**Program 1**

**Write a R program to create a Dataframe which contains details of 5 employees and display the details**

**Code:**

Employees<-data.frame(Name=c("Ujjawal","Satyajeet","Arman","Mayank","Arnav"),

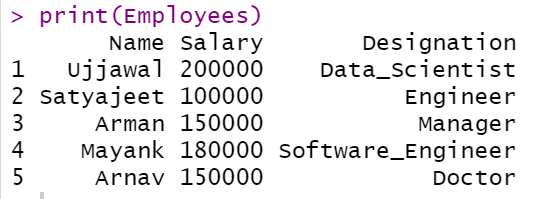
Salary=c(200000,100000,150000,180000,150000),

Designation=c("Data\_Scientist","Engineer","Manager","Software\_Engineer","Doctor"))

print("Details of the Employees")

print(Employees)

**Output:**



**Program 2**

**Write a program to print prime number up to a given number.**

**Code:**

prime\_numbers <- function(n) {

if (n >= 2) {

x = seq(2, n)

prime\_nums = c()

for (i in seq(2, n)) {

if (any(x == i)) {

prime\_nums = c(prime\_nums, i)

x = c(x[(x %% i) != 0], i)

}

}

return(prime\_nums)

}

else

{

stop("Input number should be at least 2.")

}

}

prime\_numbers(12)

**Output:**



**Program 3**

**Write a R program to print the first 10 Fibonacci series.**

**Code:**

Fibo <-numeric(10)

Fibo[1]<-0

Fibo[2]<-1

for (i in 3:11)

Fibo [i]<-Fibo[i-2]+fibo[i-1]

print(“First 10 fibonacci series are”)

print(Fibo)

**Output:**



**Program 4**

**Write a R program to find the maximum and the minimum value of a given vector.**

**Code:**

number = c(10,20,30,40,50,60)

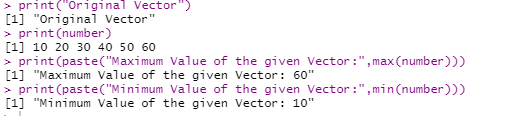
print("Original Vector")

print(number)

print(paste("Maximum Value of the given Vector:",max(number)))

print(paste("Minimum Value of the given Vector:",min(number)))

**Output**:



**Program 5**

**Create an array, passing in a vector of values and a vector of dimensions, also provide names for each dimension**

**Code:**

a <- array(6:30,

dim = c(4,3,2),

dimNames <- list(

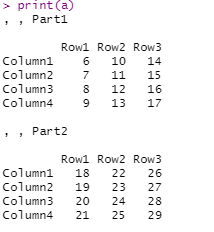
c("Column1","Column2","Column3","Column4"),

c("Row1","Row2","Row3"),

c("Part1","Part2")))

print(a)

**Output:**



**Program 6**

**Write a R program to create a list containing a vector, a matrix and a list and give names to the elements in the list. Access the first and the second element in the list.**

**Code:**

list\_data<- list(c("Red","Green","Blue","Purple"),c(3:10),matrix(4:15,4,3,TRUE))

(list\_data)

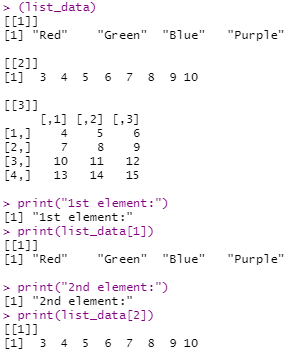
print("1st Element:")

print(list\_data[1])

print("2nd Element:")

print(list\_data[2])

**Output:**



**Program 7**

**Write a R program to create a list containing a vector, a matrix and a list and add element to the end of the list**

**Code:**

list\_data<- list(c("Red","Green","Blue","Purple"),c(3:10),matrix(4:15,4,3,TRUE))

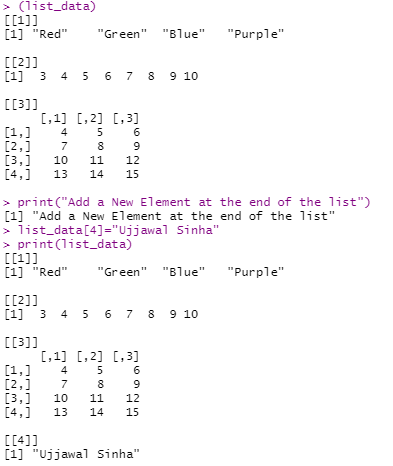
(list\_data)

print("Add a New Element at the end of the list")

list\_data[4]="Ujjawal Sinha"

print(list\_data)

**Output:**



**Program 8**

**Read the following file formats in Python/R:**

**• Comma-separated values**

**• XLSX**

**• ZIP**

**• Plain Text (txt)**

**• JSON**

**• XML**

**• HTML**

**• Images**

**Code:**

**Reading a CSV File**

print(getwd())

setwd("C:/Users/HP/Downloads")

print(getwd())

