MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS INSTITUTION – UGC, GOVT. OF INDIA)

MRCET

B.TECH(R-22)
(II YEAR – I SEM)
(2023-24)

R Programming Lab
(R22A6781)

LECTURE LAB MANUAL

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12(B) of UGC ACT 1956

(Affiliated to JNTUH, Hyderabad, Approved by AICTE-Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad–500100, Telangana State, India

Department of Computer Science and Engineering

EMERGING TECHNOLOGIES

R Programming Lab

(R22A6781)

LAB MANUAL

Prepared by

P.Srinivas, Associate Professor & G.Gayatri, Assistant Professor

On

04.8.2022

Department of Computer Science and Engineering

EMERGING TECHNOLOGIES

Vision

- "To be at the forefront of Emerging Technologies and to evolve as a Centre of Excellence in Research, Learning and Consultancy to foster the students into globally competent professionals useful to the Society."

Mission

The department of CSE (Emerging Technologies) is committed to:

- To offer highest Professional and Academic Standards in terms of Personal growth and satisfaction.
- Make the society as the hub of emerging technologies and thereby capture opportunities in new age technologies.
- To create a benchmark in the areas of Research, Education and Public Outreach.
- To provide students a platform where independent learning and scientific study are encouraged with emphasis on latest engineering techniques.

QUALITY POLICY

- To pursue continual improvement of teaching learning process of Undergraduate and Post Graduate programs in Engineering & Management vigorously.
- To provide state of art infrastructure and expertise to impart the quality education and research environment to students for a complete learning experiences.
- Developing students with a disciplined and integrated personality.
- To offer quality relevant and cost effectiveprogrammes to produce engineers as per requirements of the industry need.

For more information: www.mrcet.ac.in

II Year B.Tech CSE(DS)-I Sem(R22) *L/T/P/C*

-/-/2/1

(R22A6781)-RPROGRAMMINGLAB

Course Outcomes : At the end of the Course, the Student will be able to CO1: Show the installation of R Programming Environment.
CO2:Utilize and R Data types for developing programs.
CO3:Make use of different R Data Structures.
CO4:Develop programming logic using R Packages.
CO5:Analyze the data sets using R programming capabilities.

LIST OF PROGRAMS:

1. Download and install R-Programming environment and install basic packages using install. Packages () command in R.
2. Learn al the basics of R-Programming (Data types ,Variables , Operators etc.)
3. Implement R-Loops with different examples.
4. Learn the basics of functions in R and implement with examples.
5. Implement data frames in R. Write a program to join columns and rows in a data frame using c bind () and r bind () in R.
6. Implement different String Manipulation functions in R.
7. Implement different data structures in R(Vectors ,Lists ,Data Frames)
8. Write a program to read acsv file and analyze the data in the file in R
9. Create pie charts and bar charts using R.
10. Create a data set and do statistical analysis on the data using R.
11. Write R program to find Correlation and Covariance
12. Write R program for Regression Modeling
13. Write R program to build classification model using KNN algorithm
14. Write R program to build clustering model using K-mean algorithm

REFERENCES:

1. JaredP.Lander,*RforEveryone:AdvancedAnalyticsandGraphics*,2nd Edition,PearsonEducation,2018.
2. S.R.ManiSekharandT.V.SureshKumar,*ProgrammingwithR*,1st Edition,„CENGAGE,2017.

WEBREFERENCE:

1. https://www.r-project.org/

https://www.tutorialspoint.com/r/index.htm

INDEX

Brief Introduction of R Programming Language :

R is an open-source programming language that is widely used as a statistical software and data analysis tool. R generally comes with the Command-line interface. R is available across widely used platforms like Windows, Linux, and macOS. Also, the R programming language is the latest cutting-edge tool.

It was designed by **Ross Ihaka and Robert Gentleman** at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team. R programming language is an implementation of the S programming language. It also combines with lexical scoping semantics inspired by Scheme. Moreover, the project conceives in 1992, with an initial version released in 1995 and a stable beta version in 2000.

Use of R Programming :

- It's a platform-independent language. This means it can be applied to all operating system.
- It's an open-source free language. That means anyone can install it in any organization without purchasing a license.
- R programming is used as a leading tool for machine learning, statistics, and data analysis. Objects, functions, and packages can easily be created by R.
- R programming language is not only a statistic package but also allows us to integrate with other languages (C, C++). Thus, can easily interact with many data sources and statistical packages.
- The R programming language has a vast community of users and it's growing day by day.
- R is currently one of the most requested programming languages in the Data Science job market that makes it the hottest trend nowadays

1. Installation of R-Studio on windows:

Step – 1: With R-base installed, let's move on to installing RStudio. To begin, goto download RStudioand click on the download button for RStudio desktop.

Step–2: Click on the link for the windows version of RStudio and save the.exe file.

Step–3: Run the .exe and follow the installation instructions.

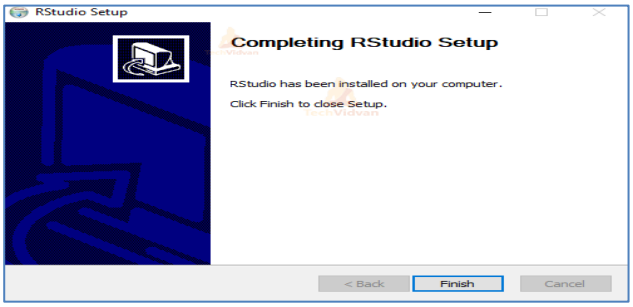
Click Next on the welcome window.

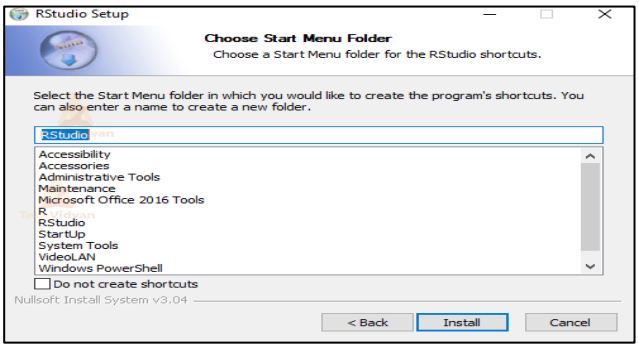
Enter/ browse the path to the installation folder and click Next to proceed.

Select the folder for the start menu shortcut or click on do not create shortcuts and then click Next. Wait for the installation process to complete.

Click Finish to end the installation.

Output :





Install the R Packages:-

- First, run RStudio.
- After clicking on the packages tab, click on install. The following dialog box will appear.
- In the Install Packages dialog, write the package name you want to install under the Packages field and then click install. This will install the packageyousearchedfororgiveyoualistofmatchingpackagesbasedonyour package text.

