PRACTICALFILEOF

DATA ANALITICS USING R

SUBMITTED TO-

Mrs. Bindu

Dept. of CSE

SUBMITTEDBY-

Name: Mohit Kumar

Roll: 200010130070

Branch: BTech cse-1

Year/Sem: 3rd Year/6th Sem

GURU JAMBHESHWAR UNIVERSITY OF SCIENCE AND TECHNOLOGY,HISAR

INDEX

08.	Apply summary() command to iris dataset of the 'datasets' package and interpret the output.		24/04/23	
09.	Check and justify the outcome of the following expressions: a. sqrt(3)^2 == 3 b. sqrt(4)^2 == 4 c. near(sqrt(3)^2,3) d. near(sqrt(4)^2,7)		24/04/23	
10.	Install MASS Package and then use apply() to find the measure of the central tendency and dispersion.		24/04/23	
11.	Create a function that gives mean and standard deviation, then save them as object.		24/04/23	
12.	Create a function that gives min, median, and max, then save them as object.		24/04/23	
13.	Write a program to find the population density for each state using mapply().		24/04/23	
14.	Write a program using tapply() to explore population by region.		24/04/23	
15.	Write a script file to compute the following of the numeric variable in Boston dataset. a. Sum b. Range c. Mean d. Standard Deviation		24/04/23	
16.	Assuming the character vector students, having 10 names of students a. Find the character count in each name. b. Find if the names "Akaash" and "Ankit" exist in the vector.		24/04/23	
17.	Write a program to output the indices of the names that contain substring "ii" in vector students of assignment – 16.		24/04/23	
18.	Find out how many strings end with letters "sh" in vector students of assignment-16 .		24/04/23	

19.	Create a vector of factor type data for the hair colors of 10 people, where value of hair colors are:- Black , Blonde , Dark Brown , Grey. a. Display the levels of factor data b. Find the maximum value in vector of hair colors using table() function.		24/04/23	
20.	Create an empty vector of data factor type for the names of first 6 months in a year. Remember to keep the levels of the data in order of months from January to June.		24/04/23	
21**.**	Perform the following operations: a. A+B b. A-B c. 3*A d. Ax=B. x=? e. diag(c(4,1,2,3), nrow=4) f. t(A) g. eigen(A) h. eigen(A*t(A)) i. A%*%B j. b=c(7,4) . A*b		24/04/23	
22.	Apply class, str and summary command to the vector students created in the assignment-16.		31/04/23	
23.	Create a vector to store the grades of 25 students for first minor grade are given as {A, B, C, D}. Compute the modal grade. Further, store the grades of the same students for second minor exam. Compare the grades for 2 exams. Count number of student who have got higher grade.		31/04/23	
24.	Create a 4*3 matrix A of uniformly distributed random integer numbers between 1 to 100. Create another 3*4 matrix B with uniformly distributed random integer numbers between 1 to 10. Perform matrix multiplication of the two matrices and store the result in third matrix C.		31/04/23	
25.	Create A and B, two 4*3 matrices of normally distributed random numbers, with mean 0 and standard deviation 1. Find the indices of all those numbers in matrix A which are less than the respective numbers in matrix B and print these numbers.		31/04/23	

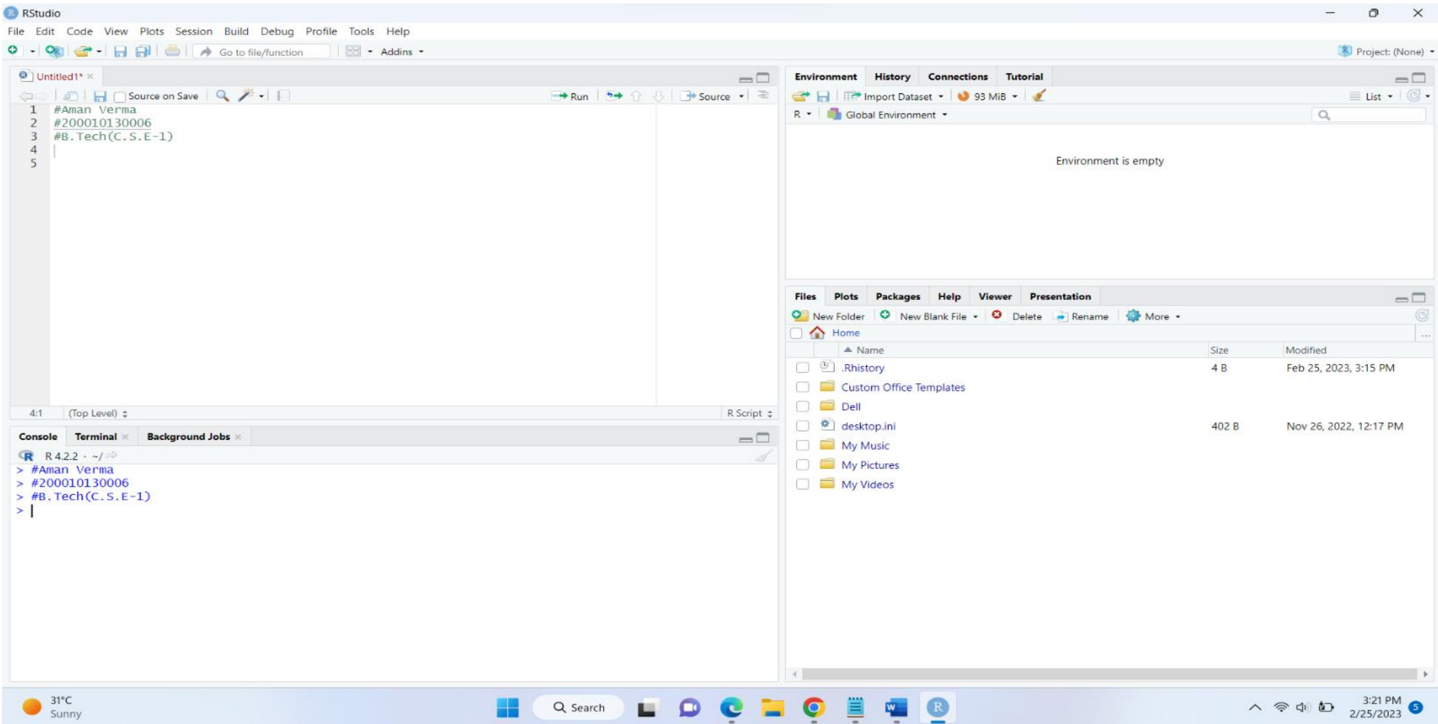
26.	Plotting pressure dataset in different forms: (a) Histogram (b) Boxplot		31/04/23	
27.	Plotting mtcars dataset in Frequency Polygon.		31/04/23	
28.	Plotting scatter plots from iris dataset with title and labels		31/04/23	

PROGRAM

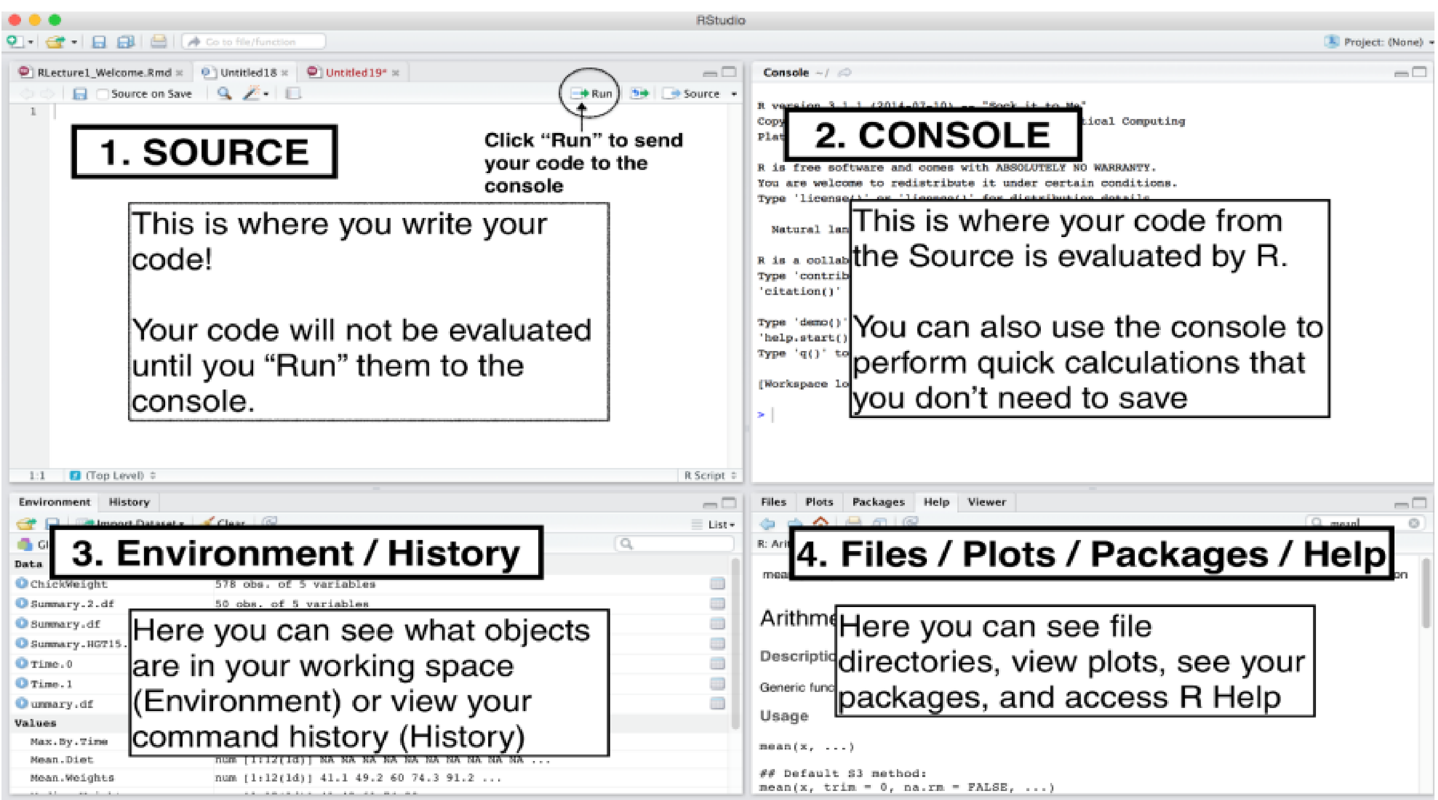
1. Install R and RStudio and learn about GUI of various working windows of RStudio.