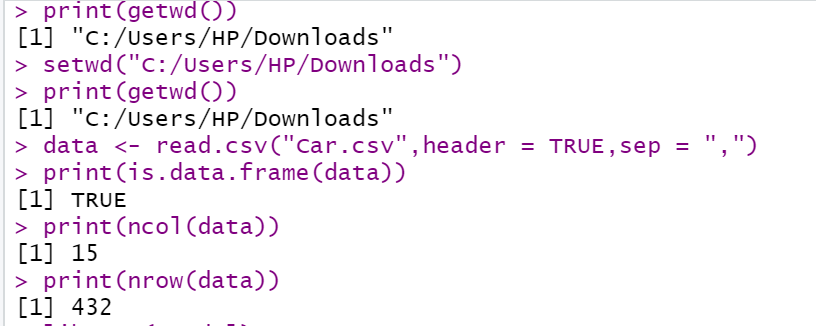
data <- read.csv("Car.csv",header = TRUE,sep = ",")

print(data)

print(is.data.frame(data))

print(ncol(data))

print(nrow(data))



**Reading xlsx File:**

library(readxl)

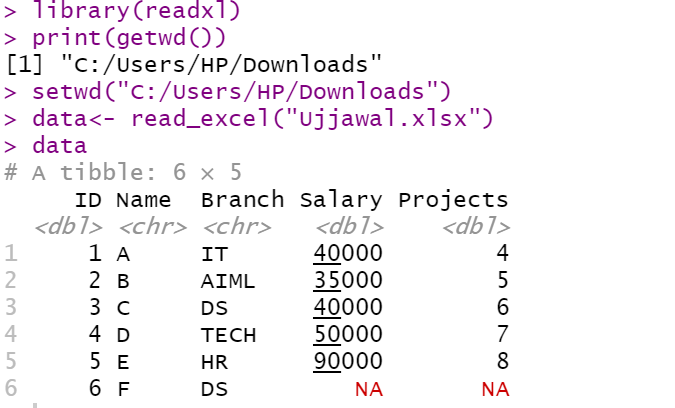
print(getwd())

setwd("C:/Users/HP/Downloads")

data<- read\_excel("Ujjawal.xlsx")

data

**Output:**



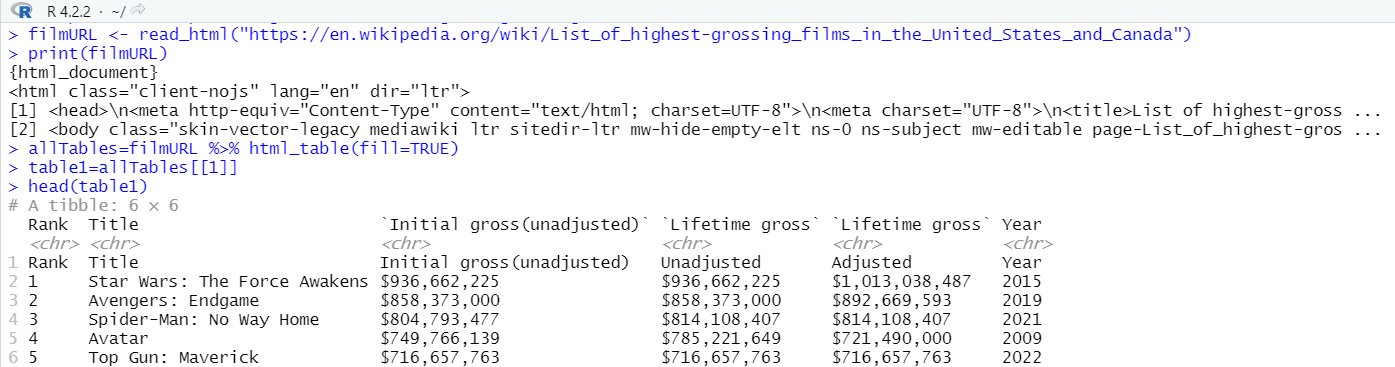
**Reading XML File:**

library("XML")

library("methods")

result <- xmlParse(file = "input.xml")

print(result)

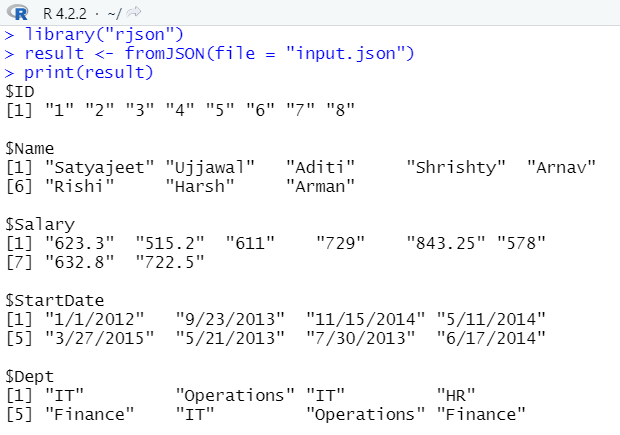
****

**Reading JSON File:**

library("rjson")

result <- fromJSON(file = "input.json")

print(result)

