**16. Box Plot**

**AIM:**

To write a R Program to create and analyze boxplots in R using numeric vectors, built-in datasets, and CSV files.

**ALGORITHM:**

**Step 1:** Start the process to visualize the distribution of data using boxplots.  
**Step 2:** Open RStudio and write the program using vectors, built-in datasets, and CSV files.  
**Step 3:** Create numeric vectors and apply the summary() function to calculate basic statistics such as min, max, median, and quartiles.  
**Step 4:** Generate boxplots for the numeric vectors (both vertical and horizontal) to study spread and outliers.  
**Step 5:** Use the mtcars dataset to plot boxplots of mpg and hp grouped by the number of cylinders.  
**Step 6:** Import a CSV file, extract the required columns (e.g., Salary and Basic), and create boxplots for analyzing the relationship between variables.  
**Step 7:** Customize the boxplots with labels, titles, colors, notches, and widths, and interpret the results for meaningful insights.

**Step 8:** Stop the process

**Program:**

x <- c(10,20,30,40,50)

summary(x)

boxplot(x,horizontal = FALSE)

boxplot(x,horizontal = TRUE)

x<- c(10,11,14,15,120,12,34,54,65,24,67,230)

boxplot(x,horizontal = FALSE)

print(mean(x))

mtcars

input <- mtcars[, c("mpg", "cyl")]

print(input)

boxplot(mpg~cyl,data=mtcars,xlab = "Number of Cylinders ",ylab = "Milege Data",main = "Milege Data")

boxplot(hp~cyl,data=mtcars,xlab = "Number of Cylinders ",ylab = "Horse Power",main = "Power Data")

boxplot(mpg ~ cyl,

data = mtcars,

xlab = "No. of Cylinders",

ylab = "Miles Per Gallon",

main = "Mileage Data",

notch = TRUE,

varwidth = TRUE,

col = c("green", "yellow", "purple"),

names = c("4", "6", "8"))

setwd("D:/24PCA014/Practical/Box Plot")

# Read CSV

df <- read.csv("combined.csv")

print(df)

# Subset only Salary and Basic

v <- df[, c("Salary", "Basic")]

print(v)

# Boxplot with correct case

boxplot(Salary ~ Basic,

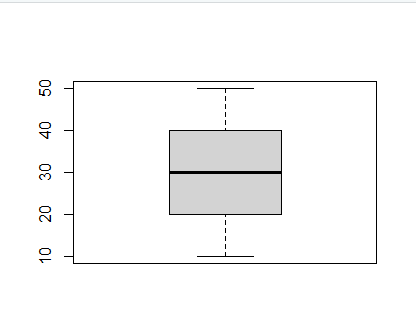
data = v,

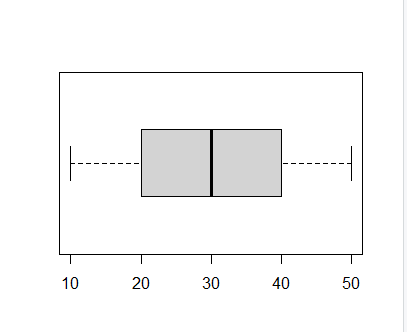
xlab = "Basic",

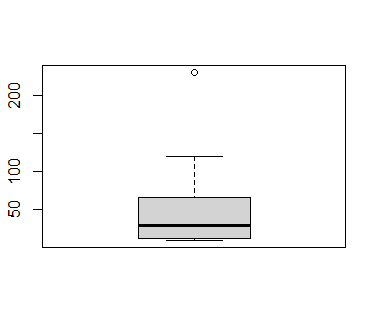
ylab = "Salary",

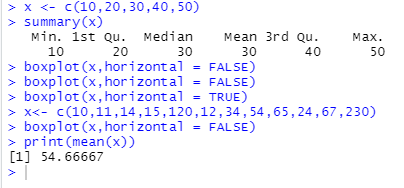
main = "Salary Chart",

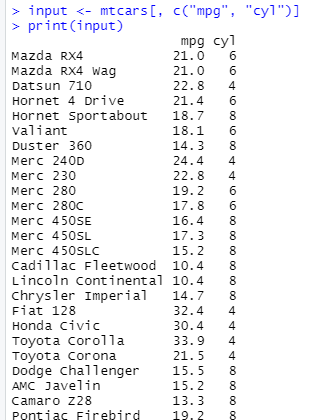
col = "lightblue")