Installing Packages:-

Loading Packages:-

Once the package is downloaded to your computer you can access the functions and Resources provided by the package in two different ways:

#load the package to use in the current R session

library (package name)

Getting Help on Packages:-

"C:/Program Files/R/R-3.2.2/library"

install.packages("Package Name")

Install the package named "XML".

install.packages("XML")

2. Learn all the basics of R-Programming (Data types, Variables, Operators etc.)

Program Description :

Variables are nothing but reserved memory locations to store values. This means that, when create a variable you reserve some space in memory.

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.

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. R language is rich in built-in operators and provides following types of operators.

Data Types :

Numeric :

```
v <-23.5
print(class(v))
```

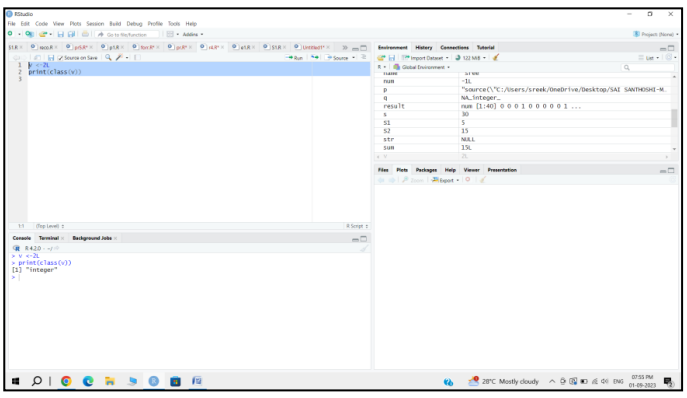
Logical

```
v <- TRUE
print(class(v))
```

Integer

```
v <-2L
print(class(v))
```

Output :



R-objects.

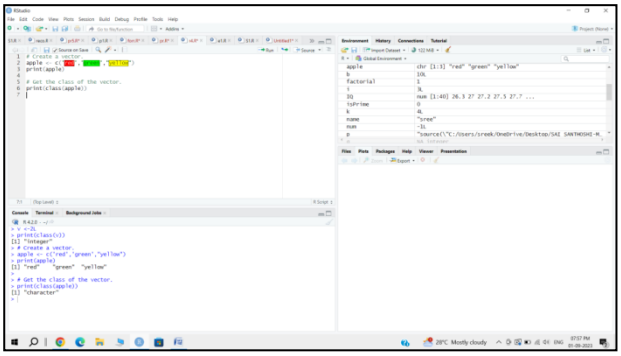
- Vectors
- Lists
- Matrices
- Arrays
- Factors
- Data Frames

Vectors

When you want to create vector with more than one element, you should use **c()** function which means to combine the elements into a vector.

```
# Create a vector.

apple <- c('red','green',"yellow")

print(apple)


# Get the class of the vector.

print(class(apple))
```

Output :

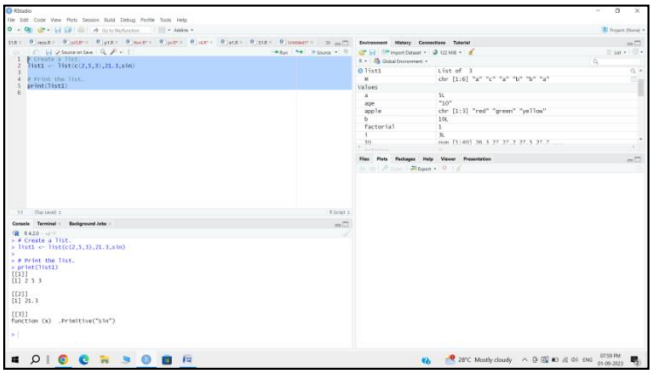


Lists

A list is an R-object which can contain many different types of elements inside it like vectors, functions and even another list inside it.

```
# Create a list.
list1 <- list(c(2,5,3),21.3,sin)

# Print the list.
print(list1)
```

Output :

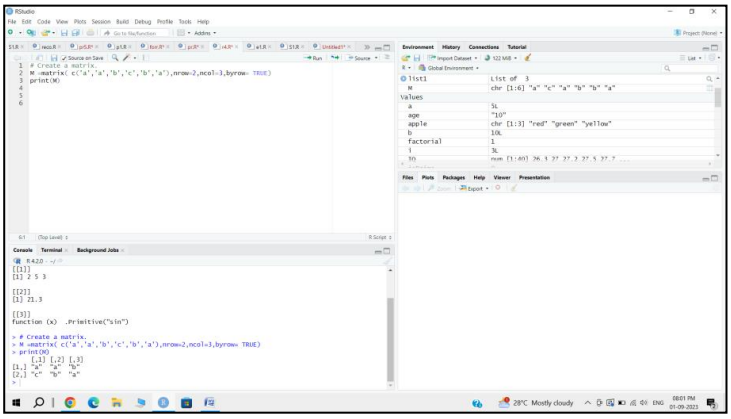


Matrices

A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

```
# Create a matrix.
M =matrix( c('a','a','b','c','b','a'),nrow=2,ncol=3,byrow= TRUE)
print(M)
```

Output :

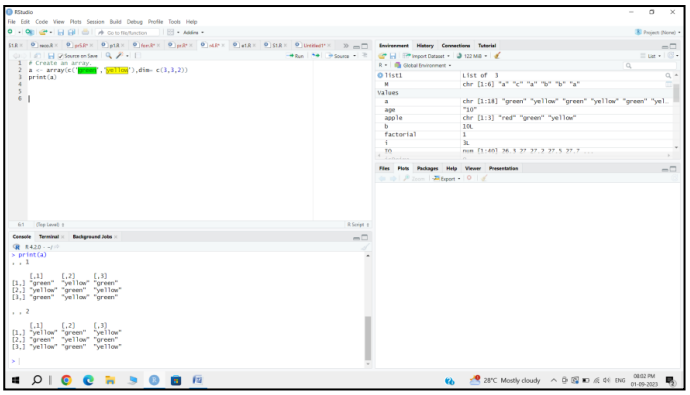


Arrays

While matrices are confined to two dimensions, arrays can be of any number of dimensions. The array function takes a dim attribute which creates the required number of dimension. In the below example we create an array with two elements which are 3x3 matrices each.

```
# Create an array.
a <- array(c('green','yellow'),dim= c(3,3,2))
print(a)
```

Output :



Factors

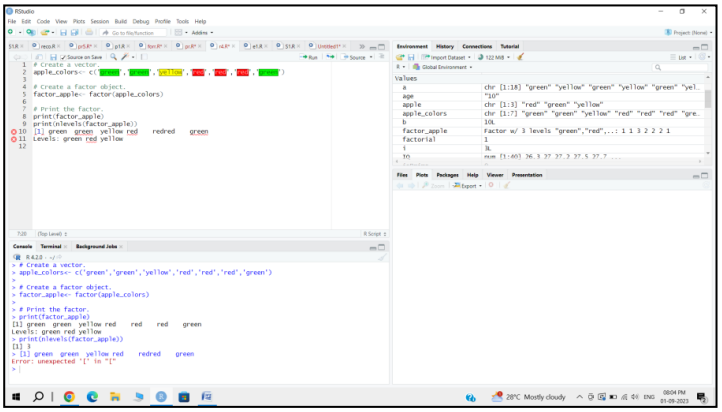
Factors are the R-objects which are created using a vector. It stores the vector along with the distinct values of the elements in the vector as labels. The labels are always character irrespective of whether it is numeric or character or Boolean etc. in the input vector. They are useful in statistical modeling.