****

**IMAGE Processing in R:**

install.packages("magick")

**library**(magick)

str(magick::magick\_config())

**Read and write**

**library**(magick)

tiger <- image\_read\_svg('http://jeroen.github.io/images/tiger.svg', width = 350)

print(tiger)



image\_write(tiger, path = "tiger.png", format = "png")

**Program 9**

**Write a iris dataset as a list of lists.**

**Compute and print the mean and standard deviation for each of the 4 measured columns ( sepal length and width , petal length and width )**

**Compute and print the mean and standard deviation for each of the 4 measured columns separately for each of the three Iris Species**

**Code:**

library(tidyverse)

view(iris)

df<-data.frame(iris)

summarize(iris)

summarize(iris,mean1=mean(Sepal.Length),SD1=sd(Sepal.Length))

summarize(iris,mean2=mean(Sepal.Width),SD2=sd(Sepal.Width))

summarize(iris,mean3=mean(Petal.Length),SD3=sd(Petal.Length))

summarize(iris,mean4=mean(Petal.Width),SD4=sd(Petal.Width))

grp\_spe<-group\_by(iris,Species)

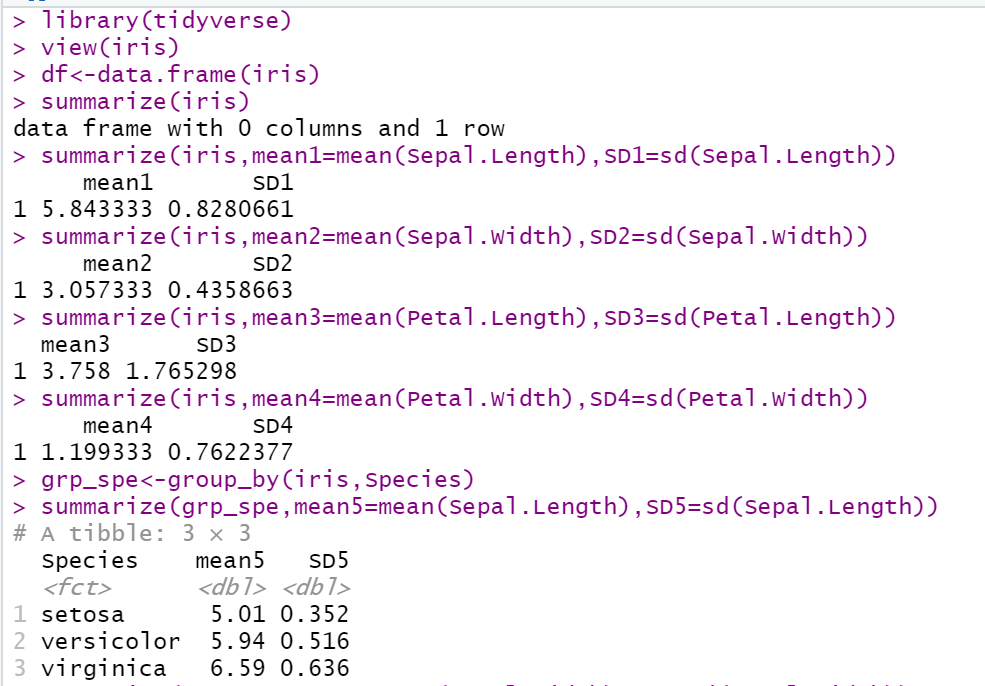
summarize(grp\_spe,mean5=mean(Sepal.Length),SD5=sd(Sepal.Length))

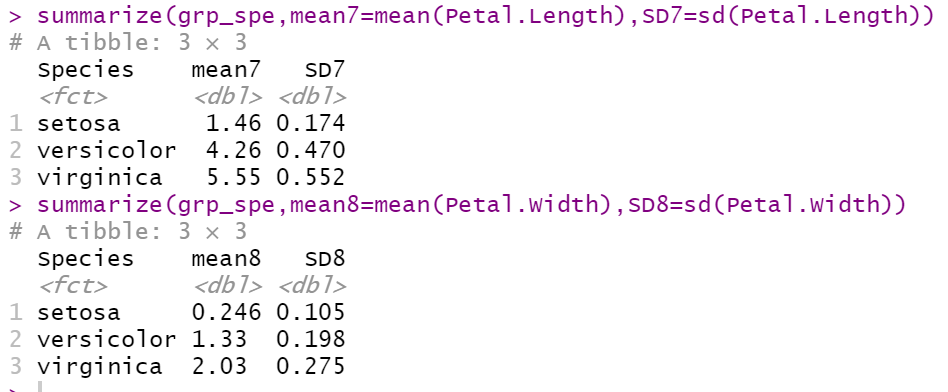
summarize(grp\_spe,mean6=mean(Sepal.Width),SD6=sd(Sepal.Width))

summarize(grp\_spe,mean7=mean(Petal.Length),SD7=sd(Petal.Length))

summarize(grp\_spe,mean8=mean(Petal.Width),SD8=sd(Petal.Width))

