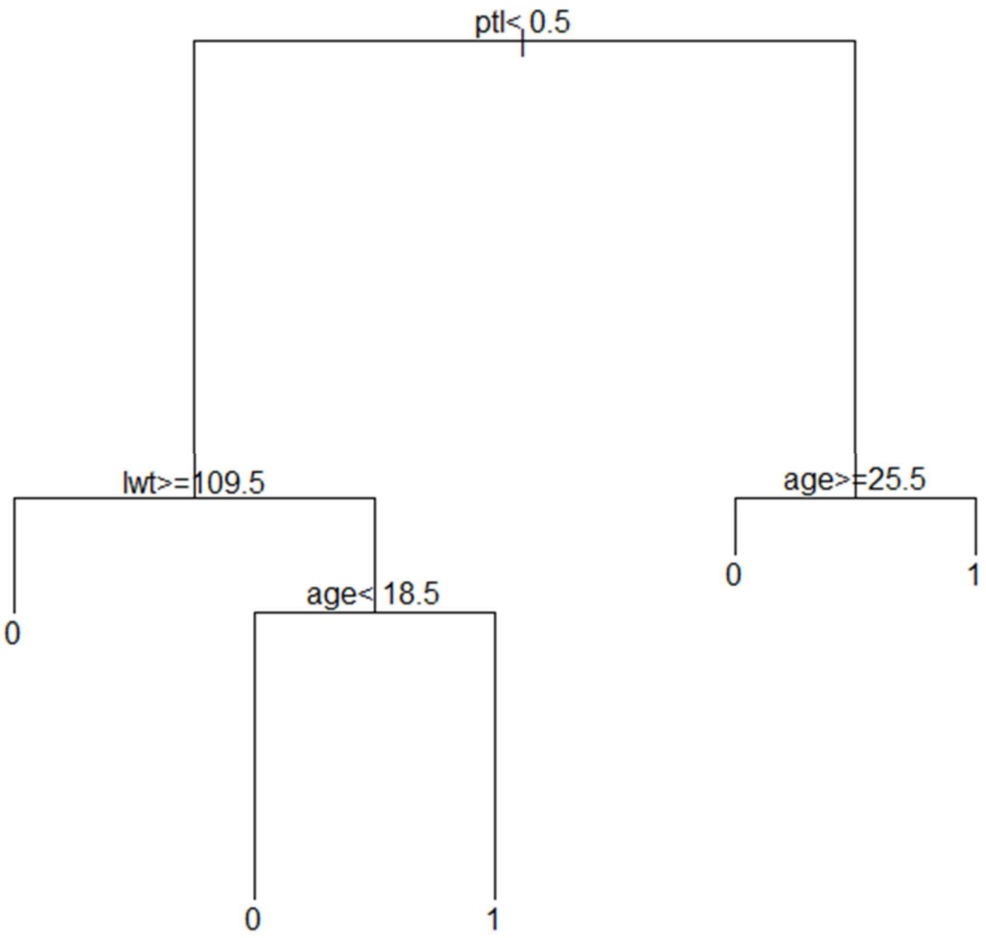
1. Open R on windows.
2. Create a new workspace.
3. Create a new script file.
4. Type the code in the script file.
5. Run the script file.
6. Close R.

PROGRAM

```
>   library(MASS)
>   library(rpart)
>   head(birthwt)
>   hist(birthwt$bwt)
>   table(birthwt$low)
>   cols <- c('low', 'race', 'smoke', 'ht', 'ui')
>   birthwt[cols] <- lapply(birthwt[cols], as.factor)
>   set.seed(1)
>   train<- sample(1:nrow(birthwt), 0.75 * nrow(birthwt))
>   birthwtTree<- rpart(low ~ . - bwt, data = birthwt[train, ], method = 'class')
>   plot(birthwtTree)
>   text(birthwtTree, pretty = 0)
>   summary(birthwtTree)
>   birthwtPred<- predict(birthwtTree, birthwt[-train, ], type = 'class')
>   table(birthwtPred, birthwt[-train, ]$low)
```

OUTPUT





RESULT

Thus, the implementation of decision tree classification technique has been executed successfully.

Exp No: 6	**Implement Clustering Techniques**
Date:	

AIM

To implement clustering techniques.