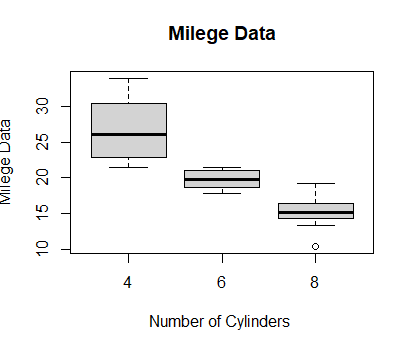
**OUTPUT:  
**

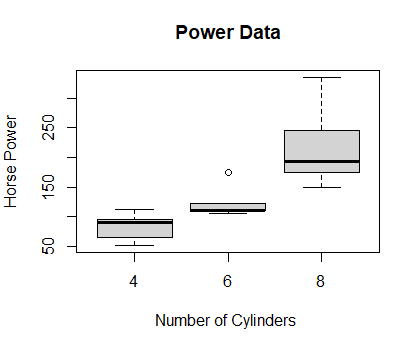
****

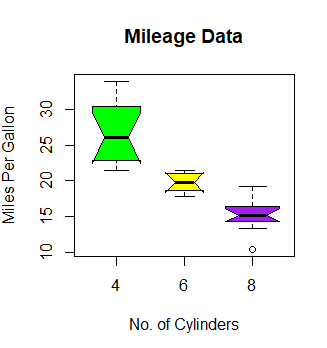
****

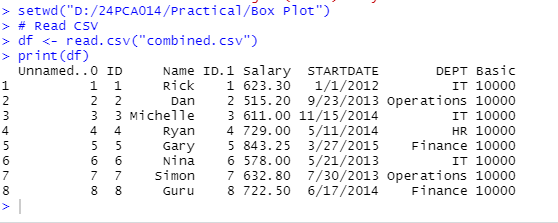
****

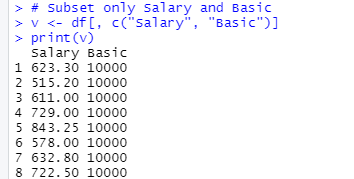
****

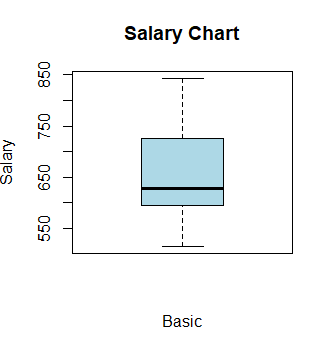
****

****

****

****

****

****

**RESULT:**

Thus, our program has been successfully saved and executed.

**17. Line Chart**

**AIM:**

To write a R Program to read data from a CSV file and plot line charts in R for visualizing and comparing **Salary** and **Basic** values.

**ALGORITHM:**

**Step 1:** Start the process to plot a line chart.  
**Step 2:** Open RStudio and set the working directory to the folder containing the CSV file.  
**Step 3:** Read the CSV file into a data frame using read.csv().  
**Step 4:** Extract the required columns (Salary and Basic) from the data frame.  
**Step 5:** Plot a line chart for **Basic** values using the plot() function.  
**Step 6:** Plot a line chart for **Salary** values using the plot() function.  
**Step 7:** Plot both **Salary** and **Basic** values in the same graph using plot() and lines() functions, and add a legend.  
**Step 8:** End the program.