**Output:**





**Program 10**

1. **Find a data distribution using box and scatter plot.**
2. **Find the outliers using the box plot**
3. **Plot the bar chart, histogram, and pie chart on sample data**
4. **Plot pie chart, histogram (3D) [including colourful ones]**

**Code:**

Using Boxplot:

input = mtcars[c('mpg','cyl')]

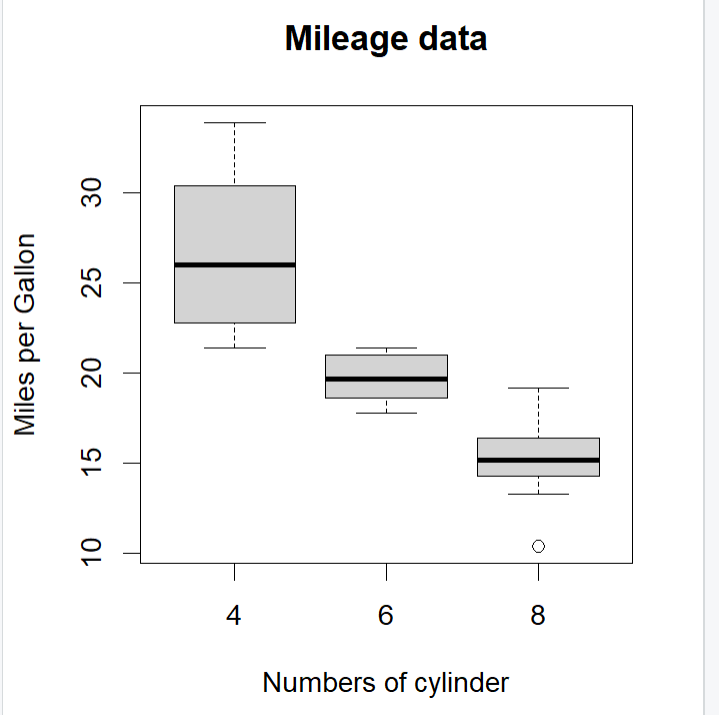
print(head(input))

boxplot(mpg~cyl , data = mtcars , xlab = "Numbers of cylinder" ,

ylab = "Miles per Gallon",

main = "Mileage data")

**Output:**



Using Scatter Plot:

**Code:**

input = mtcars[c('wt','mpg')]

plot(x=input$wt,y=input$mpg,

xlab="Weight",

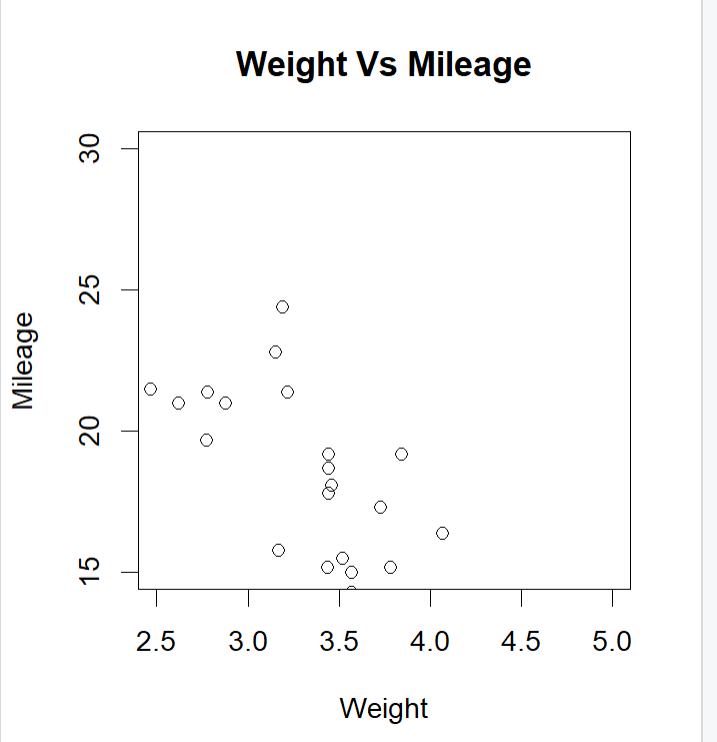
ylab="Mileage",

xlim=c(2.5,5),

ylim=c(15,30),

main="Weight Vs Mileage")