Factors are created using the **factor()** function. The **nlevels** functions gives the count of levels.

```
# Create a vector.
apple_colors<- c('green','green','yellow','red','red','red','green')

# Create a factor object.
factor_apple<- factor(apple_colors)

# Print the factor.
print(factor_apple)
print(nlevels(factor_apple))
```

```
 [1] green  green  yellow red    redred    green
Levels: green red yellow
```

Output :

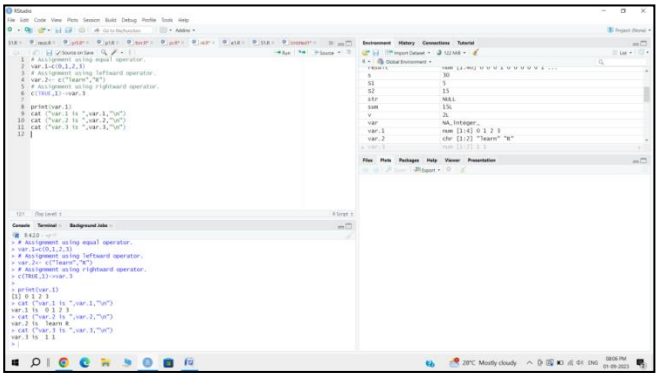


Variables:

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")
```

Output :



R Operators :

Types of Operators

Arithmetic Operators

```
v <- c( 2,5.5,6)
t <- c(8, 3, 4)
print(v+t)
```

Relational Operators

```
v <- c(2,5.5,6,9)
t <- c(8,2.5,14,9)
print(v>t)
```

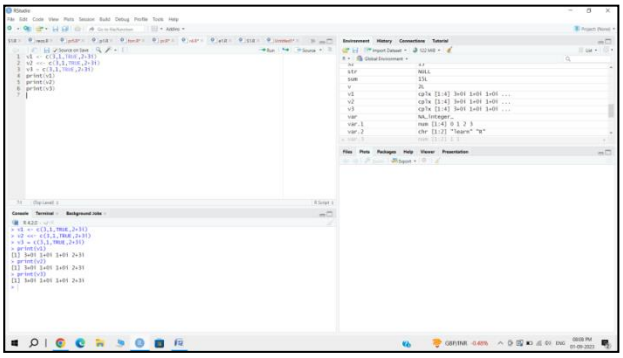
Logical Operators

```
v <- c(3,1,TRUE,2+3i)
t <- c(4,1,FALSE,2+3i)
print(v&t)
```

Assignment Operators

```
v1 <- c(3,1,TRUE,2+3i)
v2 <<- c(3,1,TRUE,2+3i)
v3 = c(3,1,TRUE,2+3i)
print(v1)
print(v2)
print(v3)
```

Output :



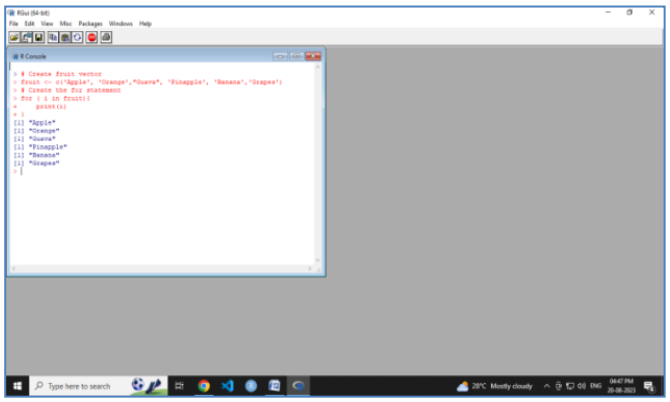
3 Implement R-Loops with different examples.

Program Description :

A for loop is the most popular control flow statement. A for loop is used to iterate a vector. It is similar to the while loop. There is only one difference between for and while, i.e., in while loop, the condition is checked before the execution of the body, but in for loop condition is checked after the execution of the body.

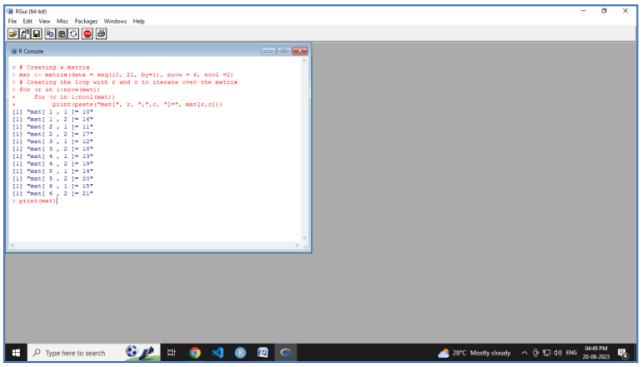
```
# Create fruit vector
fruit <- c('Apple', 'Orange',"Guava", 'Pinapple', 'Banana','Grapes')
# Create the for statement
for ( i in fruit){
   print(i)
}
```

Output :



```
# Creating a matrix

mat <- matrix(data = seq(10, 21, by=1), nrow = 6, ncol =2)
# Creating 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("mat[", r, ",",c, "]=", mat[r,c]))
print(mat)
```

Output :



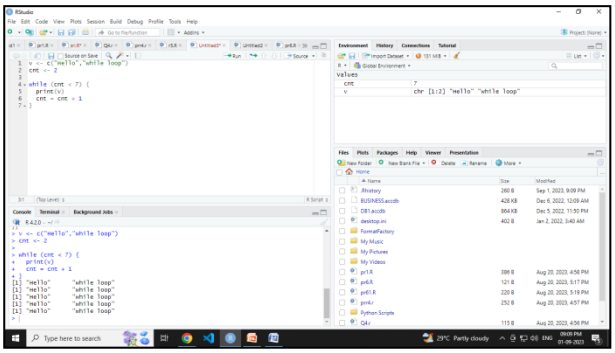
R while loop :

A while loop is a type of control flow statements which is used to iterate a block of code several numbers of times. The while loop terminates when the value of the Boolean expression will be false.

In while loop, firstly the condition will be checked and then after the body of the statement will execute. In this statement, the condition will be checked n+1 time, rather than n times.