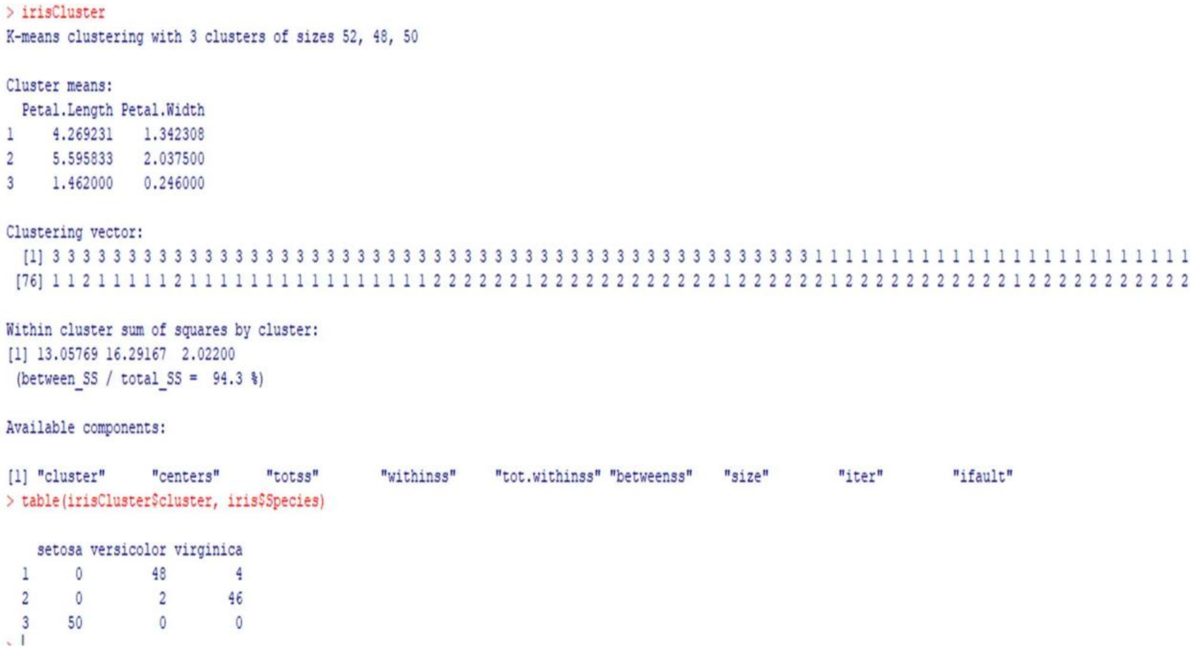
PROCEDURE

1. Open R on windows.
2. Create a new workspace.
3. Create a new script file.
4. Type the code in the script file.
5. Run the script file.
6. Close R.

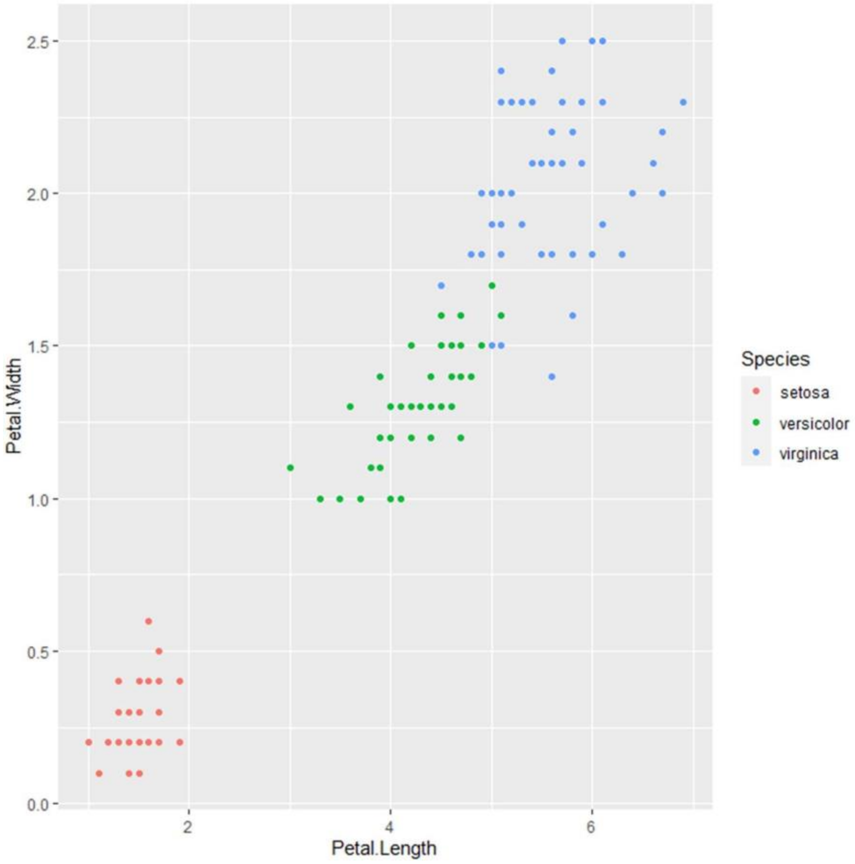
PROGRAM

```
>   library(datasets)
>   head(iris)
>   library(ggplot2)
>   ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()
>   set.seed(20)
>   irisCluster <- kmeans(iris[, 3:4], 3, nstart = 20)
>   irisCluster
>   table(irisCluster$cluster, iris$Species)
```

OUTPUT



```
> irisCluster
K-means clustering with 3 clusters of sizes 52, 48, 50

Cluster means:
  Petal.Length Petal.Width
1     4.269231    1.342308
2     5.595833    2.037500
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 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [76] 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2

Within cluster sum of squares by cluster:
[1] 13.05769 16.29167  2.02200
 (between_SS / total_SS =  94.3 %)

Available components:

[1] "cluster"     "centers"     "totss"       "withinss"    "tot.withinss" "betweenss"    "size"        "iter"        "ifault"
> table(irisCluster$cluster, iris$Species)

    setosa versicolor virginica
  1      0         48         4
  2      0          2        46
  3     50          0         0
```



RESULT

Thus, the implementation of clustering techniques has been executed successfully.

Exp No: 7	**Visualize data using any plotting framework**
Date:	

AIM

To visualize data using any plotting framework in R.

PROCEDURE

1. Open R on windows.
2. Create a new workspace.
3. Create a new script file.
4. Type the code in the script file.
5. Run the script file.
6. Close R.