Step 1: Search R for Windows on the web browser, select the appropriate version and install.

Step 2: Search RStudio IDE on the browser, select the appropriate version and install.



Various Working Windows:



PROGRAM

2. **Perform the following operations:**
   a. **Create variable of different class & print their class.**

R Code:

#Mohit

#200010130070

Creating variables of different data types

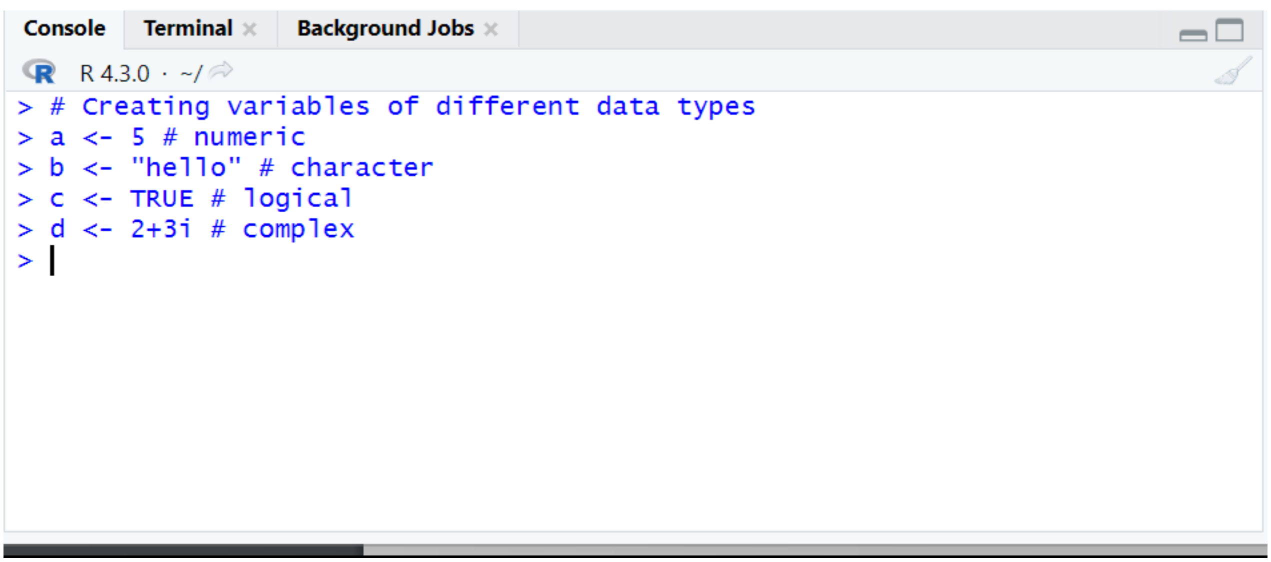
a <- 5 # numeric

b <- "hello" # character

c <- TRUE # logical

d <- 2+3i # complex

<u>Output</u>:



b.Type Conversion.

R Code:

#Perform type conversions

a <- 5

b <- as.character(a)

```
class(b)

c <- "10"
d <- as.numeric(c)
class(d)

e <- TRUE
f <- as.numeric(e)
class(e)

g <- 0
h <- as.logical(g)
class(h)

i <- c("red", "green", "blue", "red", "green")
j <- as.factor(i)
clsas(i)
```

Output:

```
Console  Terminal  Background Jobs
R 4.3.0 · ~/
> a <- 5
> b <- as.character(a)
> class(b)
[1] "character"
> c <- "10"
> d <- as.numeric(c)
> class(d)
[1] "numeric"
> e <- TRUE
> f <- as.numeric(e)
> class(e)
[1] "logical"
> g <- 0
> h <- as.logical(g)
> class(h)
[1] "logical"
> i <- c("red", "green", "blue", "red", "green")
> j <- as.factor(i)
> clsas(i)
```

c.All basic mathematical operations.

R Code:

```
# Addition

a <- 5 + 3

# Subtraction

b <- 5 - 3

# Multiplication

c <- 5 * 3

# Division

d <- 5 / 3

# Integer Division

e <- 5 %/% 3

# Modulo

f <- 5 %% 3

# Exponentiation
```

g <- 5 ^ 3

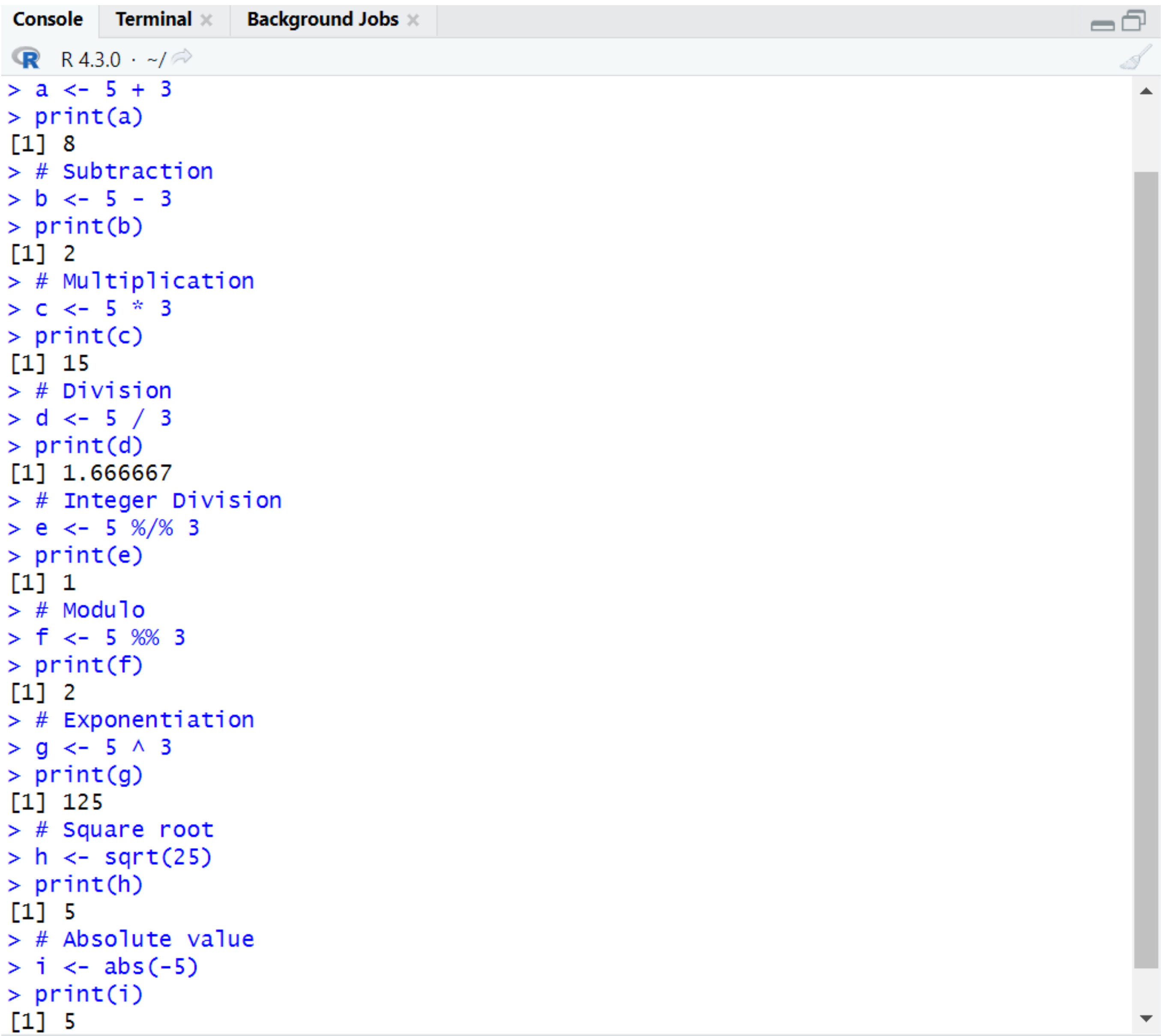
Square root

h <- sqrt(25)

Absolute value

i <- abs(-5)

Output:




```
> a <- 5 + 3
> print(a)
[1] 8
> # Subtraction
> b <- 5 - 3
> print(b)
[1] 2
> # Multiplication
> c <- 5 * 3
> print(c)
[1] 15
> # Division
> d <- 5 / 3
> print(d)
[1] 1.666667
> # Integer Division
> e <- 5 %/% 3
> print(e)
[1] 1
> # Modulo
> f <- 5 %% 3
> print(f)
[1] 2
> # Exponentiation
> g <- 5 ^ 3
> print(g)
[1] 125
> # Square root
> h <- sqrt(25)
> print(h)
[1] 5
> # Absolute value
> i <- abs(-5)
> print(i)
[1] 5
```

PROGRAM

3. **Create two vectors and find their elementwise addition, subtraction and multiplication, concatenate the two vectors and find their sum and average.**

R Code:

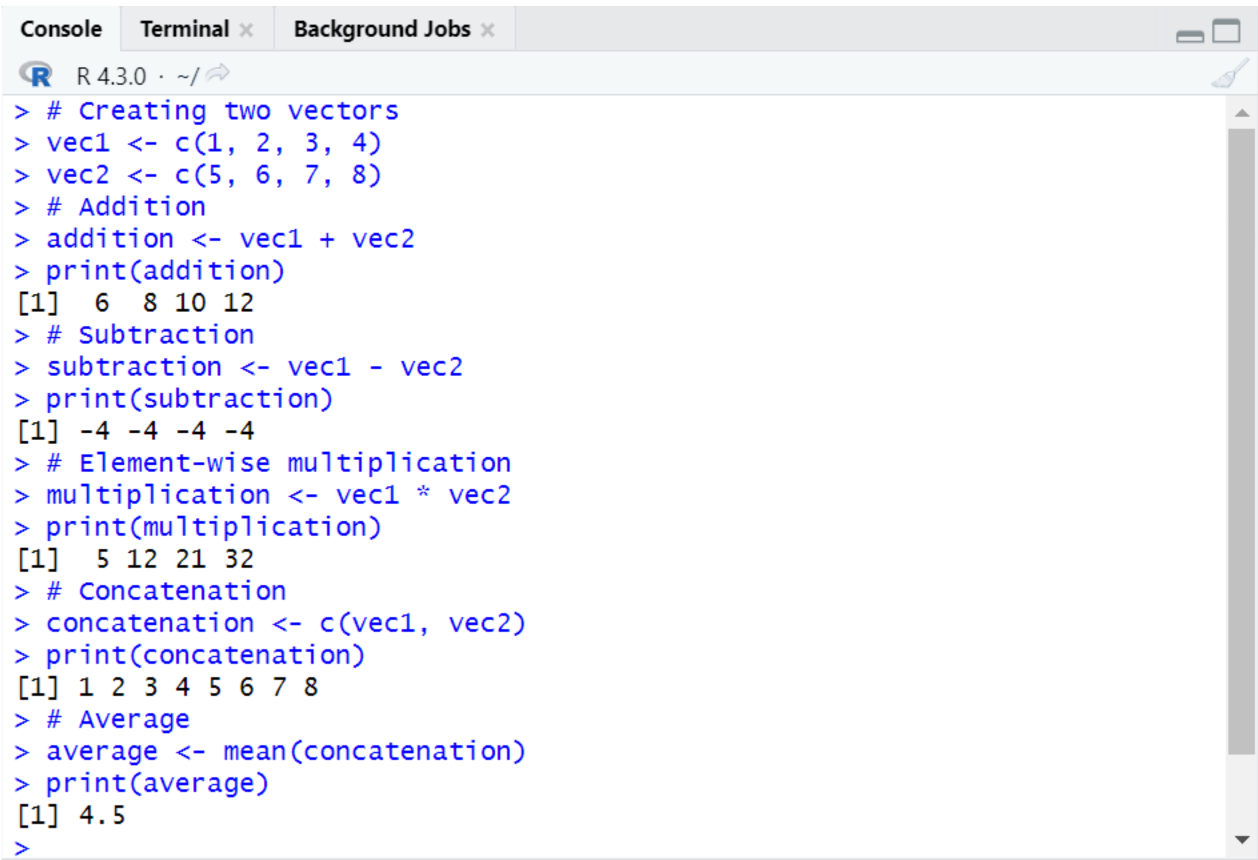
```
#Mohit

#200010130070


# Creating two vectors

vec1 <- c(1, 2, 3, 4)

vec2 <- c(5, 6, 7, 8)


# Addition

addition <- vec1 + vec2

print(addition)


# Subtraction

subtraction <- vec1 - vec2

print(substraction)


# Element-wise multiplication

multiplication <- vec1 * vec2

print(multiplication)


# Concatenation

concatenation <- c(vec1, vec2)

print(concatenation)
```

Average

average <- mean(concatenation)

print(average)

Output:




```
> # Creating two vectors
> vec1 <- c(1, 2, 3, 4)
> vec2 <- c(5, 6, 7, 8)
> # Addition
> addition <- vec1 + vec2
> print(addition)
[1]  6  8 10 12
> # Subtraction
> subtraction <- vec1 - vec2
> print(subtraction)
[1] -4 -4 -4 -4
> # Element-wise multiplication
> multiplication <- vec1 * vec2
> print(multiplication)
[1]  5 12 21 32
> # Concatenation
> concatenation <- c(vec1, vec2)
> print(concatenation)
[1] 1 2 3 4 5 6 7 8
> # Average
> average <- mean(concatenation)
> print(average)
[1] 4.5
>
```

PROGRAM

4. **Create a vector of all those values from 1:100 are divisible by 5 and perform following operations.**

- **i. Length of vector x**
- **j. Print the values stored at 3rd, 7th, 10th & 15th**
- **k. Find the sum, mean ,median, standard deviation of vector x.**
- **l. Create another vector with name color & print values**
- **m. Access 7th and 8th location values in color vector**
- **n. Change the value of 5th location in color vector**
- **o. Repeat values in color exactly thrice**
- **p. Access multiple items in color simultaneously**

R Code:

```
#Mohit

#200010130070

divisible_by_5 <- seq(from = 5, to = 100, by = 5)

# Printing the length of the vector

print(length(divisible_by_5))


divisible_by_5 <- seq(from = 5, to = 100, by = 5)

# Printing the 10th, 20th and 30th element of the vector

print(divisible_by_5[c(10, 20, 30)])


divisible_by_5 <- seq(from = 5, to = 100, by = 5)

# Printing the sum, mean, median, and standard deviation of the vector

print(sum(divisible_by_5))

print(mean(divisible_by_5))

print(median(divisible_by_5))

print(sd(divisible_by_5))
```

```
colors <- c("red", "green", "blue", "yellow", "orange")
# Printing the color vector
print(colors)

# Creating a color vector
colors <- c("red", "green", "blue", "yellow", "orange")
# Creating another color vector
more_colors <- c("purple", "pink", "brown")
# Changing the 5th element of the color vector to "black"
colors[5] <- "black"
# Printing the modified color vector
print(colors)

colors <- c("red", "green", "blue", "yellow", "orange")
# Creating another color vector and repeating each element twice
doubled_colors <- rep(colors, each = 2)
# Printing the doubled color vector
print(doubled_colors)

colors <- c("red", "green", "blue", "yellow", "orange", "purple", "pink",
        "brown")
# Printing the 7th and 8th elements of the color vector
print(colors[7:8])
```

Output:

Console Terminal Background Jobs

R 4.3.0 · ~/

```
> #Mohit
> #200010130070
> divisible_by_5 <- seq(from = 5, to = 100, by = 5)
> # Printing the length of the vector
> print(length(divisible_by_5))
[1] 20
> divisible_by_5 <- seq(from = 5, to = 100, by = 5)
> # Printing the 10th, 20th and 30th element of the vector
> print(divisible_by_5[c(10, 20, 30)])
[1]  50 100  NA
> divisible_by_5 <- seq(from = 5, to = 100, by = 5)
> # Printing the sum, mean, median, and standard deviation of the vector
> print(sum(divisible_by_5))
[1] 1050
> print(mean(divisible_by_5))
[1] 52.5
> print(median(divisible_by_5))
[1] 52.5
> print(sd(divisible_by_5))
[1] 29.5804
> colors <- c("red", "green", "blue", "yellow", "orange")
> # Printing the color vector
> print(colors)
[1] "red"    "green"  "blue"   "yellow" "orange"
```

Console Terminal Background Jobs

R 4.3.0 · ~/

```
[1] "red"    "green"  "blue"   "yellow" "orange"
> # Creating a color vector
> colors <- c("red", "green", "blue", "yellow", "orange")
> # Creating another color vector
> more_colors <- c("purple", "pink", "brown")
> # Changing the 5th element of the color vector to "black"
> colors[5] <- "black"
> # Printing the modified color vector
> print(colors)
[1] "red"    "green"  "blue"   "yellow" "black"
> colors <- c("red", "green", "blue", "yellow", "orange")
> # Creating another color vector and repeating each element twice
> doubled_colors <- rep(colors, each = 2)
> # Printing the doubled color vector
> print(doubled_colors)
 [1] "red"    "red"    "green"  "green"  "blue"   "blue"   "yellow" "yellow"
 [9] "orange" "orange"
> colors <- c("red", "green", "blue", "yellow", "orange", "purple", "pink",
+             "brown")
> # Printing the 7th and 8th elements of the color vector
> print(colors[7:8])
[1] "pink"  "brown"
>
```

PROGRAM

5. **Create a list of students names and perform the following operations:**

h. **Access the elements of the list**
i. **Change the item values at 2nd and 3rd location**
j. **Find out the length of the list**
k. **Check if an item exists in the list or not**
l. **Add new student names in the list**
m. **Access the elements of the list through loop**
n. **Make another list of student names and merge it with the original list**

R Code:

```
#Mohit

#200010130070

student<-list("Mohit","Yash","Rohan","Aman","Rahul","Hemant")

student[2]<-"Nikhil"

student[3]<-"Rohit"

student[3]

length(student) #finding length of list

"gaurav" %in% student

append(student,"ashish",after=4) #append list after location 4

for (i in student){ print(i)}

student2<-list("Rachit","Sachin","Raman","Deepak","Navdeep","Sahil")

c(student,student2) #merge two list in a single list
```

Output:

```
Console   Terminal ×   Background Jobs ×
R 4.3.0 · ~/
> #Mohit
> #200010130070
> student<-list("Mohit","Yash","Rohan","Aman","Rahul","Hemant")
> student[2]<-"Nikhil"
> student[3]<-"Rohit"
> student[3]
[[1]]
[1] "Rohit"

> length(student) #finding length of list
[1] 6
> "gaurav" %in% student
[1] FALSE
> append(student,"ashish",after=4) #append list after location 4
[[1]]
[1] "Mohit"

[[2]]
[1] "Nikhil"

[[3]]
[1] "Rohit"

[[4]]
[1] "Aman"

[[5]]
[1] "ashish"

[[6]]
[1] "Rahul"

[[7]]
[1] "Hemant"
```

```
[1] "Mohit"
[1] "Nikhil"
[1] "Rohit"
[1] "Aman"
[1] "Rahul"
[1] "Hemant"
> student2<-list("Rachit","Sachin","Raman","Deepak","Navdeep","Sahil")
> c(student,student2) #merge two list in a single list
[[1]]
[1] "Mohit"

[[2]]
[1] "Nikhil"

[[3]]
[1] "Rohit"

[[4]]
[1] "Aman"

[[5]]
[1] "Rahul"

[[6]]
[1] "Hemant"

[[7]]
[1] "Rachit"

[[8]]
[1] "Sachin"

[[9]]
[1] "Raman"
```

PROGRAM

6. **Use plot(iris) function and interpret the output. Write down your findings about the dataset.**

IRIS: iris is a data.frame, which is probably the most commonly used data structure in R. It is basically a table where each column is a variable and each row has one set of values for each of those variables (much like a single sheet in a program like LibreOffice Calc or Microsoft Excel). In the iris data, there are five columns: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species. Each row corresponds to the measurements for an individual flower. Note that all the values in a column of a data.frame must be of the same type - if you try to mix numbers and words in the same column, R will “coerce” the data to a single type, which may cause problems for downstream analyses.