```
v <- c("Hello","while loop")
cnt <- 2
while (cnt < 7) {
  print(v)
  cnt = cnt + 1
}
```

Output :



4. Learn the basics of functions in R and implement with examples.

Program Description :

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.

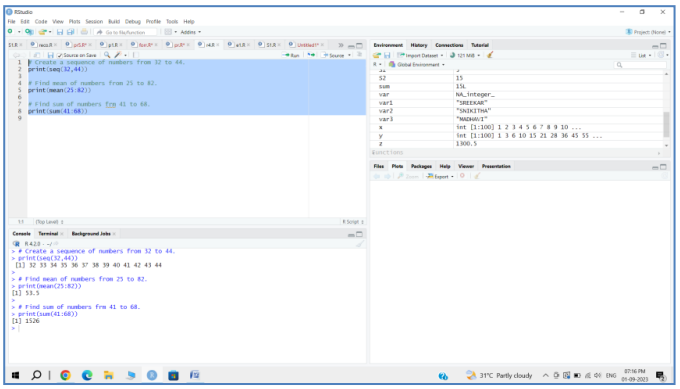
In R, a function is an object so the R interpreter is able to pass control to the function, along with arguments that may be necessary for the function to accomplish the actions.

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.

Built-in Function

```
# 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))
```

Output :



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)
  }
}

# Call the function new.function supplying 6 as an argument.
new.function(6)
```

5. Implement data frames in R. Write a program to join columns and rows in a data frame using cbind() and rbind() in R.

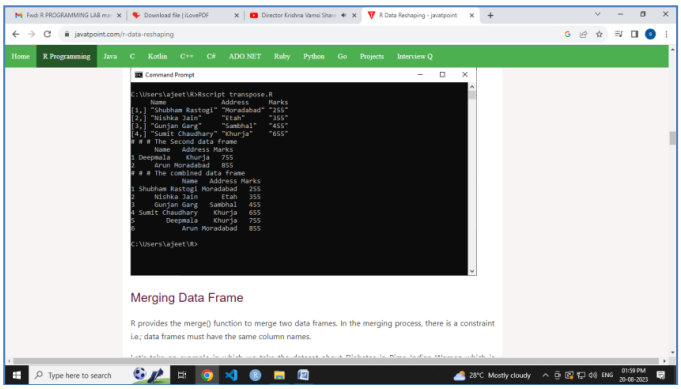
Program Description :

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

```
#Creating vector objects
Name <- c("Shubham Rastogi","Nishka Jain","Gunjan Garg","Sumit Chaudhary")
Address <- c("Moradabad","Etah","Sambhal","Khurja")
Marks <- c(255,355,455,655)
 #Combining vectors into one data frame
info <- cbind(Name,Address,Marks)
 #Printing data frame
print(info)
 # Creating another data frame with similar columns
new.stuinfo <- data.frame(
   Name = c("Deepmala","Arun"),
   Address = c("Khurja","Moradabad"),
   Marks = c("755","855"),
   stringsAsFactors=FALSE
)
```

```
  #Printing a header.
cat("# # # The Second data frame\n")

#Printing the data frame.
print(new.stuinfo)
  # Combining rows form both the data frames.
all.info <- rbind(info,new.stuinfo)
  # Printing a header.
cat("# # # The combined data frame\n")
  # Printing the result.
print(all.info)
```

Output :



6. Implement different String Manipulation functions in R

Program Description :

String manipulation basically refers to the process of handling and analyzing strings. It involves various operations concerned with modification and parsing of strings to use and change its data. R offers a series of in-built functions to manipulate the contents of a string. In this article, we will study different functions concerned with the manipulation of strings in R.

Concatenation of Strings

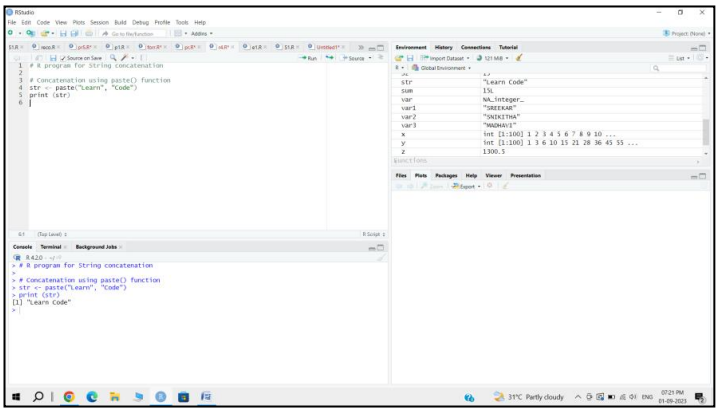
String Concatenation is the technique of combining two strings. String Concatenation can be done using many ways:

- **paste() function** Any number of strings can be concatenated together using the **paste()** function to form a larger string. This function takes separator as argument which is used between the individual string elements and another argument 'collapse' which reflects if we wish to print the strings together as a single larger string. By default, the value of collapse is NULL

pr-1

```
# R program for String concatenation


# Concatenation using paste() function
str <- paste("Learn", "Code")
print (str)
```

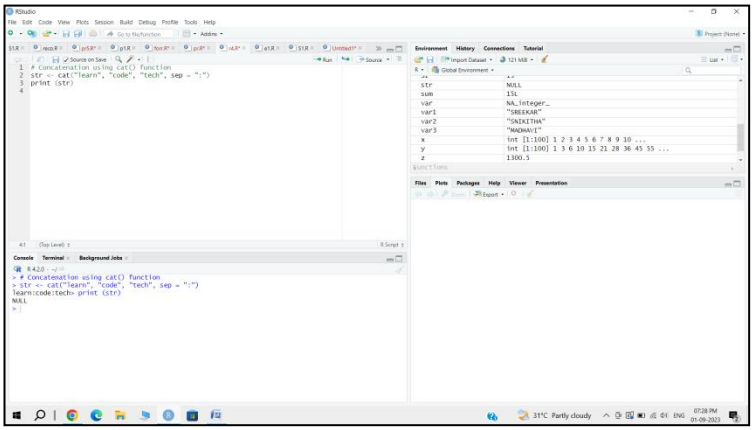
Output :



pr-2

```
# Concatenation using cat() function
str <- cat("learn", "code", "tech", sep = ":")
print (str)
```

Output :



7 Implement different data structures in R (Vectors, Lists, Data Frames)

Program Description :

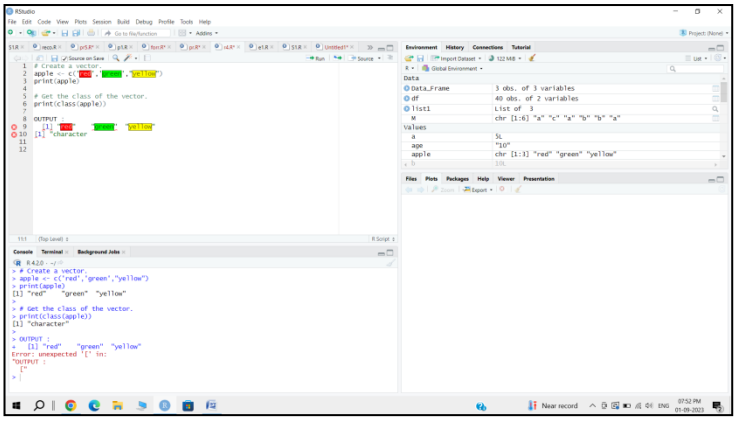
Vectors are the most basic R data objects and there are six types of atomic vectors. They are logical, integer, double, complex, character and raw.