**Output:**



Boxplot with notch:

**Code:**

input = mtcars[c('mpg','cyl')]

print(head(input))

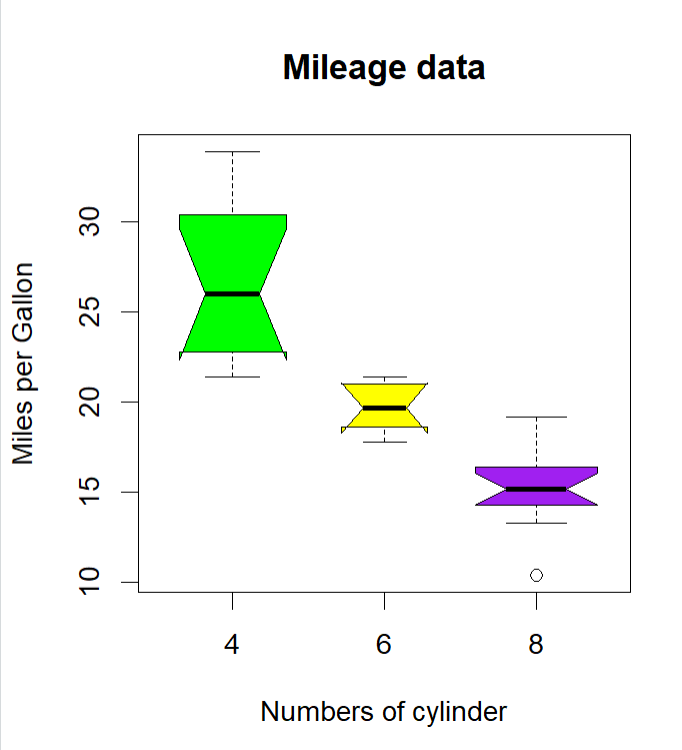
boxplot(mpg~cyl , data = mtcars , xlab = "Numbers of cylinder" ,

ylab = "Miles per Gallon",

main = "Mileage data", notch = TRUE, varwidth = TRUE,

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

**Output:**



Histogram:

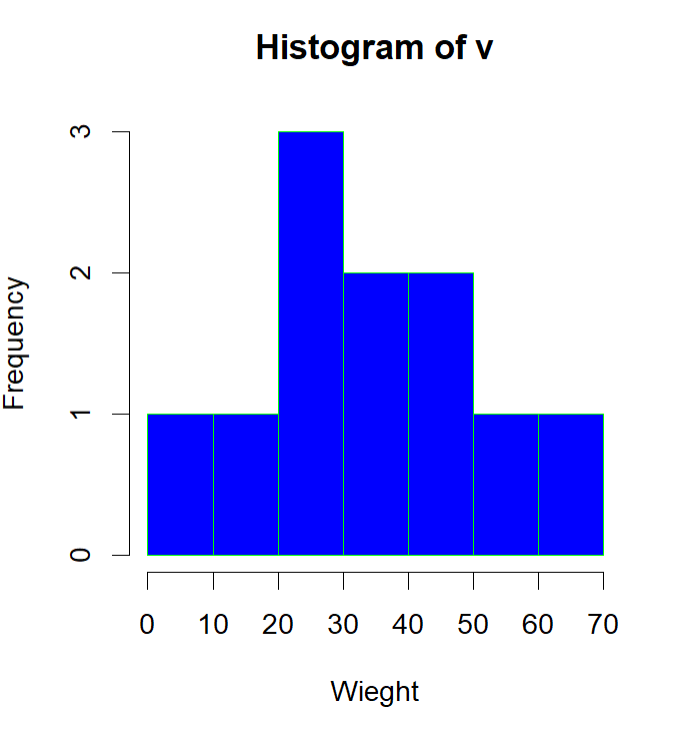
**Code:**

library(graphics)

v <- c(9,21,31,24,25,34,56,43,45,65,11)

hist(v, xlab = "Wieght",col = "Blue",border = "green")

**Output:**



**Bar Chart:**

**Code:**

library(graphics)