PROGRAM

1. Histogram

```
> library(RColorBrewer)
> data(VADeaths)
> par(mfrow=c(2,3))
> hist(VADeaths,breaks=10, col=brewer.pal(3,"Set3"),main="Set3 3 colors")
> hist(VADeaths,breaks=3 ,col=brewer.pal(3,"Set2"),main="Set2 3 colors")
> hist(VADeaths,breaks=7, col=brewer.pal(3,"Set1"),main="Set1 3 colors")
> hist(VADeaths,,breaks= 2, col=brewer.pal(8,"Set3"),main="Set3 8 colors")
> hist(VADeaths,col=brewer.pal(8,"Greys"),main="Greys 8 colors")
> hist(VADeaths,col=brewer.pal(8,"Greens"),main="Greens 8 colors")\
```

2. Line Chart

```
> data(AirPassengers)
> plot(AirPassengers,type="l")
```

3. Bar Chart

```
> data("iris")
> barplot(iris$Petal.Length)
> barplot(iris$Sepal.Length,col = brewer.pal(3,"Set1"))
> barplot(table(iris$Species,iris$Sepal.Length),col = brewer.pal(3,"Set1"))
```

4. Box Plot

```
> data(iris)
> par(mfrow=c(2,2))
> boxplot(iris$Sepal.Length,col="red")
> boxplot(iris$Sepal.Length~iris$Species,col="red")
> boxplot(iris$Sepal.Length~iris$Species,col=heat.colors(3))
> boxplot(iris$Sepal.Length~iris$Species,col=topo.colors(3))
> boxplot(iris$Petal.Length~iris$Species)
```

5. Scatter Plot

```
> plot(x=iris$Petal.Length)
> plot(x=iris$Petal.Length,y=iris$Species)
```

6. Heat Map

```
>  x <- rnorm(10,mean=rep(1:5,each=2),sd=0.7)
>  y <- rnorm(10,mean=rep(c(1,9),each=5),sd=0.1)
>  dataFrame<- data.frame(x=x,y=y)
>  set.seed(143)
>  dataMatrix<-as.matrix(dataFrame)[sample(1:10),]
>  heatmap(dataMatrix)
```

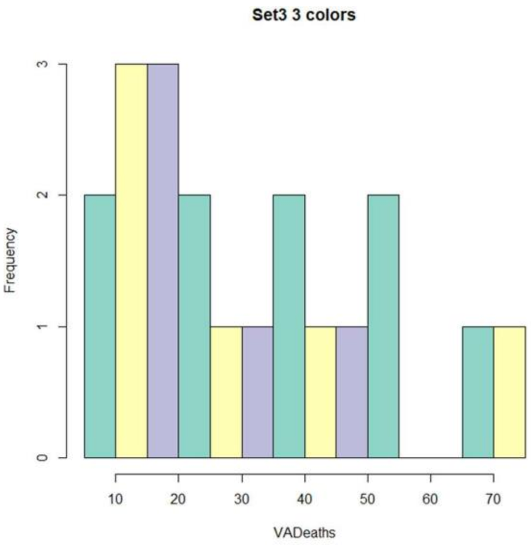
7. Correlogram

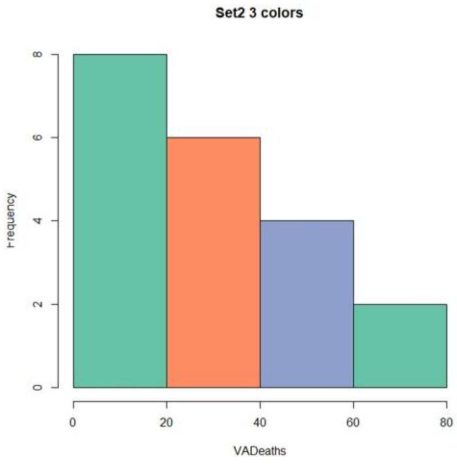
```
>  library("corrplot")
>  data("mtcars")
>  corr_matrix <- cor(mtcars)
>  corrplot(corr_matrix)
>  corrplot(corr_matrix,method = 'number',type = "lower")
```