Lists are the R objects which contain elements of different types like − numbers, strings, vectors and another list inside it. A list can also contain a matrix or a function as its elements. List is created using list() function.

Vectors

```
# Create a vector.
apple <- c('red','green',"yellow")
print(apple)

# Get the class of the vector.
print(class(apple))
```

Output :



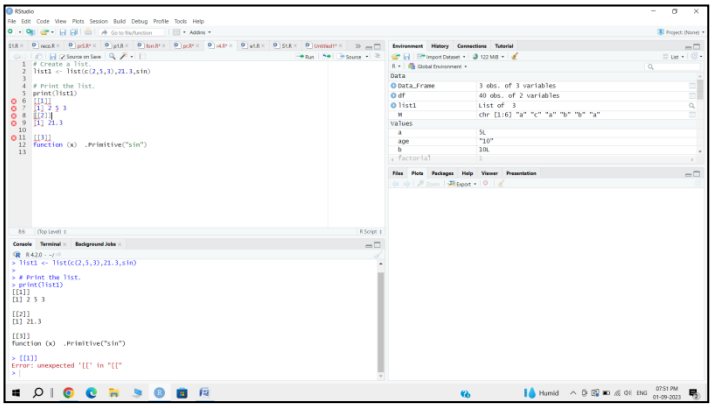
Lists

A list is an R-object which can contain many different types of elements inside it like vectors, functions and even another list inside it.

```
# Create a list.
list1 <- list(c(2,5,3),21.3,sin)

# Print the list.
print(list1)
```

```
[[1]]
[1] 2 5 3
[[2]]
[1] 21.3

[[3]]
function (x)  .Primitive("sin")
```

Output :

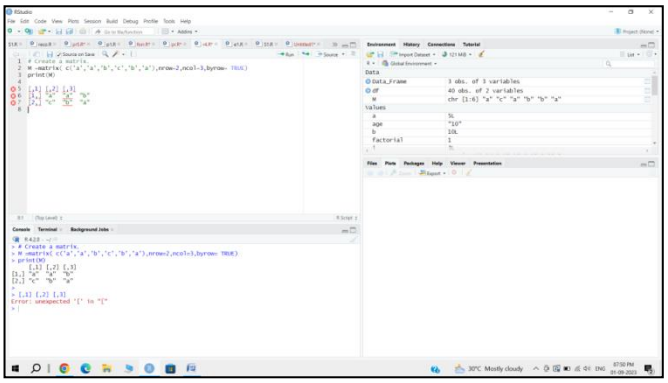


Matrices

A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

```
# Create a matrix.
M =matrix ( c ('a','a','b','c','b','a'),nrow=2,ncol=3,byrow= TRUE)
print(M)
```

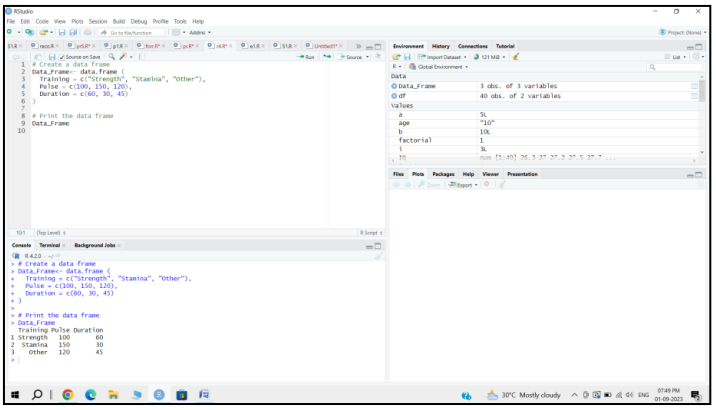
Output :



Data Frames :

```
# Create a data frame
Data_Frame<- data.frame (
  Training = c("Strength", "Stamina", "Other"),
  Pulse = c(100, 150, 120),
  Duration = c(60, 30, 45)
)

# Print the data frame
Data_Frame
```

Output :



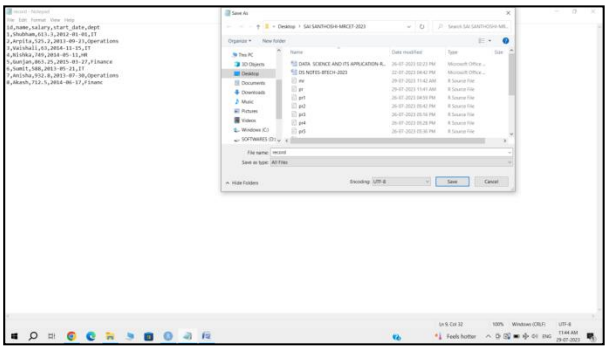
8. Write a program to read a csv file and analyze the data in the file in R

Program Description :

In R, we can read data from files stored outside the R environment. We can also write data into files which will be stored and accessed by the operating system. R can read and write into various file formats like csv, excel, xml etc.

```
# Getting and printing current working directory.
print(getwd())
# Setting the current working directory.
setwd("C:\Users\sreek\OneDrive\Desktop\SAI SANTHOSHI-MRCET-2023")
# Getting and printingthe current working directory.
print(getwd())
```

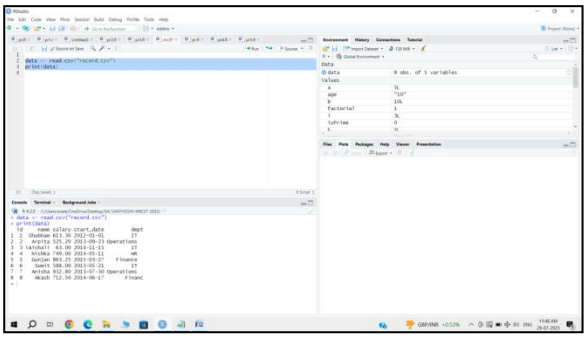
Output :



Reading a CSV file

```
data <- read.csv("record.csv")
print(data)
```

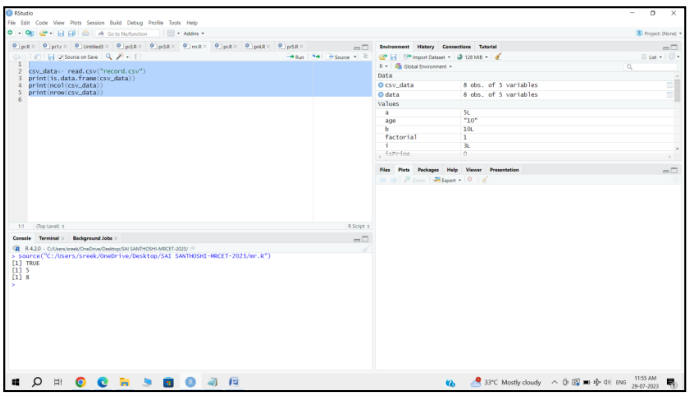
Output :