**PROGRAM:**

v<-c(7,12,28,3,4,1)

print(v)

plot(v,type="o")

plot(v,tyle="l")

plot(c)

plot(v,type="o",col="red",xlab="Month",ylab="Rainfall",main="Rainfall Chart")

#multiple Lines

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

t<-c(14,18,7,6,19,3)

b<-c(15,7,18,19,13)

plot(v,type="o",col="red",xlab="Month",ylab = "Rainfall",main = "Rainfall Chart")

lines(t,type="o",col="green")

lines(b,type="o",col="blue")

colors<-c("green","red","blue")

regions<-c("2005","2010","2020")

legend("topleft",regions,cex=0.2,fill=colors)

t<-0:10

z=exp(t/2)

print(t)

print(z)

plot(t/2,type="l",col="green",lwd=5,xlab="time",ylab="Concentration")

x=-10:110

y=x\*x

plot(x,y,type="o",col="red",lwd=5,xlab="X--Axis",ylab = "Y--Axis")

# Set working directory

setwd("D:/24PCA014/Practical/Line Chart")

# Read CSV

df <- read.csv("combined.csv")

print(df)

# Extract Basic and Salary

basic <- df$Basic

salary <- df$Salary

# Plot Basic

plot(basic, type="o", col="red",

xlab="Person", ylab="Basic",

main="Basic Chart")

# Plot Salary

plot(salary, type="o", col="blue",

xlab="Person", ylab="Salary",

main="Salary Chart")

# Plot both Basic and Salary together

plot(basic, type="o", col="red",

xlab="Person", ylab="Value",

main="Salary vs Basic")

lines(salary, type="o", col="green")

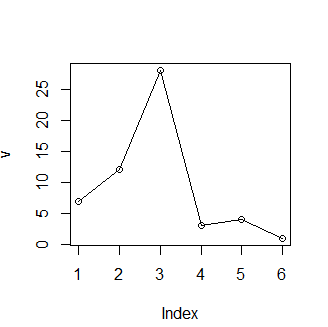
# Add legend

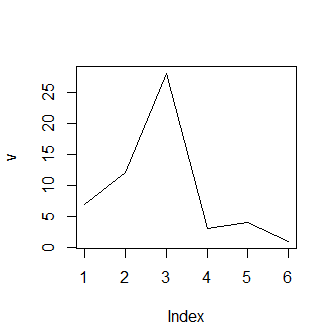
legend("topleft", c("Basic", "Salary"),

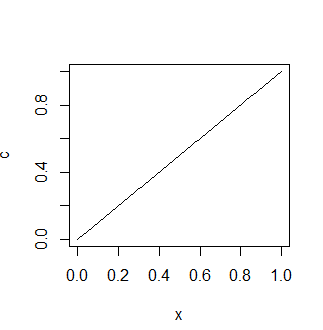
col=c("red","green"),

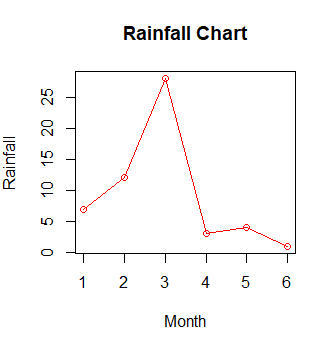
lty=1, pch=1, cex=0.8)

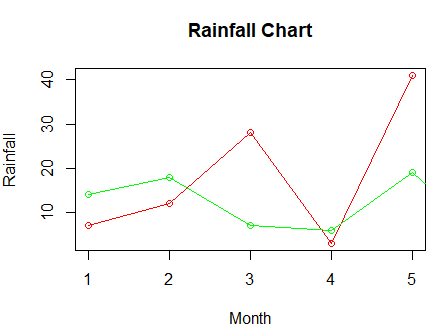
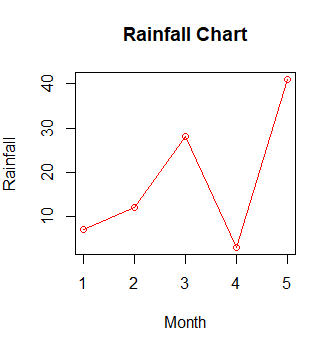
**OUTPUT:**