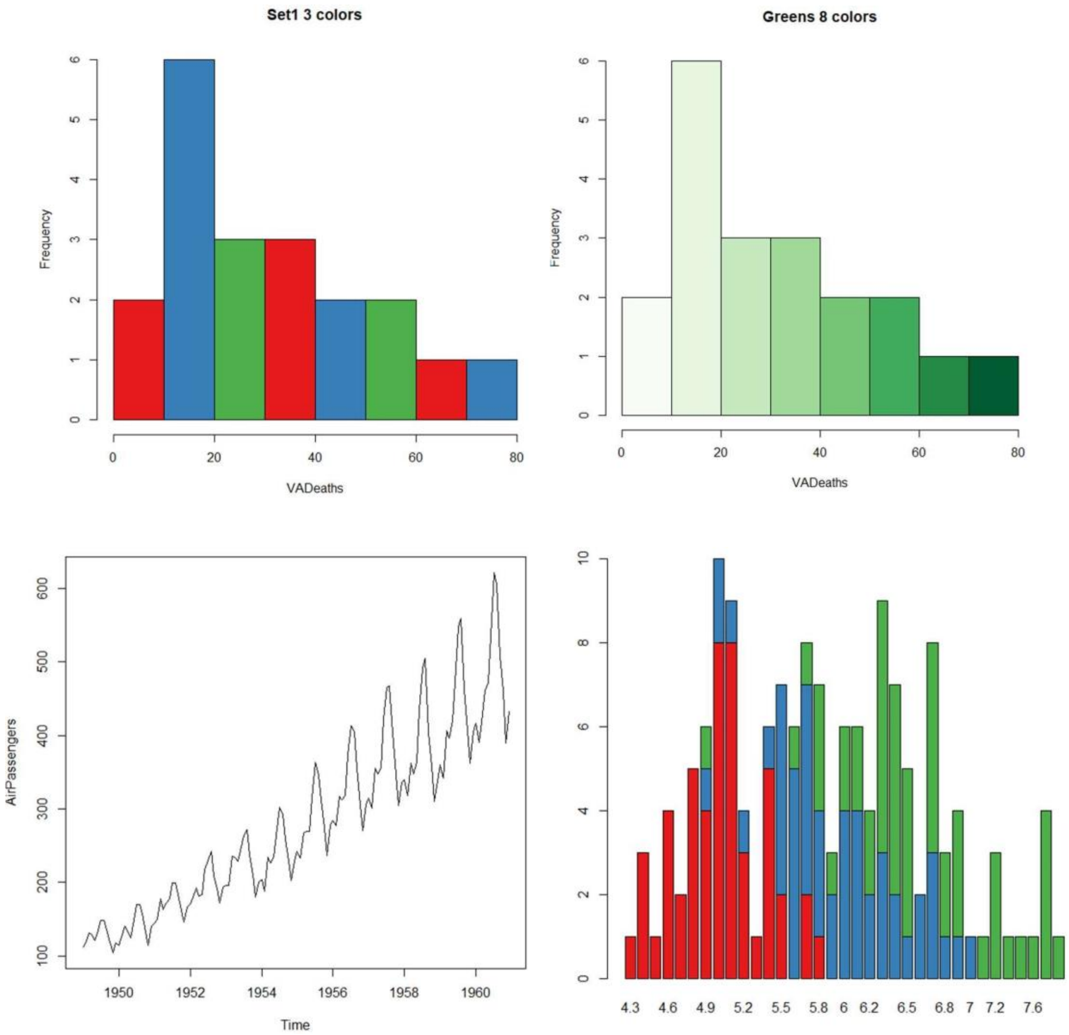
8. Area Chart

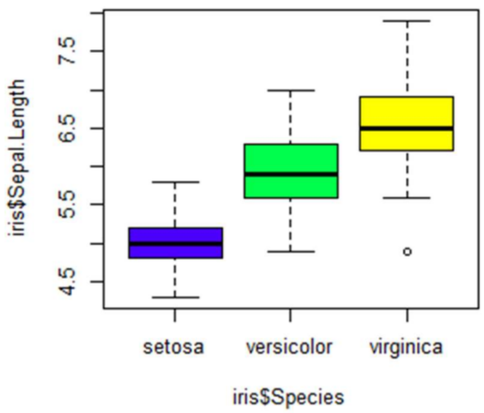
```
>  library(dplyr)
>  library(ggplot2)
>  airquality %>%
       ○  group_by(Day) %>%
       ○  summarise(mean_wind = mean(Wind)) %>%
       ○  ggplot() +
       ○  geom_area(aes(x = Day, y = mean_wind)) +
       ○  labs(title = "Area Chart of Average Wind per Day",
              ▪  subtitle = "using airquality data",
              ▪  y = "Mean Wind")
```