Analyzing the CSV File

```
csv_data<- read.csv("record.csv")
print(is.data.frame(csv_data))
print(ncol(csv_data))
print(nrow(csv_data))
```

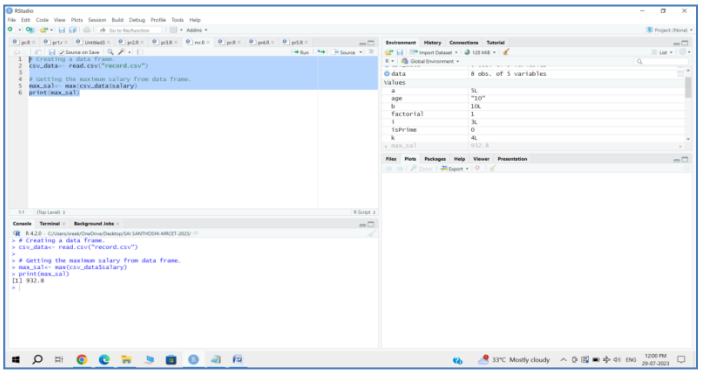
Output :



Getting the maximum salary

```
# Creating a data frame.
csv_data<- read.csv("record.csv")
  # Getting the maximum salary from data frame.
max_sal<- max(csv_data$salary)
print(max_sal)
```

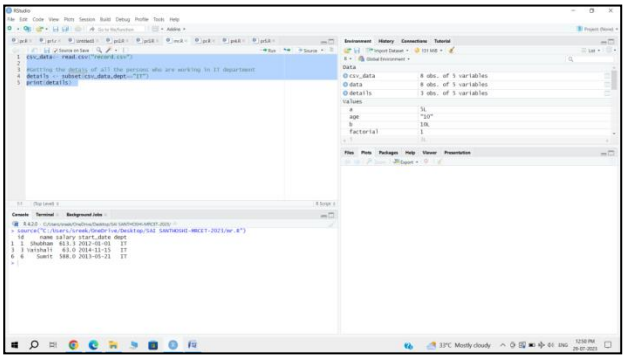
Output :



Getting the details of all the persons who are working in the IT department

```
# Creating a data frame.
csv_data<- read.csv("record.csv")
  #Getting the detais of all the pweson who are working in IT department
details <- subset(csv_data,dept=="IT")
print(details)
```

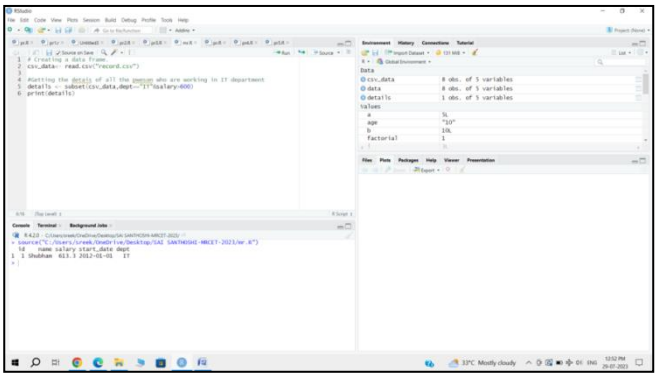
Output :



Getting the details of the persons whose salary is greater than 600 and working in the IT department.

```
# Creating a data frame.
csv_data<- read.csv("record.csv")
  #Getting the detais of all the pweson who are working in IT department
details <- subset(csv_data,dept=="IT"&salary>600)
print(details)
```

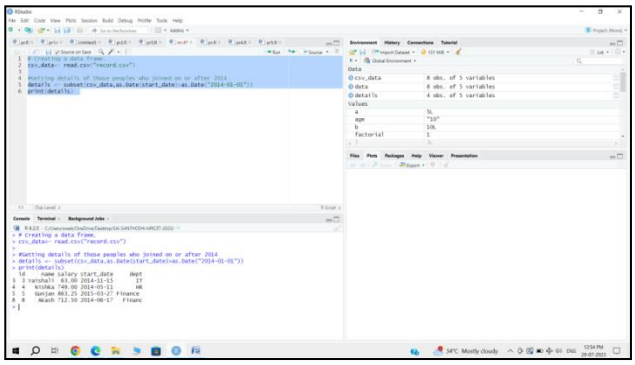
Output :



Getting details of those peoples who joined on or after 2014.

```
# Creating a data frame.
csv_data<- read.csv("record.csv")
  #Getting details of those peoples who joined on or after 2014
details <- subset(csv_data,as.Date(start_date)>as.Date("2014-01-01"))
print(details)
```

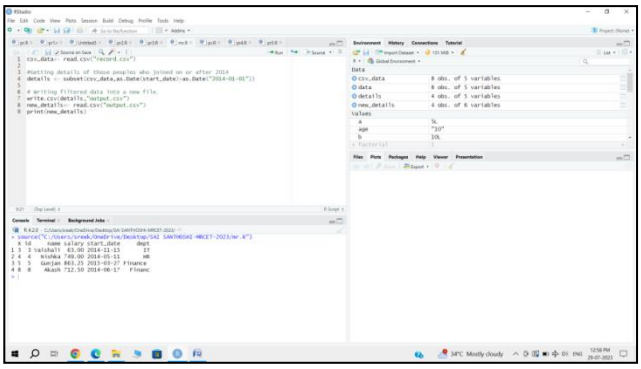
Output :



Writing into a CSV file:

```
csv_data<- read.csv("record.csv")
  #Getting details of those peoples who joined on or after 2014
details <- subset(csv_data,as.Date(start_date)>as.Date("2014-01-01"))
  # Writing filtered data into a new file.
write.csv(details,"output.csv")
new_details<- read.csv("output.csv")
print(new_details)
```

Output :



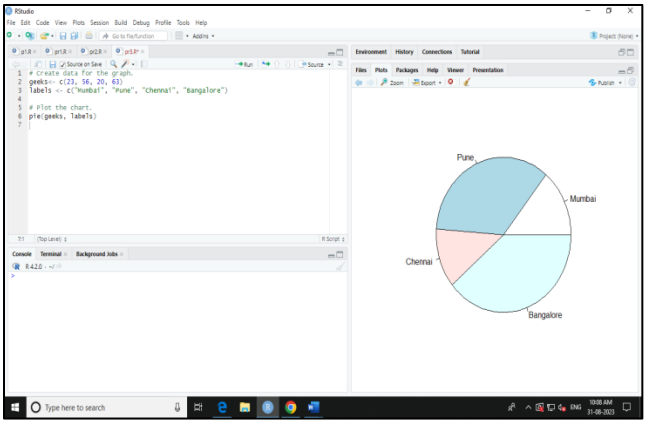
9. Create pie charts and bar charts using R