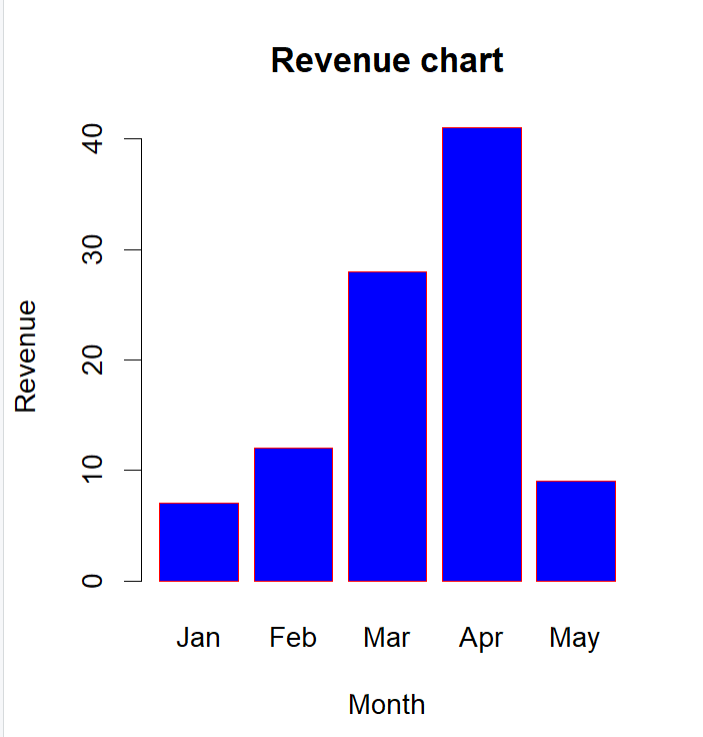
h <- c(7,12,28,41,9)

m <- c("Jan","Feb","Mar","Apr","May")

barplot(h,names.arg = m, xlab="Month",ylab="Revenue",col="blue",

main="Revenue chart",border="red")

**Output:**



**Pie chart :**

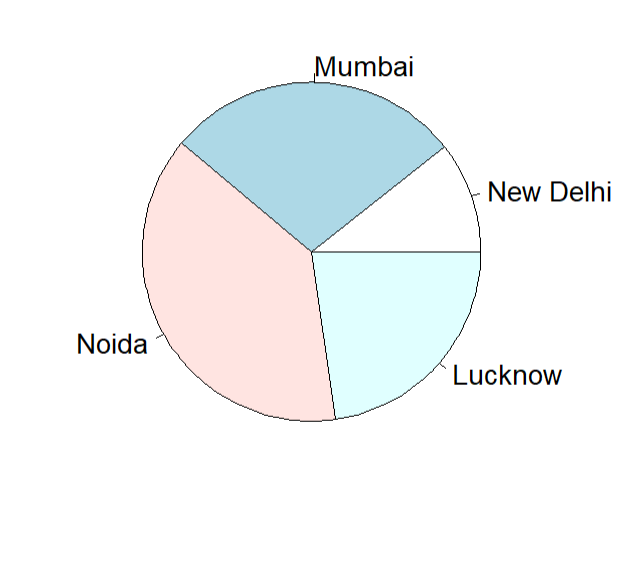
library(graphics)

x <- c(21,56,76,45)

labels <- c("New Delhi","Mumbai","Noida","Lucknow")

pie(x,labels)

**Output:**



**Pie Chart(3D):**

**Code:**

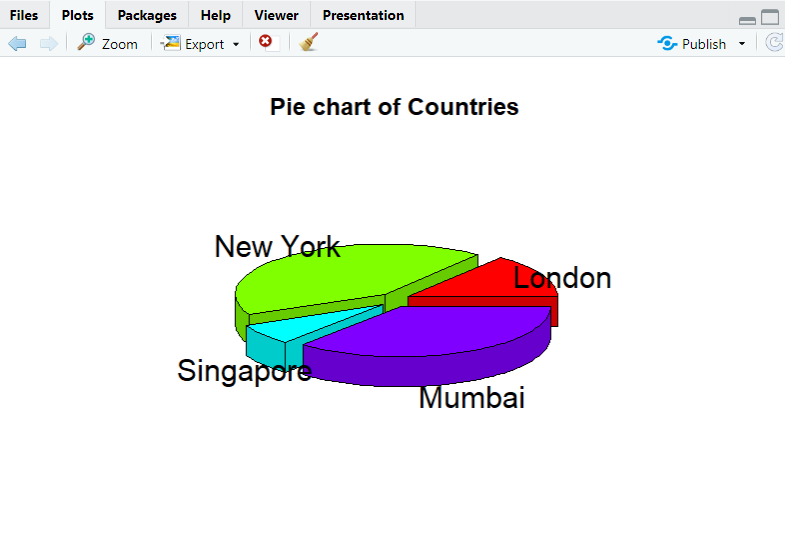
library(plotrix)

x=c(21,62,10,53)

lbl=c("London","New York","Singapore","Mumbai")

pie3D(x,labels=lbl,explode=0.1,main="Pie chart of Countries")

**Output:**



**Program 11**

**Import a sample dataset and perform Linear Regression to find out the relation between variables**

**Code:**

x <- c(151,174,138,165,128,136,179,163,152,131)

y <- c(63,81,56,91,57,67,76,72,62,48)

relation <- lm(y~x)

summary(relation)

print(relation)

a <- data.frame(x=170)