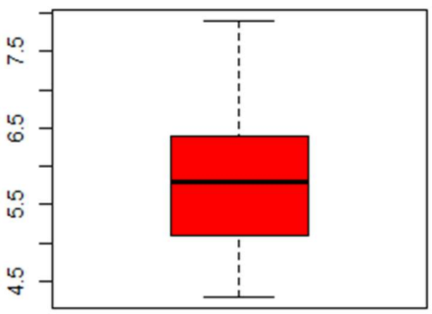
OUTPUT

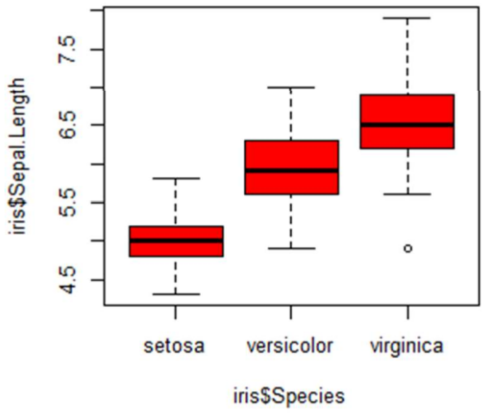


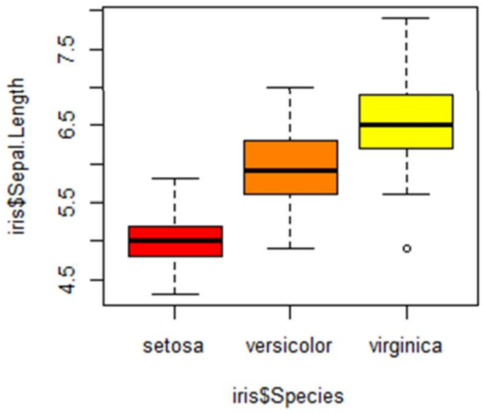


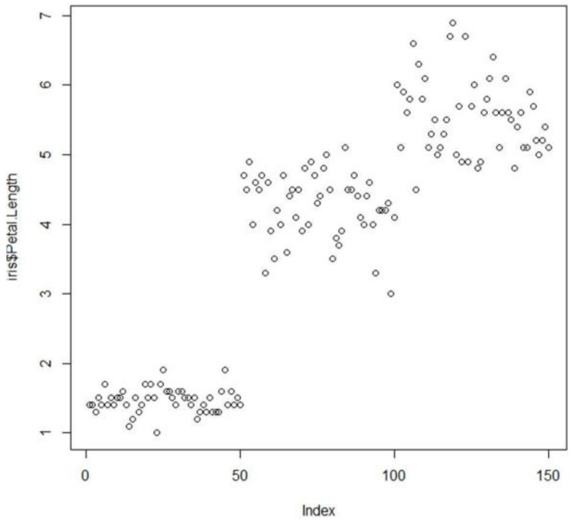

Set1 3 colors
Frequency
VADeaths
0
20
40
60
80
0
1
2
3
4
5
6
Greens 8 colors
Frequency
VADeaths
AirPassengers
Time
1950
1952
1954
1956
1958
1960
100
200
300
400
500
600
0
2
4
6
8
10
4.3
4.6
4.9
5.2
5.5
5.8
6
6.2
6.5
6.8
7
7.2
7.6