An investigation of our call to the head command illustrates two fundamental concepts in R: variables and functions.

head(x = iris)

iris is a variable. That is, it is a name we use to refer to some information in our computer’s memory. In this case, the information is a table of flower measurements.

head is the name of the function that prints out the first six rows of a data.frame. Most functions require some form of input; in this example, we provided one piece of input to head: the name of the variable for which we want the first six lines.

Iris Dataset is considered as the Hello World for data science. It contains five columns namely - Petal Length, Petal Width, Sepal Length, Sepal Width. And Species Type. Iris is a flowering plant, the researchers have measured various features of the different iris flowers and recorded them digitally.

The iris dataset is a built-in dataset in R that contains measurements on 4 different attributes (in centimeters) for 50 flowers from 3 different species. Using the plot(iris) function in R, we can generate a scatterplot matrix of the iris dataset. The plot shows the pairwise relationships between the four attributes, with each point representing an individual flower.

From the scatterplot matrix, we can observe that the petal length and petal width are highly correlated with each other, while sepal length and sepal width are less strongly correlated.

The 3 species are:

• Species Setosa has smaller sepal lengths but larger sepal widths.

• Versicolor Species lies in the middle of the other two species in terms of sepal length & width

• Species Virginica has larger sepal lengths but smaller sepal widths.

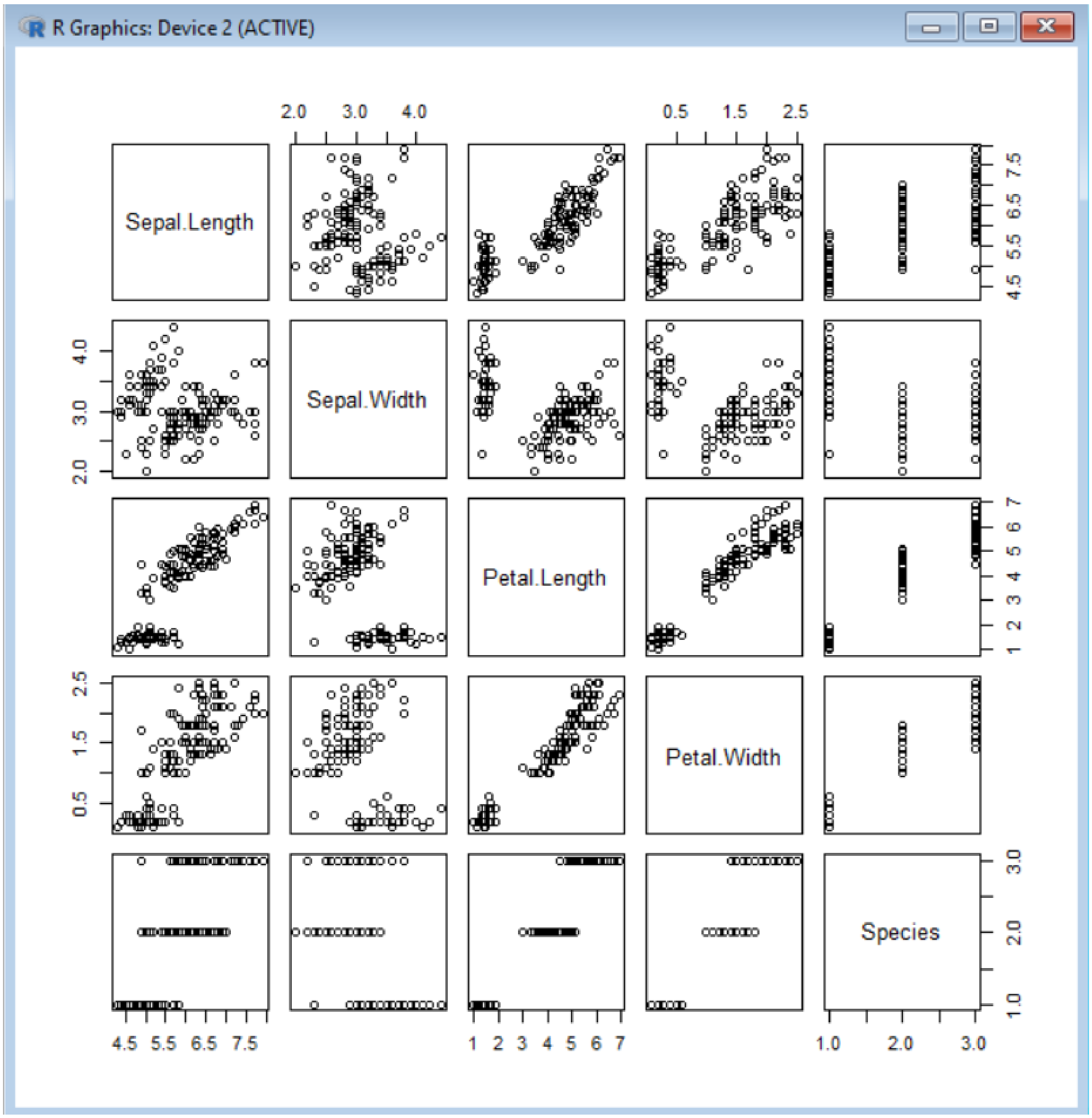
We can use the dim() function to get the dimensions of the dataset in terms of number of rows and number of columns. We find that the dataset has 150 rows and 5 columns.

Overall, the iris dataset is a useful and informative dataset that can be used to explore different data analysis and machine learning techniques. It provides a good example of a dataset with multiple attributes and multiple classes, making it suitable for classification problems.

R Code:

plot(iris)

Output:



PROGRAM

7. **Install and load mass package and access the Boston dataset. Study the dataset from the resources available on the internet and write what you find relevant on the dataset.**

R Code:

```
#Mohit Kumar
#200010130070

t=library(MASS)
t
Boston
```

Output:

```
> t=library(MASS)
Warning message:
package 'MASS' was built under R version 4.2.3
> t
[1] "MASS"      "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
> Boston
      crim   zn indus chas    nox    rm   age    dis rad tax ptratio  black lstat medv
1  0.00632 18.0  2.31    0 0.5380 6.575  65.2 4.0900   1 296    15.3 396.90  4.98 24.0
2  0.02731  0.0  7.07    0 0.4690 6.421  78.9 4.9671   2 242    17.8 396.90  9.14 21.6
3  0.02729  0.0  7.07    0 0.4690 7.185  61.1 4.9671   2 242    17.8 392.83  4.03 34.7
4  0.03237  0.0  2.18    0 0.4580 6.998  45.8 6.0622   3 222    18.7 394.63  2.94 33.4
5  0.06905  0.0  2.18    0 0.4580 7.147  54.2 6.0622   3 222    18.7 396.90  5.33 36.2
6  0.02985  0.0  2.18    0 0.4580 6.430  58.7 6.0622   3 222    18.7 394.12  5.21 28.7
7  0.08829 12.5  7.87    0 0.5240 6.012  66.6 5.5605   5 311    15.2 395.60 12.43 22.9
8  0.14455 12.5  7.87    0 0.5240 6.172  96.1 5.9505   5 311    15.2 396.90 19.15 27.1
9  0.21124 12.5  7.87    0 0.5240 5.631 100.0 6.0821   5 311    15.2 386.63 29.93 16.5
10 0.17004 12.5  7.87    0 0.5240 6.004  85.9 6.5921   5 311    15.2 386.71 17.10 18.9
11 0.22489 12.5  7.87    0 0.5240 6.377  94.3 6.3467   5 311    15.2 392.52 20.45 15.0
12 0.11747 12.5  7.87    0 0.5240 6.009  82.9 6.2267   5 311    15.2 396.90 13.27 18.9
13 0.09378 12.5  7.87    0 0.5240 5.889  39.0 5.4509   5 311    15.2 390.50 15.71 21.7
14 0.62976  0.0  8.14    0 0.5380 5.949  61.8 4.7075   4 307    21.0 396.90  8.26 20.4
15 0.63796  0.0  8.14    0 0.5380 6.096  84.5 4.4619   4 307    21.0 380.02 10.26 18.2
16 0.62739  0.0  8.14    0 0.5380 5.834  56.5 4.4986   4 307    21.0 395.62  8.47 19.9
17 1.05393  0.0  8.14    0 0.5380 5.935  29.3 4.4986   4 307    21.0 386.85  6.58 23.1
18 0.78420  0.0  8.14    0 0.5380 5.990  81.7 4.2579   4 307    21.0 386.75 14.67 17.5
19 0.80271  0.0  8.14    0 0.5380 5.456  36.6 3.7965   4 307    21.0 288.99 11.69 20.2
20 0.72580  0.0  8.14    0 0.5380 5.727  69.5 3.7965   4 307    21.0 390.95 11.28 18.2
21 1.25179  0.0  8.14    0 0.5380 5.570  98.1 3.7979   4 307    21.0 376.57 21.02 13.6
22 0.85204  0.0  8.14    0 0.5380 5.965  89.2 4.0123   4 307    21.0 392.53 13.83 19.6
```

Relevance:

The **Boston** dataset contains information about various attributes for suburbs in Boston, Massachusetts.