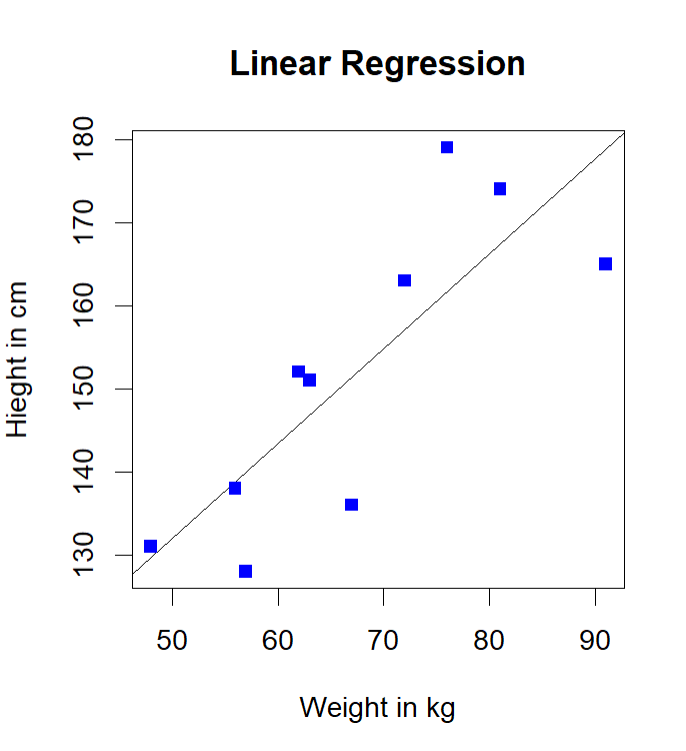
result <- predict (relation,a)

print(result)

plot(y,x,col= "blue",main = "Linear Regression",

abline(lm(x~y)),cex=1,pch=15, xlab= "Weight in kg", ylab = "Hieght in cm")

**Output :**



**Program 12**

**Find the correlation matrix.**

**a. Plot the correlation plot on dataset and visualize giving an overview of relationships among variables on data set.**

**b. Analysis of covariance: variance (ANOVA)if data have categorical variables on data set.**

**Code:**

getwd()

#install.packages("ggplot2")

library(ggplot2)

#install.packages("tidyr")

library(tidyr)

library(datasets)

data("iris")

summary(iris)

#install.packages("DataExplorer")

library(DataExplorer)

install.packages("corrplot")

library(corrplot)

title<-"matrix\_iris"

str(iris)

m<-levels(iris$Species)

title0<-"Setosa"

setosaCorr=cor(iris[iris$Species==m[1],1:4])

setosaCorr

corrplot(setosaCorr,method = "number" , title=title,

mar=c(0.1,1,1,0.1))

plot\_correlation(iris)

**# Covariance ANNOVA = Analylsis if variance**

library(ggplot2)

data(iris)

str(iris)

df = iris

#install.packages("car")

library(car)

leveneTest(Petal.Length~Species,df)

**#Running the actual ANNOVA**

fit=aov(Petal.Length~Species,df)

summary(fit)

#Reporting the result of ANNOVA

#install.packages("psych")

library(psych)

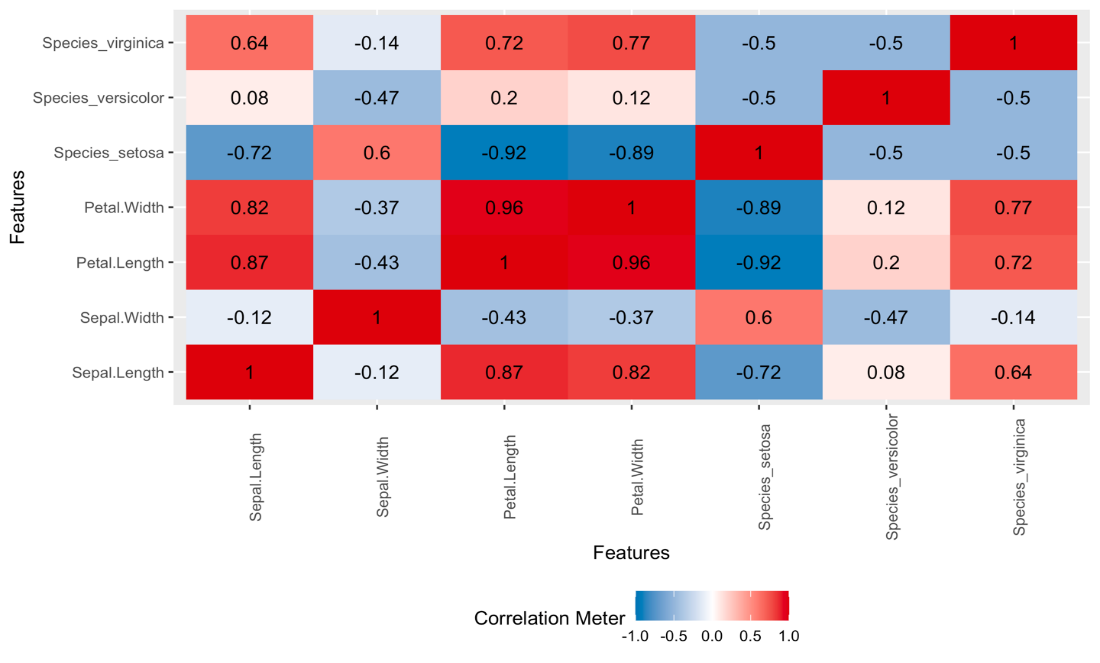
describeBy(df$Petal.Length,df$Species)