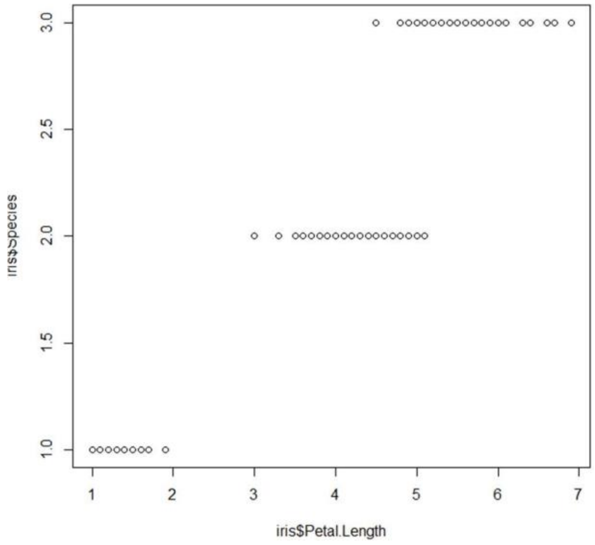

iris$Sepal.Length
4.5
5.5
6.5
7.5
setosa
versicolor
virginica
iris$Species

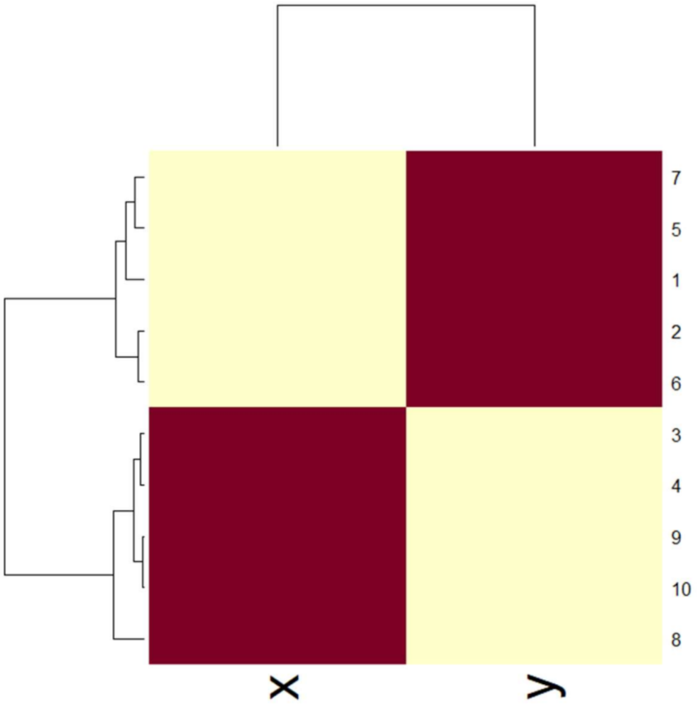

4.5
5.5
6.5
7.5

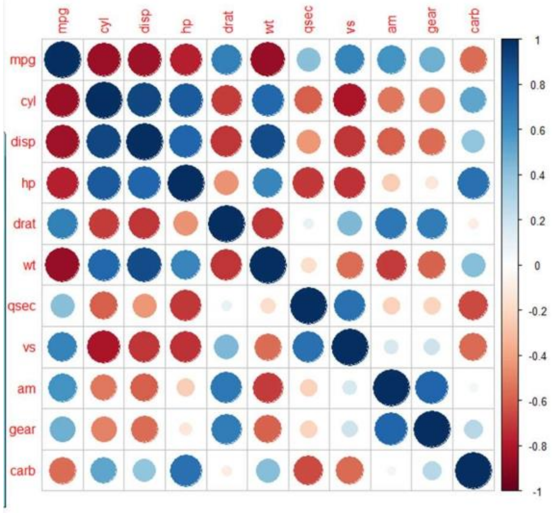

iris$Sepal.Length
4.5
5.5
6.5
7.5
setosa
versicolor
virginica
iris$Species

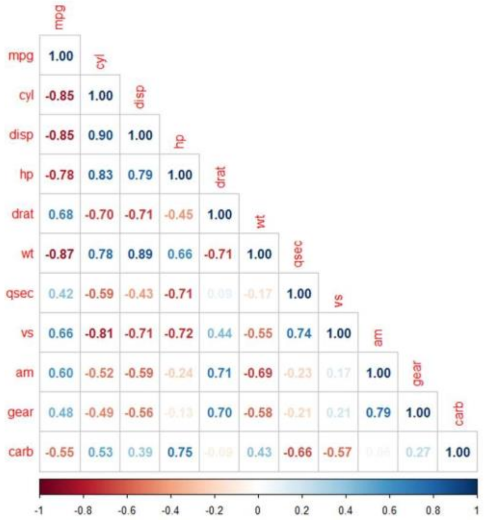

iris$Sepal.Length
4.5
5.5
6.5
7.5
setosa
versicolor
virginica
iris$Species


iris$Petal.Length
1
2
3
4
5
6
7
0
50
100
150
Index

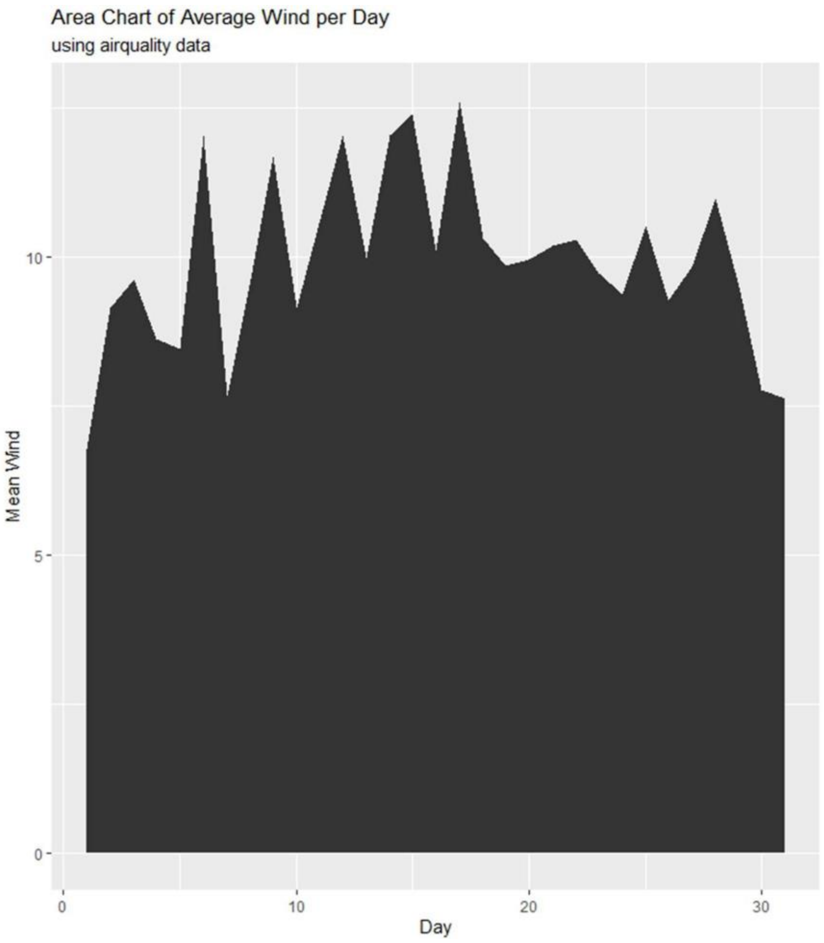

iris$Species
1.0
1.5
2.0
2.5
3.0
1
2
3
4
5
6
7
iris$Petal.Length







	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
mpg	1.00										
cyl	-0.85	1.00									
disp	-0.85	0.90	1.00								
hp	-0.78	0.83	0.79	1.00							
drat	0.68	-0.70	-0.71	-0.45	1.00						
wt	-0.87	0.78	0.89	0.66	-0.71	1.00					
qsec	0.42	-0.59	-0.43	-0.71	0.09	-0.17	1.00				
vs	0.66	-0.81	-0.71	-0.72	0.44	-0.55	0.74	1.00			
am	0.60	-0.52	-0.59	-0.24	0.71	-0.69	-0.23	0.17	1.00		
gear	0.48	-0.49	-0.56	-0.13	0.70	-0.58	-0.21	0.21	0.79	1.00	
carb	-0.55	0.53	0.39	0.75	-0.09	0.43	-0.66	-0.57	0.06	0.27	1.00



RESULT

Thus, the visualization of data using plotting framework has been executed successfully.

Exp No: 8	Implement an application that stores big data in Hbase / MongoDB / Pig using Hadoop / R
Date:	

AIM

To implement an application that stores big data in mongoDB using R.

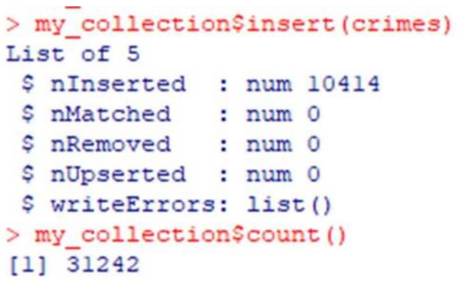
PROCEDURE

1. Open R on windows.
2. Create a new workspace.
3. Create a new script file.
4. Type the code in the script file.
5. Run the script file.
6. Close R.