Program Description :

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

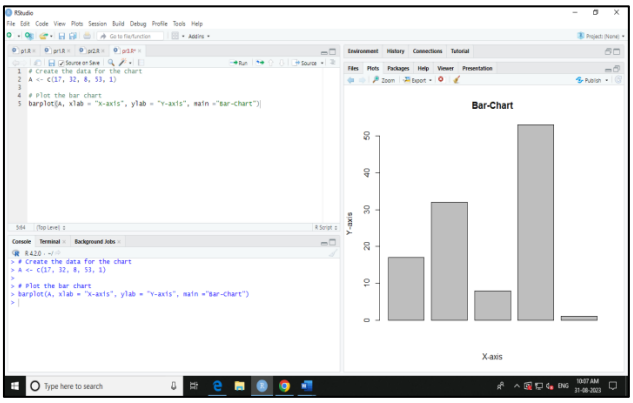
```
# Create data for the graph.
geeks<- c(23, 56, 20, 63)
labels <- c("Mumbai", "Pune", "Chennai", "Bangalore")

# Plot the chart.
pie(geeks, labels)
```

Output :



```
# Create the data for the chart
A <- c(17, 32, 8, 53, 1)

# Plot the bar chart
barplot(A, xlab = "X-axis", ylab = "Y-axis", main ="Bar-Chart")
```

Output :



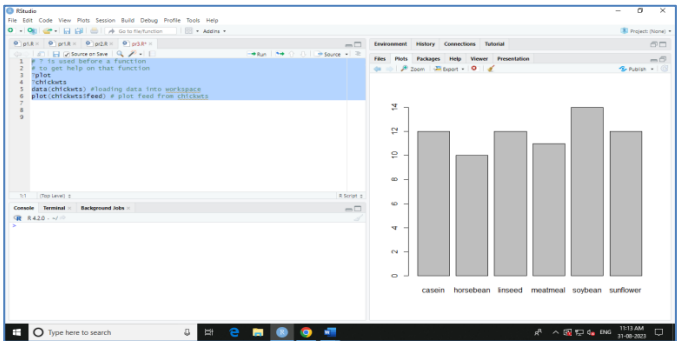
10. Create a data set and do statistical analysis on the data using R

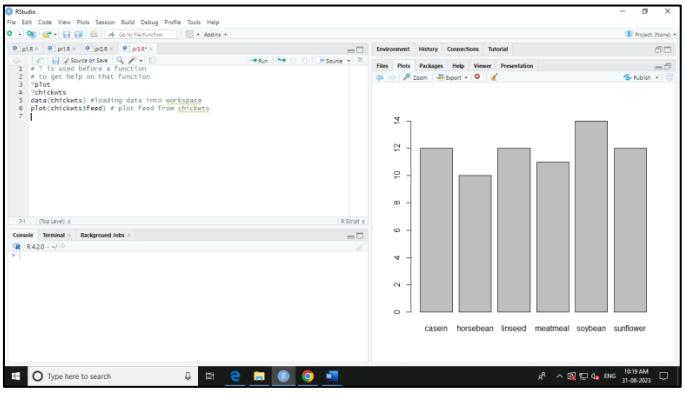
Program Description :

The R Programming Language provides some easy and quick tools that let us convert our data into visually insightful elements like graphs.

```
# ? is used before a function
# to get help on that function
?plot
?chickwts
data(chickwts) #loading data into workspace
plot(chickwts$feed) # plot feed from chickwts
feeds=table(chickwts$feed)
 # plots graph in decreasing order
barplot(feeds[order(feeds, decreasing=TRUE)])
```

Output :





11. Write R program to find Correlation and Covariance

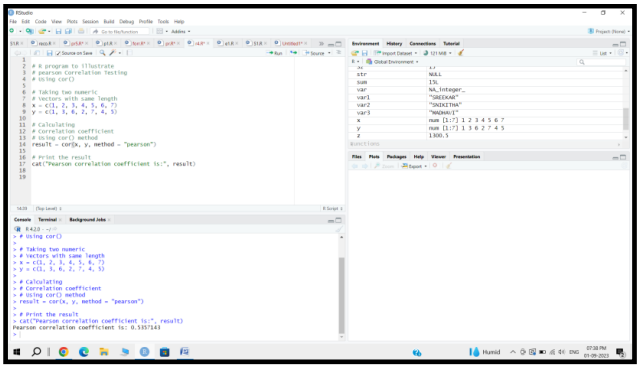
Program Description :

Covariance shows the direction of the path of the linear relationship between the variables while a function is applied to them.

Correlation on the contrary measures both the power and direction of the linear relationship between two variables.

```
# R program to illustrate
# pearson Correlation Testing
# Using cor()
# Taking two numeric
# Vectors with same length
x = c(1, 2, 3, 4, 5, 6, 7)
y = c(1, 3, 6, 2, 7, 4, 5)

# Calculating
# Correlation coefficient
# Using cor() method
result = cor(x, y, method = "pearson")
# Print the result
cat("Pearson correlation coefficient is:", result)
```

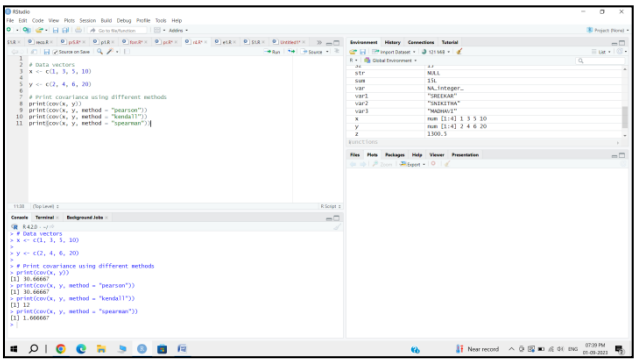
Output :



Covariance

```
# Data vectors
x <- c(1, 3, 5, 10)

y <- c(2, 4, 6, 20)

# Print covariance using different methods
print(cov(x, y))
print(cov(x, y, method = "pearson"))
print(cov(x, y, method = "kendall"))
print(cov(x, y, method = "spearman"))
```

Output :



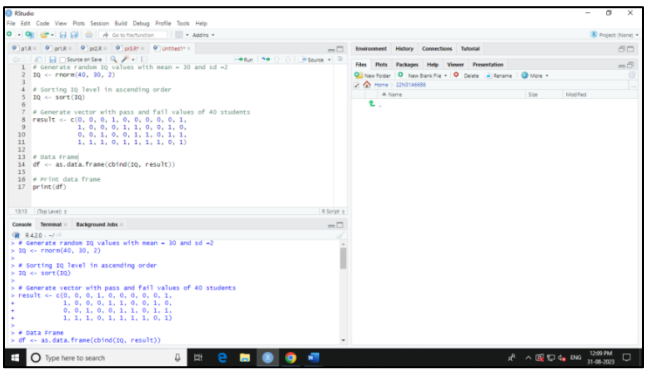
12. Write R program for Regression Modeling