The Boston data set is found in the MASS R package. One can load the Boston data set in R by issuing the following command at the console data("Boston"). This will load the data into a variable called Boston. If R says the Boston data set is not found, you can try installing the package by issuing this command install.packages("MASS") and then attempt to reload the data. The size of this file is about 34,727 bytes.

The Boston data frame has 506 rows and 14 columns.

This data frame contains the following columns:

- crim - per capita crime rate by town.
- zn - proportion of residential land zoned for lots over 25,000 sq.ft.
- indus - proportion of non-retail business acres per town.
- chas - Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
- nox - nitrogen oxides concentration (parts per 10 million).
- rm - average number of rooms per dwelling.
- age - proportion of owner-occupied units built prior to 1940.
- dis - weighted mean of distances to five Boston employment centres.
- rad - index of accessibility to radial highways.
- tax - full-value property-tax rate per \$10,000.
- ptratio - pupil-teacher ratio by town.
- black - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.
- lstat - lower status of the population (percent).
- medv - median value of owner-occupied homes in \$1000s.

PROGRAM

8. **Apply summary() command to iris dataset of the ‘datasets’ package and interpret the output.**

R Code:

#Mohit Kumar
#200010130070

summary(iris)

Output:

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
>
```

Data summaries usually present the dataset's average (mean, median, and/or mode); standard deviation from mean or interquartile range; how the data is distributed across the range of data (for example is it skewed to one side of the range); and statistical dependence.
For each of the numeric variables we can see the following information:

- **Min**: The minimum value.
- **1st Qu**: The value of the first quartile (25th percentile).
- **Median**: The median value.
- **Mean**: The mean value.
- **3rd Qu**: The value of the third quartile (75th percentile).
- **Max**: The maximum value.

For the only categorical variable in the dataset (Species) we see a frequency count of each value:

- **setosa**: This species occurs 50 times.
- **versicolor**: This species occurs 50 times.
- **virginica**: This species occurs 50 times.

PROGRAM

9. **Check and justify the outcome of the following expressions :**
   a. **sqrt(3)^2 == 3**
   b. **sqrt(4)^2 == 4**
   c. **near(sqrt(3)^2,3)**
   d. **near(sqrt(4)^2,7)**

R Code:

```
#Mohit Kumar
#200010130070

library(dplyr)
sqrt(5)^2==2
sqrt(7)^2==4
near(sqrt(3)^2,3)
near(sqrt(9)^2,5)
```

Output:

```
> library(dplyr)
> sqrt(5)^2==2
[1] FALSE
> sqrt(7)^2==4
[1] FALSE
> near(sqrt(3)^2,3)
[1] TRUE
> near(sqrt(9)^2,5)
[1] FALSE
>
```

<u>Note</u>: - Install **dplyr** package for data manipulation, to use near function.

PROGRAM

10. Install MASS Package and then use apply() to find the measure of the central tendency and dispersion.

R Code:

```
#Mohit Kumar
#200010130070

library(MASS)
data(state)
head(state.x77)

str(state.x77)
apply(state.x77,2,mean)
apply(state.x77,2,median)
apply(state.x77,2,sd)
```

Output:

```
> library(MASS)
> data(state)
> head(state.x77)
           Population Income Illiteracy Life Exp Murder HS Grad Frost   Area
Alabama          3615   3624        2.1    69.05   15.1    41.3    20  50708
Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
Arizona          2212   4530        1.8    70.55    7.8    58.1    15 113417
Arkansas         2110   3378        1.9    70.66   10.1    39.9    65  51945
California      21198   5114        1.1    71.71   10.3    62.6    20 156361
Colorado         2541   4884        0.7    72.06    6.8    63.9   166 103766
>
> str(state.x77)
 num [1:50, 1:8] 3615 365 2212 2110 21198 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas" ...
  ..$ : chr [1:8] "Population" "Income" "Illiteracy" "Life Exp" ...
> apply(state.x77,2,mean)
Population     Income Illiteracy   Life Exp     Murder    HS Grad      Frost       Area 
 4246.4200  4435.8000     1.1700    70.8786     7.3780    53.1080   104.4600 70735.8800 
> apply(state.x77,2,median)
Population     Income Illiteracy   Life Exp     Murder    HS Grad      Frost       Area 
  2838.500   4519.000      0.950     70.675      6.850     53.250    114.500  54277.000 
> apply(state.x77,2,sd)
  Population       Income   Illiteracy     Life Exp       Murder      HS Grad        Frost         Area 
4.464491e+03 6.144699e+02 6.095331e-01 1.342394e+00 3.691540e+00 8.076998e+00 5.198085e+01 8.532730e+04 
> |
```

PROGRAM

11. Create a function that gives mean and standard deviation, then save them as object.

R Code:

```
#Mohit Kumar
#200010130070

state.summary<-apply(state.x77,2,function(x) c(mean(x),sd(x)))
state.summary
```

Output:

```
> state.summary<-apply(state.x77,2,function(x) c(mean(x),sd(x)))
> state.summary
     Population    Income Illiteracy  Life Exp  Murder   HS Grad     Frost     Area
[1,]   4246.420 4435.8000  1.1700000 70.878600 7.37800 53.108000 104.46000 70735.88
[2,]   4464.491  614.4699  0.6095331  1.342394 3.69154  8.076998  51.98085 85327.30
> |
```

PROGRAM

12. Create a function that gives min, median, and max, then save them as object.

R Code:

```
#Mohit Kumar
#200010130070

state.range<-apply(state.x77,2,function(x)c(min(x),median(x),max(x)))
state.range
```

Output:

```
> state.range<-apply(state.x77,2,function(x)c(min(x),median(x),max(x)))
> state.range
     Population Income Illiteracy Life Exp Murder HS Grad Frost   Area
[1,]      365.0   3098       0.50   67.960   1.40   37.80   0.0   1049
[2,]     2838.5   4519       0.95   70.675   6.85   53.25 114.5  54277
[3,]    21198.0   6315       2.80   73.600  15.10   67.30 188.0 566432
>
```

PROGRAM

13. Write a program to find the population density for each state using mapply().

R Code:

```
#Mohit Kumar
#200010130070

population <-state.x77 [1:50]
area<-state.area
pop.dens<-mapply(function(x,y)x/y,population,area)
pop.dens
```

Output:

```
> population <-state.x77 [1:50]
> area<-state.area
> pop.dens<-mapply(function(x,y)x/y,population,area)
> pop.dens
 [1] 0.070045922 0.000618899 0.019419010 0.039733353 0.133578671 0.024374802 0.618886005 0.281477880 0.141342213
[10] 0.083752293 0.134573643 0.009729885 0.198528369 0.146399934 0.050826079 0.027715647 0.083847011 0.078437030
[19] 0.031853078 0.389713529 0.704129829 0.156503367 0.046640815 0.049061112 0.068406854 0.005070070 0.019993008
[28] 0.005337434 0.087274291 0.935809086 0.009402791 0.364611909 0.103468604 0.009014364 0.260419194 0.038830647
[37] 0.023551005 0.261619571 0.766886326 0.090677830 0.008838761 0.098783259 0.045773344 0.014166941 0.049120616
[46] 0.122038466 0.052190873 0.074397254 0.081721694 0.003840105
> 
```

PROGRAM

14. Write a program using tapply() to explore population by region.

R Code:

```
#Mohit Kumar
#200010130070

region.info<-tapply(population,state.region,function(x)c(min(x),median(x),max(x)))
region.info
```

Output:

```
> region.info<-tapply(population,state.region,function(x)c(min(x),median(x),max(x)))
> region.info
$Northeast
[1]   472  3100 18076

$South
[1]   579.0  3710.5 12237.0

$`North Central`
[1]   637  4255 11197

$West
[1]   365  1144 21198

> |
```

PROGRAM

15. **Write a script file to compute the following of the numeric variable in Boston dataset.**
    a. **Sum**
    b. **Range**
    c. **Mean**
    d. **Standard Deviation**

R Code:

```
#Mohit Kumar
#200010130070

sapply(Boston,sum)
sapply(Boston,range)
sapply(Boston,mean)
sapply(Boston,sd)
```

Output:

```
> sapply(Boston,sum)
       crim          zn       indus        chas         nox          rm         age         dis         rad         tax 
  1828.4429   5750.0000   5635.2100     35.0000    280.6757   3180.0250  34698.9000   1920.2916   4832.0000 206568.0000 
    ptratio       black       lstat        medv 
  9338.5000 180477.0600   6402.4500  11401.6000 
> sapply(Boston,range)
         crim  zn indus chas   nox    rm   age     dis rad tax ptratio  black lstat medv
[1,]  0.00632   0  0.46    0 0.385 3.561   2.9  1.1296   1 187    12.6   0.32  1.73    5
[2,] 88.97620 100 27.74    1 0.871 8.780 100.0 12.1265  24 711    22.0 396.90 37.97   50
> sapply(Boston,mean)
        crim           zn        indus         chas          nox           rm          age          dis          rad 
  3.61352356  11.36363636  11.13677866   0.06916996   0.55469506   6.28463439  68.57490119   3.79504269   9.54940711 
         tax      ptratio        black        lstat         medv 
408.23715415  18.45553360 356.67403162  12.65306324  22.53280632 
> sapply(Boston,sd)
       crim          zn       indus        chas         nox          rm         age         dis         rad         tax 
  8.6015451  23.3224530   6.8603529   0.2539940   0.1158777   0.7026171  28.1488614   2.1057101   8.7072594 168.5371161 
    ptratio       black       lstat        medv 
  2.1649455  91.2948644   7.1410615   9.1971041 
> 
```

PROGRAM

16. Assuming the character vector students, having 10 names of students

  c. Find the character count in each name.

  d. Find if the names “Akaash” and “Ankit” exist in the vector.

R Code:

```
#Mohit Kumar
#200010130070

#a. Finding Character count
students<-
c("Swastik","Hitesh","Deepak","Akshay","Ravi","Vipul","Vishal","Jai","Pulkit","Ankit")
nchar(students)