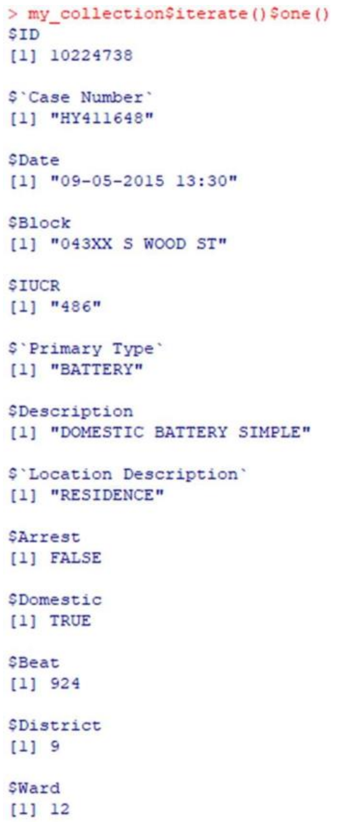
PROGRAM

```
  >  library(ggplot2)
  >  library(mongolite)
  >  library(dplyr)
  >  crimes=data.table::fread("crimes.csv")
  >  connection_string="mongodb://localhost:27017/?tls=false&readPreference=primary"
  >  my_collection = mongo(collection = "crimes", db = "chicago",url=connection_string)
  >  my_collection$insert(crimes)
  >  my_collection$count()
  >  my_collection$iterate()$one()
  >  df <- as.data.frame(my_collection$find())
  >  head(df)
  >  length(my_collection$distinct("Primary Type"))
  >  my_collection$aggregate('[{"$group":{"_id":"$Location Description", "Count":
     {"$sum":1}}}]')%>%na.omit()%>%
>  arrange(desc(Count))%>%head(10)%>%
  >  ggplot(aes(x=reorder(`_id`,Count),y=Count))+
  >  geom_bar(stat="identity",color='skyblue',fill='#b35900')+geom_text(aes(label = Count),
     color = "blue") +coord_flip()+xlab("Location Description")
  >  crimes=my_collection$find('{}', fields = '{"_id":0, "Primary Type":1,"Year":1}')
  >  crimes%>%group_by("Primary
     Type")%>%summarize(Count=n())%>%arrange(desc(Count))%>%head(4)
```