Program Description :

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

```
# Generate random IQ values with mean = 30 and sd =2
IQ <- rnorm(40, 30, 2)

# Sorting IQ level in ascending order
IQ <- sort(IQ)

# Generate vector with pass and fail values of 40 students
result <- c(0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
            1, 0, 0, 0, 1, 1, 0, 0, 1, 0,
            0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
            1, 1, 1, 0, 1, 1, 1, 1, 0, 1)
# Data Frame
df<- as.data.frame(cbind(IQ, result))
# Print data frame
print(df)
```

Output :



13 .Write R program to build classification model using KNN algorithm

Program Description :

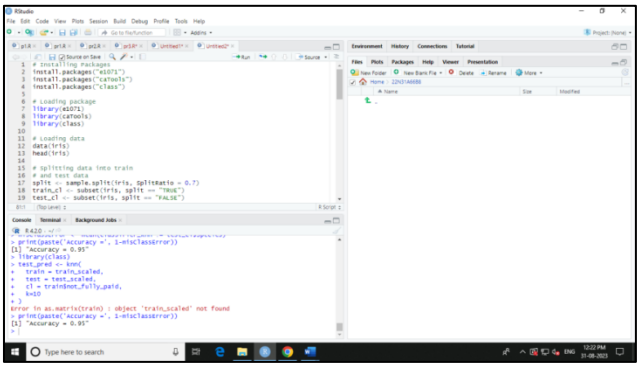
K-Nearest Neighbor or K-NN is a Supervised Non-linear classification algorithm. K-NN is a Non-parametric algorithm i.e it doesn't make any assumption about underlying data or its distribution. It is one of the simplest and widely used algorithm which depends on it's k value(Neighbors) and finds it's applications in many industries like finance industry, healthcare industry etc.

```
# Loading data
data(iris)
# Structure
str(iris)
# Installing Packages
install.packages("e1071")
install.packages("caTools")
install.packages("class")
  # Loading package
library(e1071)
library(caTools)
library(class)
  # Loading data
data(iris)
head(iris)
  # Splitting data into train
# and test data
split <-sample.split(iris, SplitRatio= 0.7)
train_cl<-subset(iris, split == "TRUE")
test_cl<-subset(iris, split == "FALSE")

# Feature Scaling
train_scale<-scale(train_cl[, 1:4])
test_scale<-scale(test_cl[, 1:4])
```

```
# Fitting KNN Model
# to training dataset
classifier_knn<-knn(train =train_scale,
                    test =test_scale,
                    cl =train_cl$Species,
                    k = 1)
classifier_knn

# Confusiin Matrix
cm <-table(test_cl$Species, classifier_knn)
cm

# Model Evaluation - Choosing K
# Calculate out of Sample error
misClassError<-mean(classifier_knn !=test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 3
classifier_knn<-knn(train =train_scale,
                    test =test_scale,
                    cl =train_cl$Species,
                    k = 3)
misClassError<-mean(classifier_knn !=test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 5
classifier_knn<-knn(train =train_scale,
                    test =test_scale,
                    cl =train_cl$Species,
                    k = 5)
misClassError<-mean(classifier_knn !=test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 7
classifier_knn<-knn(train =train_scale,
                    test =test_scale,
                    cl =train_cl$Species,
                    k = 7)
misClassError<-mean(classifier_knn !=test_cl$Species)
print(paste('Accuracy =', 1-misClassError))
```

```
# K = 15
classifier_knn<-knn(train =train_scale,
                    test =test_scale,
                    cl =train_cl$Species,
                    k = 15)
misClassError<-mean(classifier_knn !=test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 19
classifier_knn<-knn(train =train_scale,
                    test =test_scale,
                    cl =train_cl$Species,
                    k = 19)
misClassError<-mean(classifier_knn !=test_cl$Species)
print(paste('Accuracy =', 1-misClassError))
```

Output :



14 Write R program to build clustering model using K-mean algorithm

Program Description :

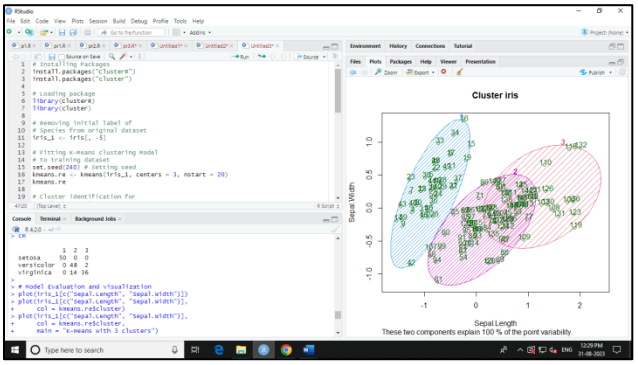
K Means Clustering in R Programming is an Unsupervised Non-linear algorithm that cluster data based on similarity or similar groups. It seeks to partition the observations into a pre-specified number of clusters. Segmentation of data takes place to assign each training example to a segment called a cluster.

```
# Loading data
data(iris)

# Structure
str(iris)
# Installing Packages
install.packages("ClusterR")
install.packages("cluster")

# Loading package
library(ClusterR)
library(cluster)

# Removing initial label of
# Species from original dataset
iris_1 <- iris[, -5]

# Fitting K-Means clustering Model
# to training dataset
set.seed(240) # Setting seed
kmeans.re <- kmeans(iris_1, centers = 3, nstart = 20)
kmeans.re
```

```
# Cluster identification for
# each observation
kmeans.re$cluster

# Confusion Matrix
cm <- table(iris$Species, kmeans.re$cluster)
cm

# Model Evaluation and visualization
plot(iris_1[c("Sepal.Length", "Sepal.Width")])
plot(iris_1[c("Sepal.Length", "Sepal.Width")],
    col = kmeans.re$cluster)
plot(iris_1[c("Sepal.Length", "Sepal.Width")],
    col = kmeans.re$cluster,
    main = "K-means with 3 clusters")

## Plotiing cluster centers
kmeans.re$centers
kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")]

# cex is font size, pch is symbol
points(kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")],
      col = 1:3, pch = 8, cex = 3)
```

```
## Visualizing clusters
y_kmeans<- kmeans.re$cluster
clusplot(iris_1[, c("Sepal.Length", "Sepal.Width")],
y_kmeans,
        lines = 0,
        shade = TRUE,
        color = TRUE,
        labels = 2,
plotchar = FALSE,
        span = TRUE,
        main = paste("Cluster iris"),
xlab = 'Sepal.Length',
ylab = 'Sepal.Width')
```

Output :



***