#b Checking if given names exist
"Akaash" %in% students
"Hitesh" %in% students
```

Output:

```
>
> #a. Finding Character count
> students<-c("Swastik","Hitesh","Deepak","Akshay","Ravi","Vipul","Vishal","Jai","Pulkit","Ankit")
> nchar(students)
 [1] 7 6 6 6 4 5 6 3 6 5
>
> #b Checking if given names exist
> "Akaash" %in% students
[1] FALSE
> "Hitesh" %in% students
[1] TRUE
>
```

PROGRAM

17. Write a program to output the indices of the names that contain substring “ii” in vector students of assignment – 16.

R Code:

```
#Mohit Kumar
#200010130070

students<-
c("Hitesh","Deepak","Akshay","Ravi","Swastik","Vipul","Vishal","Jai","Pulkit","Ankit")
for(i in 1:length(students))
{
  if(grepl("a",students[i])==TRUE)
    print(i)
}
```

Output:

```
>
> students<-c("Hitesh","Deepak","Akshay","Ravi","Swastik","Vipul","Vishal","Jai","Pulkit","Ankit")
> for(i in 1:length(students))
+ {
+   if(grepl("a",students[i])==TRUE)
+     print(i)
+ }
[1] 2
[1] 3
[1] 4
[1] 5
[1] 7
[1] 8
>
```

PROGRAM

18. Find out how many string end with letters “sh” in vector students of assignment-16.

R Code:

```
#Mohit Kumar
#200010130070

students<-
c("Hitesh","Deepi","Akshay","Ravi","Swastik","Vipul","Vishal","Jai","Pulkit","Ankit")
count<-0;
for(i in 1:length(students))
{
  if(endsWith(students[i],"it")==TRUE)
    count<-count+1;
}
count
```

Output:

```
> students<-c("Hitesh","Deepi","Akshay","Ravi","Swastik","Vipul","Vishal","Jai","Pulkit","Ankit")
> count<-0;
> for(i in 1:length(students))
+ {
+   if(endsWith(students[i],"it")==TRUE)
+     count<-count+1;
+ }
> count
[1] 2
> |
```

PROGRAM

19. **Create a vector of factor type data for the hair colors of 10 people, where value of hair colors are :- Black , Blonde , Dark Brown , Grey.**
    c. **Display the levels of factor data**
    d. **Find the maximum value in vector of hair colors using table() function.**

R Code:

```
#Mohit Kumar
#200010130070

haircolor<-factor(c("Black","DarkBrown","Black","Blonde","Grey","Blonde","Grey","Black",
"Dark Brown","Black"))
haircolor
levels(haircolor)
table(haircolor)
max(table(haircolor))
```

Output:

```
>
> haircolor<-factor(c("Black","DarkBrown","Black","Blonde","Grey","Blonde","Grey","Black", "Dark Brown","Black"))
> haircolor
 [1] Black      DarkBrown  Black      Blonde     Grey       Blonde     Grey       Black      Dark Brown Black
Levels: Black Blonde Dark Brown DarkBrown Grey
> levels(haircolor)
[1] "Black"      "Blonde"     "Dark Brown" "DarkBrown"  "Grey"
> table(haircolor)
haircolor
     Black     Blonde Dark Brown  DarkBrown       Grey
         4          2          1          1          2
> max(table(haircolor))
[1] 4
>
```

PROGRAM

20. Create an empty vector of data factor type for the names of first 6 months in a year. Remember to keep the levels of the data in order of months from January to June .

R Code:

```
#Mohit Kumar
#200010130070

months<-factor(c(),levels=c("Jan","Feb","March","April","May","June","July","August","September"),ordered=TRUE)
months
```

Output:

```
>
> months<-factor(c(),levels=c("Jan","Feb","March","April","May","June","July","August","September"),ordered=TRUE)
> months
ordered(0)
Levels: Jan < Feb < March < April < May < June < July < August < September
>
```

PROGRAM

21.Perform the following operations:

- **a. A+B**
- **b. A-B**
- **c. 3*A**
- **d. Ax=B. x=?**
- **e. diag(c(4,1,2,3), nrow=4)**
- **f. t(A)**
- **g. eigen(A)**
- **h. eigen(A*t(A))**
- **i. A%*%B**
- **j. b=c(7,4) . A*b**

R Code:

```
#Mohit Kumar
#200010130070

A<-matrix(c(2,0,1,3),ncol=2)
B<-matrix(c(5,2,4,-1),ncol=2)

A+B
A-B
3*A
diag(c(4,1,2,3), nrow=4)
Ax=B
x
t(A)
eigen(A)
eigen(A*t(A))
A%*%B
b=c(7,4)
A*b
```

Output:

```
>
> A<-matrix(c(2,0,1,3),ncol=2)
> B<-matrix(c(5,2,4,-1),ncol=2)
>
> A+B
     [,1] [,2]
[1,]    7    5
[2,]    2    2
> A-B
     [,1] [,2]
[1,]   -3   -3
[2,]   -2    4
> 3*A
     [,1] [,2]
[1,]    6    3
[2,]    0    9
> diag(c(4,1,2,3), nrow=4)
     [,1] [,2] [,3] [,4]
[1,]    4    0    0    0
[2,]    0    1    0    0
[3,]    0    0    2    0
[4,]    0    0    0    3
> Ax=B
> x
[1] "Kartik"
> t(A)
     [,1] [,2]
[1,]    2    0
[2,]    1    3
> eigen(A)
eigen() decomposition
$values
[1] 3 2

$vectors
          [,1] [,2]
[1,] 0.7071068    1
[2,] 0.7071068    0

> eigen(A*t(A))
eigen() decomposition
$values
[1] 9 4

$vectors
     [,1] [,2]
[1,]    0   -1
[2,]    1    0

> A%*%B
     [,1] [,2]
[1,]   12    7
[2,]    6   -3
> b=c(7,4)
> A*b
     [,1] [,2]
[1,]   14    7
[2,]    0   12
>
```

PROGRAM

23. Create a vector to store the grades of 25 students for first minor grade are given as {A, B, C, D}. Compute the modal grade. Further, store the grades of the same students for second minor exam. Compare the grades for 2 exams. Count number of student who have got higher grade.

R Code:

```
#Mohit Kumar
#200010130070

Major1 <-factor(c('A','B','D','C','A','A','B','D','C','A','A','B','D','C','A','A','B','D',
              'C','A','D','C','A','B','B'),levels=c('A','B','C','D') ,ordered = TRUE)
Major1
table(Major1)
Major2<-factor(c('B','A','D','C','A','B','A','D','C','A','B','A','D','C','A','B','A','D','C','A','B',
              'A','D','C','A'),levels=c('A','B','C','D'),ordered = TRUE)
Major2
table(Major2)
Major1==Major2
table(Major1==Major2)
sum(Major1>Major2)
which.max(table(Major1))
which.min(table(Major2))
```

Output:

```
>
> Major1 <-factor(c('A','B','D','C','A','A','B','D','C','A','A','B','D','C','A','A','B','D',
+                   'C','A','D','C','A','B','B'),levels=c('A','B','C','D') ,ordered = TRUE)
> Major1
 [1] A B D C A A B D C A A B D C A A B D C A D C A B B
Levels: A < B < C < D
> table(Major1)
Major1
A B C D
9 6 5 5
> Major2<-factor(c('B','A','D','C','A','B','A','D','C','A','B','A','D','C','A','B','A','D','C','A','B',
+                  'A','D','C','A'),levels=c('A','B','C','D'),ordered = TRUE)
> Major2
 [1] B A D C A B A D C A B A D C A B A D C A B A D C A
Levels: A < B < C < D
> table(Major2)
Major2
 A  B  C  D
10  5  5  5
> Major1==Major2
 [1] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
[20]  TRUE FALSE FALSE FALSE FALSE FALSE
> table(Major1==Major2)

FALSE  TRUE
   13    12
> sum(Major1>Major2)
[1] 7
> which.max(table(Major1))
A
1
> which.min(table(Major2))
B
2 .
```

PROGRAM

24. Create a 4*3 matrix A of uniformly distributed random integer numbers between 1 to 100. Create another 3*4 matrix B with uniformly distributed random integer numbers between 1 to 10 .Perform matrix multiplication of the two matrices and store the result in third matrix C.

R Code:

```
#Mohit Kumar
#200010130070

a<-matrix (runif(12,1,100),nrow=4,ncol=3)
a
b<-matrix(runif(12,1,10),nrow=3,ncol=4)
b
c<-a%*%b
c
```

Output:

```
>
> a<-matrix (runif(12,1,100),nrow=4,ncol=3)
> a
         [,1]      [,2]     [,3]
[1,] 64.84463 82.535682 79.71824
[2,] 69.28946 25.205449 85.47155
[3,] 95.47445  7.221324 88.05637
[4,] 71.26990 85.166679 42.46793
> b<-matrix(runif(12,1,10),nrow=3,ncol=4)
> b
         [,1]     [,2]     [,3]     [,4]
[1,] 4.481783 1.422638 3.334459 6.823840
[2,] 8.657629 7.351927 2.696080 5.136798
[3,] 4.786385 6.360988 5.226688 6.880813
> c<-a%*%b
> c
          [,1]      [,2]     [,3]     [,4]
[1,] 1386.7451 1206.1335 855.4069 1414.985
[2,]  937.8595  827.5659 745.7319 1190.409
[3,]  911.8870  749.0417 798.0681 1294.496
[4,] 1260.0256  997.6684 689.2293 1216.032
>
```

Note: The runif() function generates random deviates of the uniform distribution.

PROGRAM

25. Create A and B, two 4*3 matrices of normally distributed random numbers, with mean 0 and standard deviation 1. Find the indices of all those numbers in matrix A which are less than the respective numbers in matrix B and print these numbers.

R Code:

```
#Mohit Kumar
#200010130070

a=matrix(rnorm(12,mean=0,sd=1),4,3)
a
b=matrix(rnorm(12,mean=0,sd=1),4,3)
b
nj=a<b
nj
asd=which(nj,arr.ind=TRUE)
asd
a[nj]
```

Output:

```
>
> a=matrix(rnorm(12,mean=0,sd=1),4,3)
> a
           [,1]       [,2]       [,3]
[1,]  2.4682172  0.9596861 -0.5222798
[2,]  1.2720768 -0.7195945  0.1136506
[3,] -0.5689821  0.1671057 -1.1104238
[4,]  0.1037349 -0.5428166 -0.3983680
> b=matrix(rnorm(12,mean=0,sd=1),4,3)
> b
           [,1]        [,2]       [,3]
[1,]  0.2998691 -0.04589419  0.2989089
[2,] -1.6649794 -0.21445571 -0.9870229
[3,]  0.9113570 -0.62501069 -0.1748194
[4,] -1.9929465  0.41881524  0.1241259
> nj=a<b
> nj
      [,1]  [,2]  [,3]
[1,] FALSE FALSE  TRUE
[2,] FALSE  TRUE FALSE
[3,]  TRUE FALSE  TRUE
[4,] FALSE  TRUE  TRUE
> asd=which(nj,arr.ind=TRUE)
> asd
     row col
[1,]   3   1
[2,]   2   2
[3,]   4   2
[4,]   1   3
[5,]   3   3
[6,]   4   3
> a[nj]
[1] -0.5689821 -0.7195945 -0.5428166 -0.5222798 -1.1104238 -0.3983680
> |
```

PROGRAM

26. Plotting pressure dataset in different forms:

(a) Histogram
(b) Boxplot

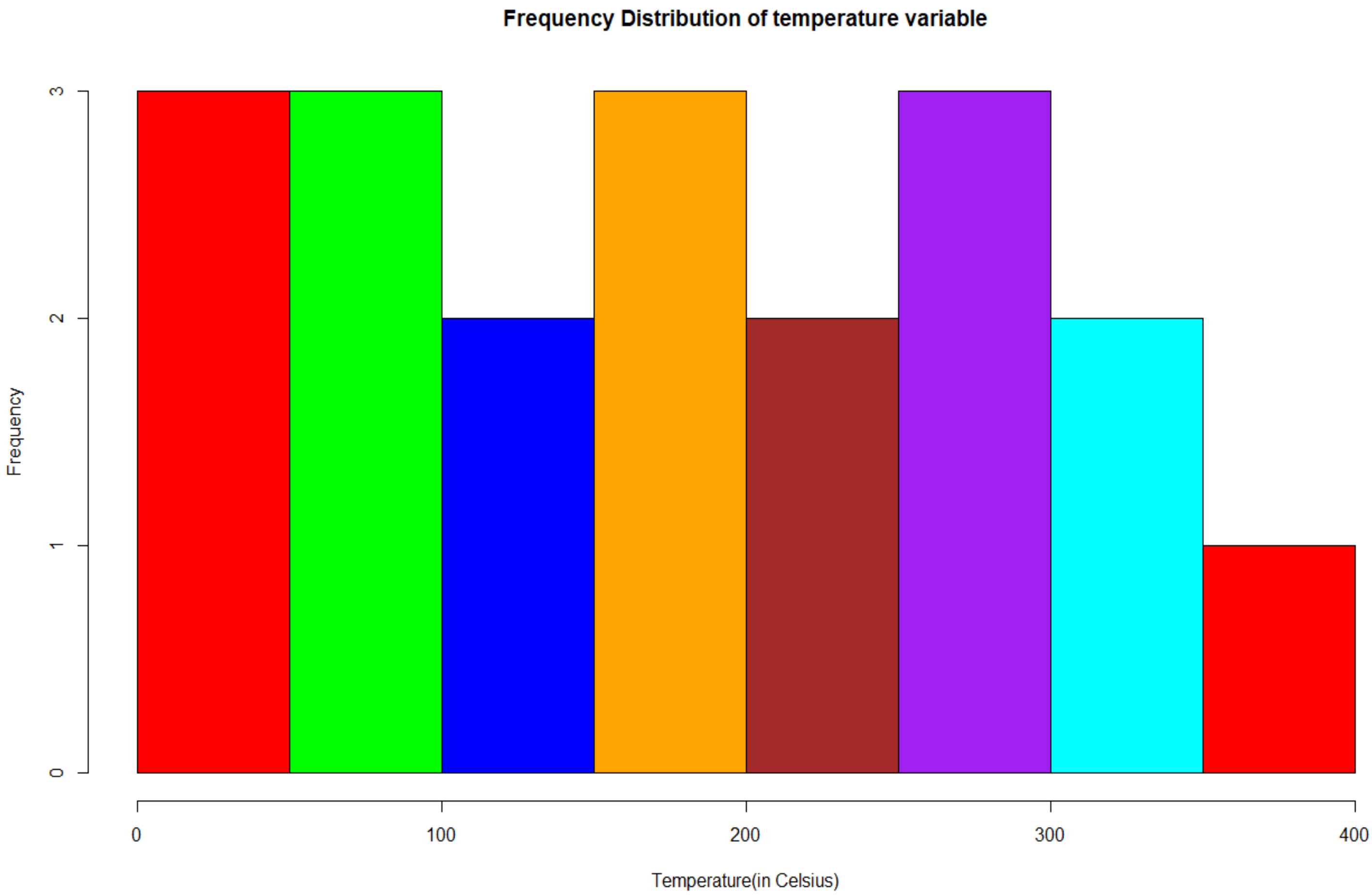
(a) Histogram

R Code:

```
#Mohit Kumar
#200010130070

hist(pressure$temperature,main="Frequency Distribution of temperature variable",
    xlab="Temperature(in Celsius)",ylab="Frequency",border="black",
    col=c("red","green","blue","orange","brown","purple","cyan"))
box("figure")
```

Output:

```
>
> hist(pressure$temperature,main="Frequency Distribution of temperature variable",
+       xlab="Temperature(in Celsius)",ylab="Frequency",border="black",
+       col=c("red","green","blue","orange","brown","purple","cyan"))
> box("figure")
>
```


Frequency Distribution of temperature variable
Frequency
0
1
2
3
0
100
200
300
400
Temperature(in Celsius)

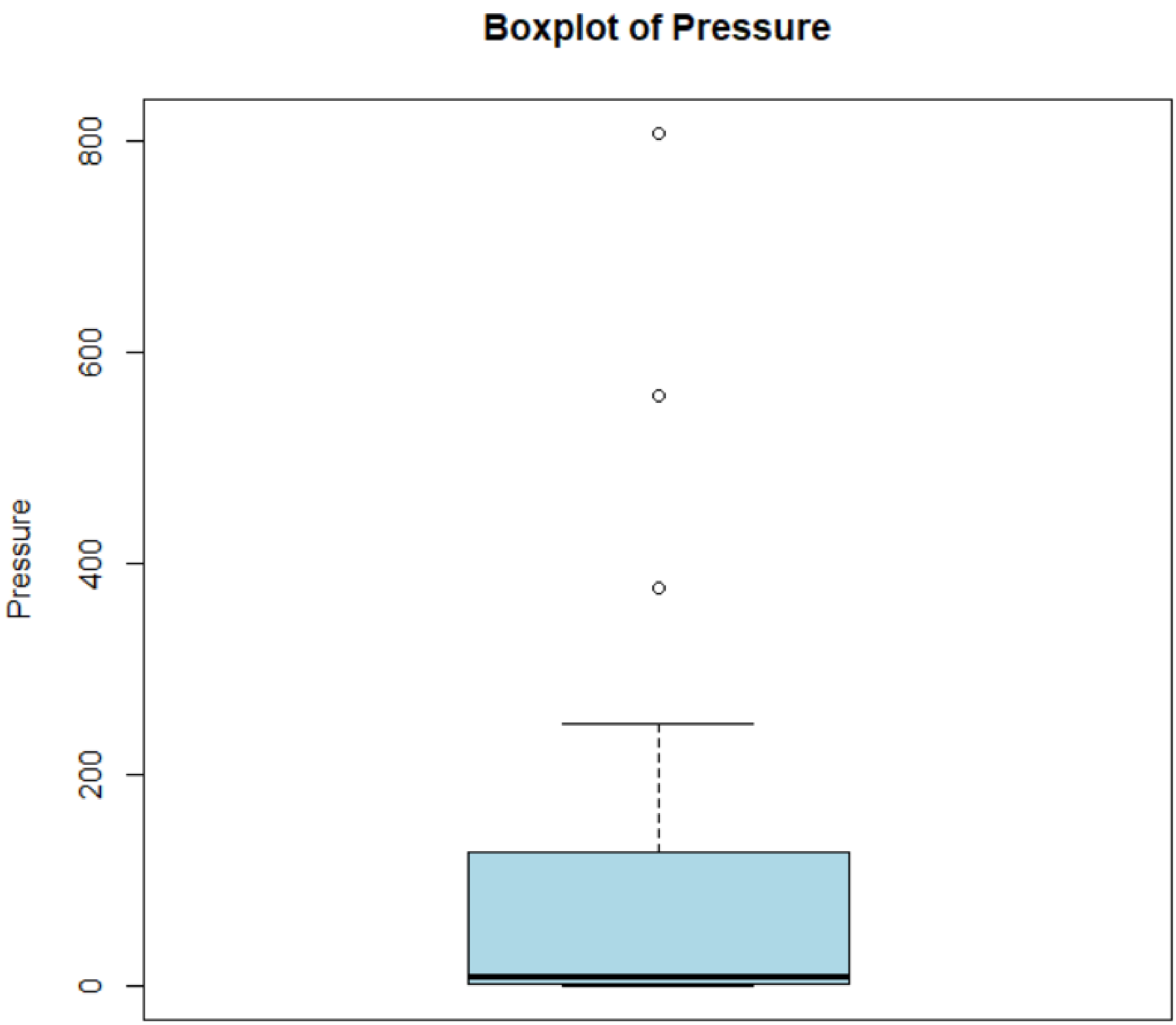
(b) **<u>Boxplot</u>**

R Code:

```
#Mohit Kumar
#200010130070
boxplot(pressure$pressure, main = "Boxplot of Pressure",
        ylab = "Pressure", col = "lightblue")
```

Output:

```
>
> boxplot(pressure$pressure, main = "Boxplot of Pressure",
+         ylab = "Pressure", col = "lightblue")
> 
```


Boxplot of Pressure
Pressure
0
200
400
600
800

PROGRAM

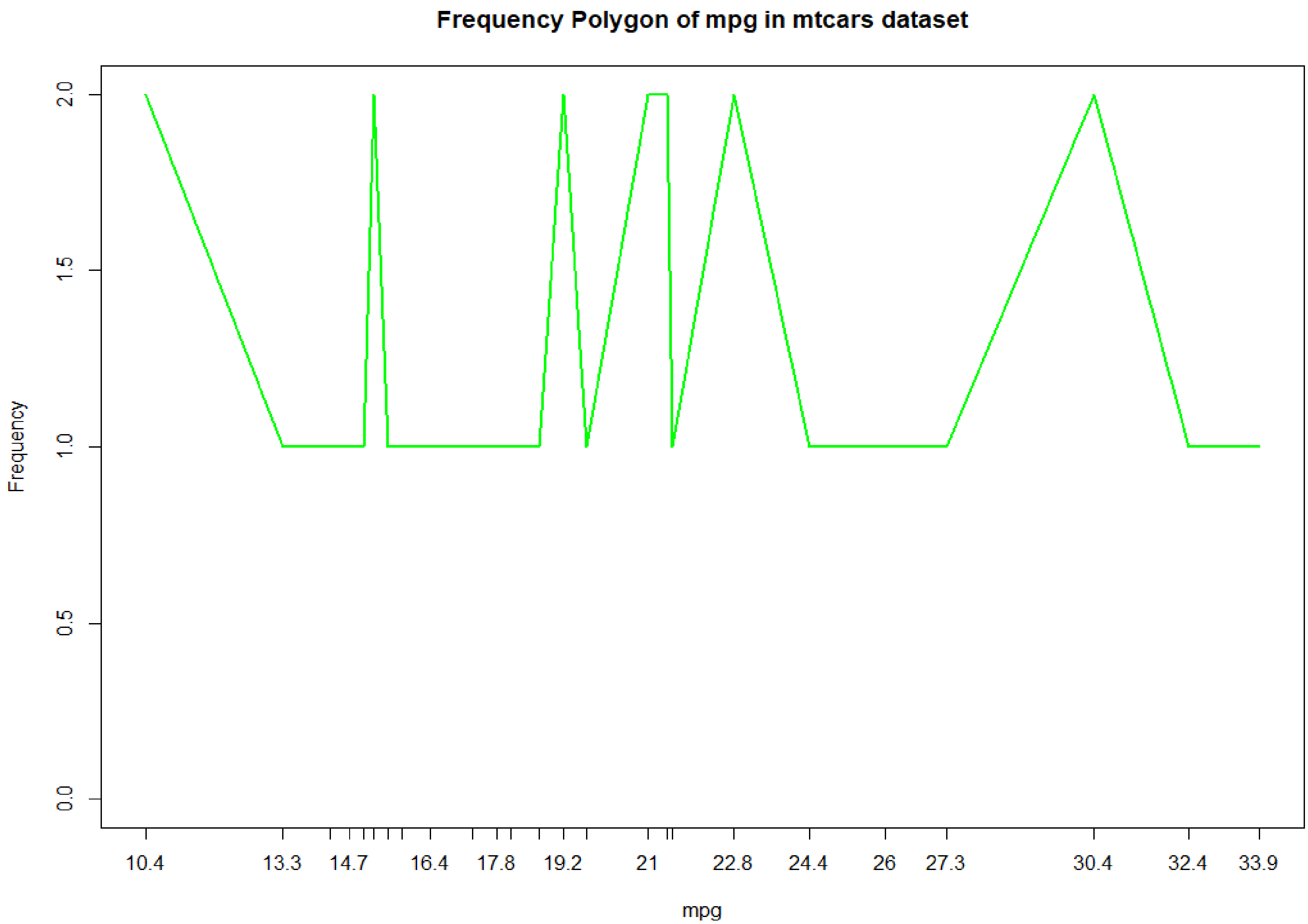
27. Plotting mtcars dataset in Frequency Polygon.

R Code:

```
#Mohit Kumar
#200010130070

data(mtcars)

mpg_freq <- table(mtcars$mpg)

plot(mpg_freq, type = "l", col = "green",
    main = "Frequency Polygon of mpg in mtcars dataset",
    xlab = "mpg", ylab = "Frequency")
```

Output:

```
>
> data(mtcars)
>
> mpg_freq <- table(mtcars$mpg)
>
> plot(mpg_freq, type = "l", col = "green",
+      main = "Frequency Polygon of mpg in mtcars dataset",
+      xlab = "mpg", ylab = "Frequency")
> |
```


Frequency Polygon of mpg in mtcars dataset
Frequency
0.0
0.5
1.0
1.5
2.0
10.4
13.3
14.7
16.4
17.8
19.2
21
22.8
24.4
26
27.3
30.4
32.4
33.9
mpg

PROGRAM

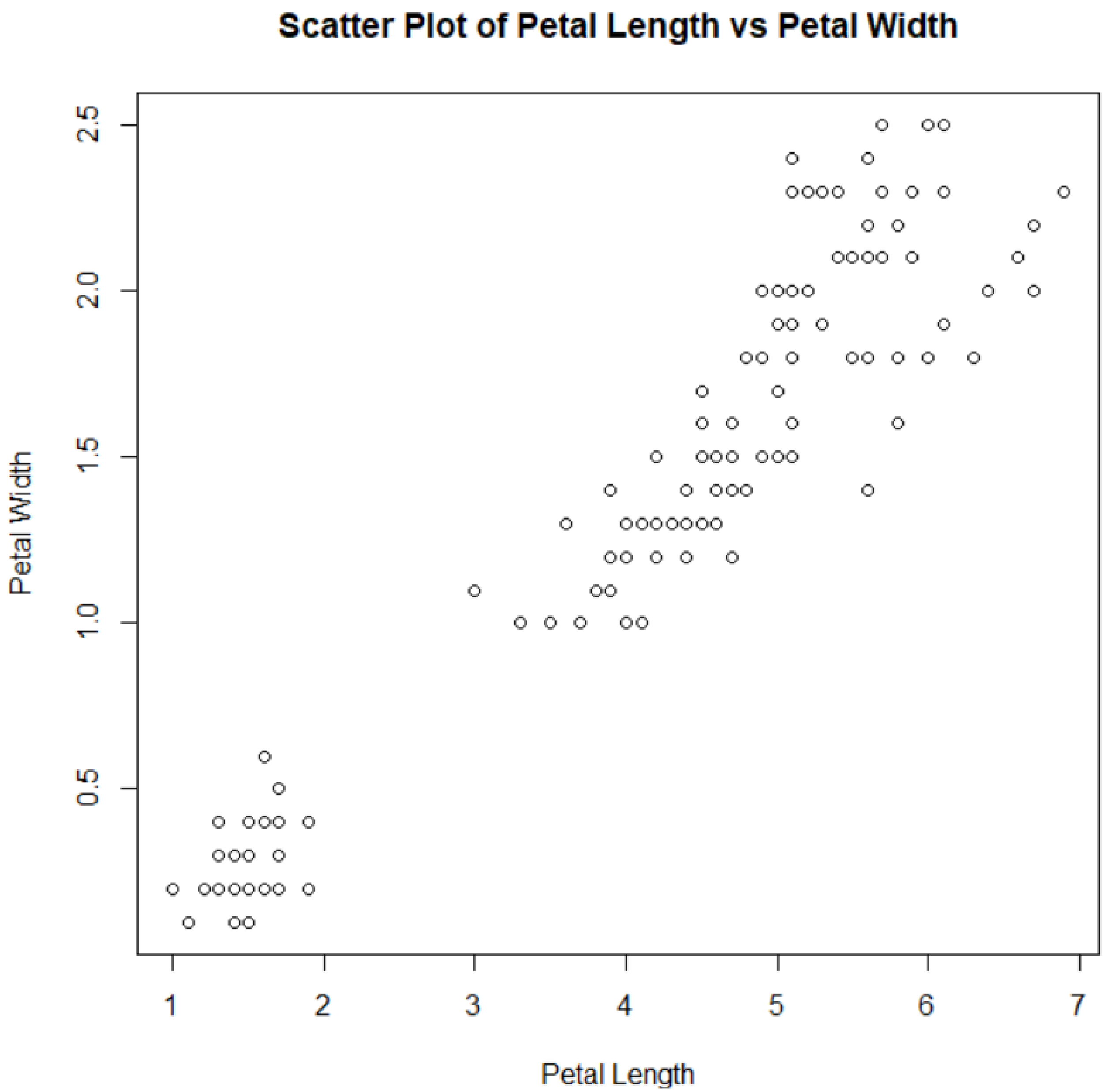
28. Plotting scatter plots from iris dataset with title and labels

R Code:

```
#Mohit Kumar
#200010130070

data(iris)
plot(iris$Sepal.Length, iris$Sepal.Width,
    main = "Scatter Plot of Sepal Length vs Sepal Width",
    xlab = "Sepal Length", ylab = "Sepal Width")

plot(iris$Petal.Length, iris$Petal.Width,
    main = "Scatter Plot of Petal Length vs Petal Width",
    xlab = "Petal Length", ylab = "Petal Width")
```

Output:

```
>
> data(iris)
> plot(iris$Sepal.Length, iris$Sepal.Width,
+      main = "Scatter Plot of Sepal Length vs Sepal Width",
+      xlab = "Sepal Length", ylab = "Sepal Width")
>
> plot(iris$Petal.Length, iris$Petal.Width,
+      main = "Scatter Plot of Petal Length vs Petal Width",
+      xlab = "Petal Length", ylab = "Petal Width")
>
```



Scatter Plot of Petal Length vs Petal Width
Petal Width
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7
Petal Length