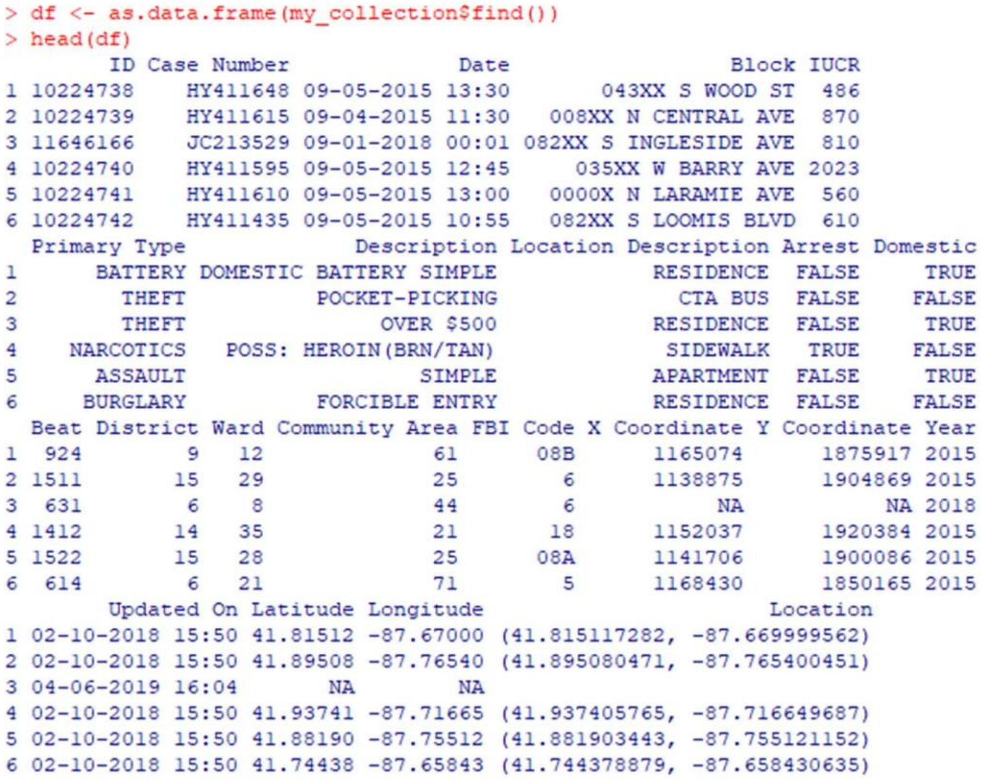
OUTPUT



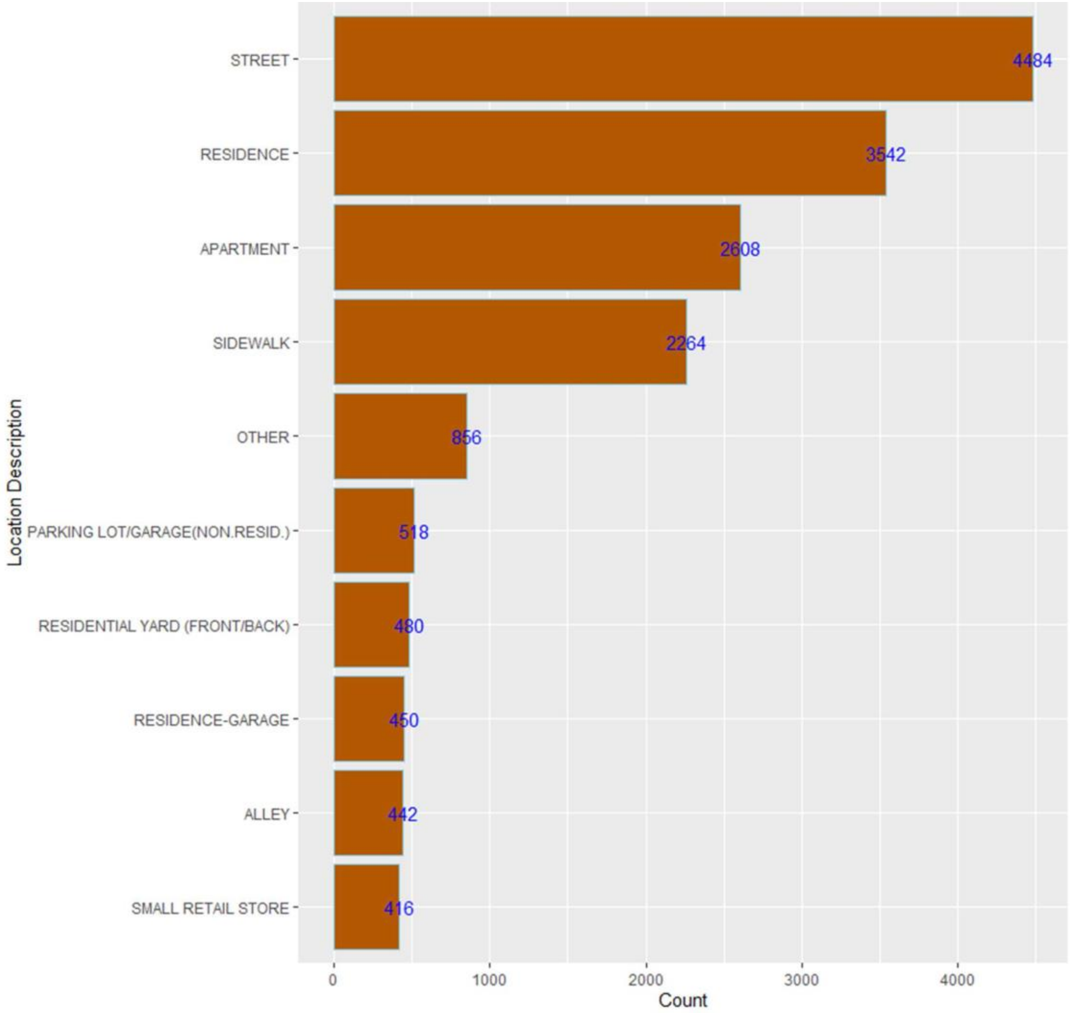
```
> my_collection$insert(crimes)
List of 5
 $ nInserted  : num 10414
 $ nMatched   : num 0
 $ nRemoved   : num 0
 $ nUpserted  : num 0
 $ writeErrors: list()
> my_collection$count()
[1] 31242
```



```
> my_collection$iterate()$one()
$ID
[1] 10224738

$`Case Number`
[1] "HY411648"

$Date
[1] "09-05-2015 13:30"

$Block
[1] "043XX S WOOD ST"

$IUCR
[1] "486"

$`Primary Type`
[1] "BATTERY"

$Description
[1] "DOMESTIC BATTERY SIMPLE"

$`Location Description`
[1] "RESIDENCE"

$Arrest
[1] FALSE

$Domestic
[1] TRUE

$Beat
[1] 924

$District
[1] 9

$Ward
[1] 12
```



```
> df <- as.data.frame(my_collection$find())
> head(df)
        ID Case Number             Date                 Block IUCR
1 10224738    HY411648 09-05-2015 13:30       043XX S WOOD ST  486
2 10224739    HY411615 09-04-2015 11:30   008XX N CENTRAL AVE  870
3 11646166    JC213529 09-01-2018 00:01 082XX S INGLESIDE AVE  810
4 10224740    HY411595 09-05-2015 12:45     035XX W BARRY AVE 2023
5 10224741    HY411610 09-05-2015 13:00   0000X N LARAMIE AVE  560
6 10224742    HY411435 09-05-2015 10:55   082XX S LOOMIS BLVD  610
  Primary Type             Description Location Description Arrest Domestic
1      BATTERY DOMESTIC BATTERY SIMPLE            RESIDENCE  FALSE     TRUE
2        THEFT          POCKET-PICKING              CTA BUS  FALSE    FALSE
3        THEFT               OVER $500            RESIDENCE  FALSE     TRUE
4    NARCOTICS   POSS: HEROIN(BRN/TAN)             SIDEWALK   TRUE    FALSE
5      ASSAULT                  SIMPLE            APARTMENT  FALSE     TRUE
6     BURGLARY          FORCIBLE ENTRY            RESIDENCE  FALSE    FALSE
  Beat District Ward Community Area FBI Code X Coordinate Y Coordinate Year
1  924        9   12             61      08B      1165074      1875917 2015
2 1511       15   29             25        6      1138875      1904869 2015
3  631        6    8             44        6           NA           NA 2018
4 1412       14   35             21       18      1152037      1920384 2015
5 1522       15   28             25      08A      1141706      1900086 2015
6  614        6   21             71        5      1168430      1850165 2015
        Updated On Latitude Longitude                      Location
1 02-10-2018 15:50 41.81512 -87.67000 (41.815117282, -87.669999562)
2 02-10-2018 15:50 41.89508 -87.76540 (41.895080471, -87.765400451)
3 04-06-2019 16:04       NA        NA
4 02-10-2018 15:50 41.93741 -87.71665 (41.937405765, -87.716649687)
5 02-10-2018 15:50 41.88190 -87.75512 (41.881903443, -87.755121152)
6 02-10-2018 15:50 41.74438 -87.65843 (41.744378879, -87.658430635)
```



RESULT

Thus, the implementation of application to store big data in mongoDB using R has been executed successfully.