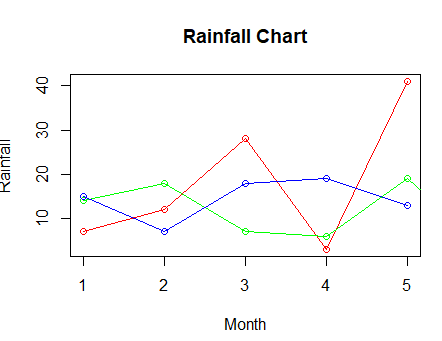
****

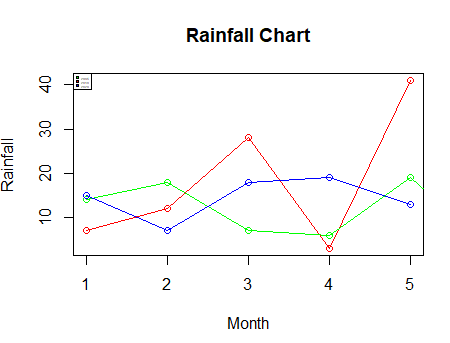
****

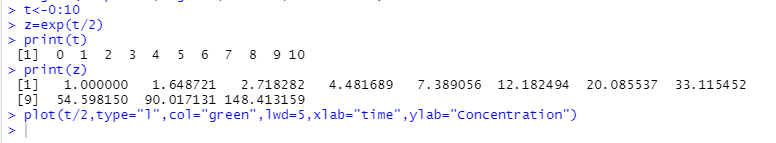
****

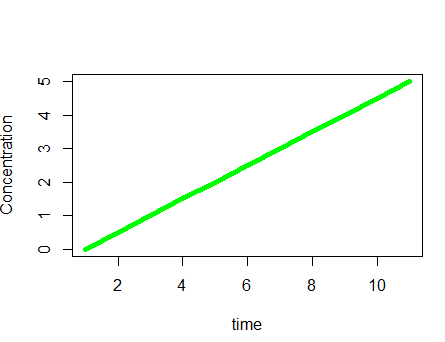
****

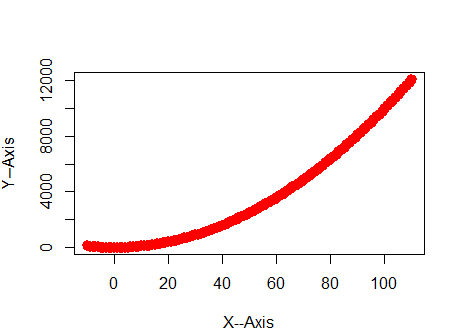
****

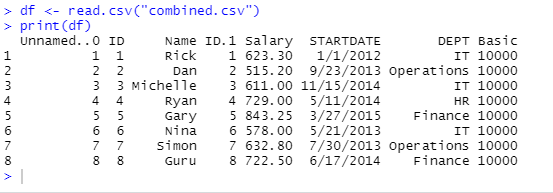
****

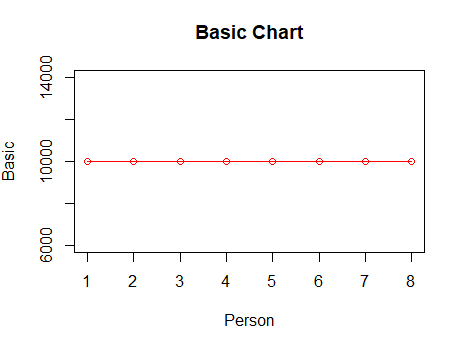
****

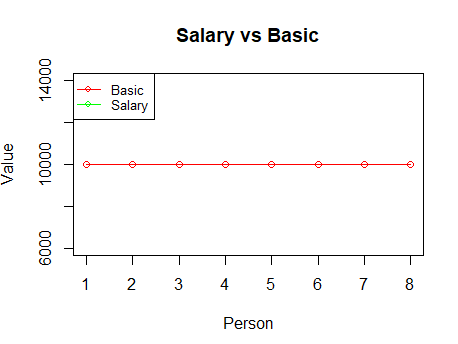
****

****

****

****

****

****

**RESULT:**

Thus, our program has been successfully saved and executed.

**18.Scatter Plot**

**AIM:**

To visualize the relationship between multiple variables using scatter plots and scatter plot matrices in R.

**ALGORITHM:**

**Step 1:** Start the process to create scatter plots in R.  
**Step 2:** Open RStudio and load the built-in dataset (mtcars) into a variable.  
**Step 3:** Select the required columns (e.g., wt, mpg, disp, hp) and use the pairs() function to generate scatter plot matrices.  
**Step 4:** Set the working directory and read external data from a CSV file using read.csv().  
**Step 5:** Extract the required columns (e.g., Salary and Basic) into a new data frame.  
**Step 6:** Use the pairs() function again to create scatter plot matrices for the CSV data.  
**Step 7:** End the program.

**PROGRAM:**

mtcars

input<-mtcars[,c("wt","mpg","disp","cyl")]

pairs(~wt+mpg+disp,data=mtcars,mian="SactterPlot Matrix")

pairs(~wt+mpg+disp+hp,data=mtcars,main="Sactter Plot Matrix")

#work with CSV File

setwd("D:/24PCA014/Practical/Scatter Plot")

df<-read.csv("combined.csv")

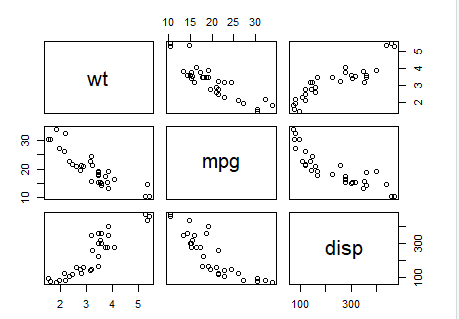
print(df)

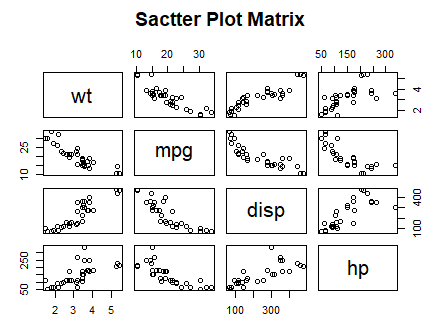
v<-df[,c("Salary","Basic")]

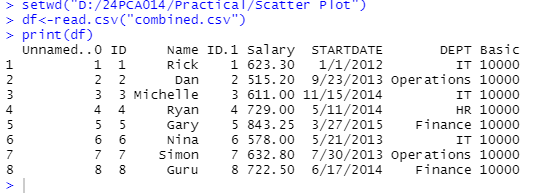
print(v)

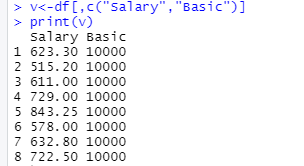
pairs(~Salary+Basic,data = v,main="Scatter Plot Matrix")

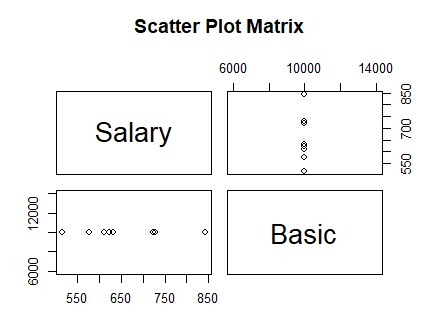
**OUTPUT:**

****

****

****

****

****

**RESULT:**

Thus, our program has been successfully saved and executed.