ggplot(df, aes(y=Petal.Length, x=Species, fill = Species))+

stat\_summary(fun.y="mean" , geom = "bar" , position = "dodge")+

stat\_summary(fun.data = mean\_se, geom = "errorbar" , position = "dodge" ,width=.8)

**Output:**

****

Text

Description automatically generated

Chart, bar chart

Description automatically generated

**Program 13**

**Write a program to create 3D plot, to add title, change viewing direction, add color and shade to the plot.**

**Code:**

cone = function(x,y){sqrt(x^2 +y^2)}

x = y = seq(-1,1,length=20)

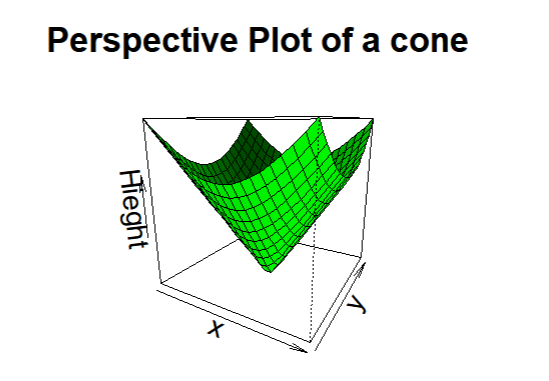
z = outer(x,y,cone)

persp(x,y,z)

persp(x,y,z,main="Perspective Plot of a cone",

zlab = "Hieght", theta = 30,phi=15,col = "green",shade=0.5)

**Output:**



**Program 14**

**a. Create a data frame from the sample data set.**

1. **b. Create a table with the needed variables**
2. **c. Perform the Chi-Square test.**

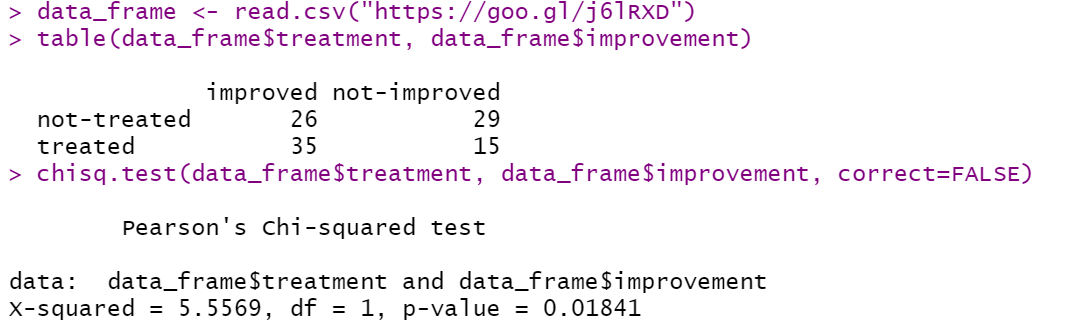
**Code:**

data\_frame <- read.csv("https://goo.gl/j6lRXD")

table(data\_frame$treatment, data\_frame$improvement)

chisq.test(data\_frame$treatment, data\_frame$improvement, correct=FALSE)

**Output:**



**Program 15**

**Perform complete steps of exploratory data analysis on a standard dataset**

**Code:**

data(iris)

iris

head(iris)

dim(iris)

ncol(iris)

nrow(iris)

names(iris)

colnames(iris)

str(iris)

class(iris[,1])

class(iris[,5])

summary(iris)

sd(iris$Sepal.Length)

quantile(iris$Sepal.Length)

#Using apply() to calculate a particular statistic for multiple variables at the same time.

apply(iris[,1:4], 2, sd)

#Use aggregate() function to find summary statistics by group.

aggregate(.~Species, iris, mean)

# group standard deviation

aggregate(.~Species, iris, sd)

# One-way count table

table(iris$Species)

#Re-ordering columns and sorting rows

iris[order(iris$Sepal.Length, decreasing = TRUE)[1:5], ]

iris[order(iris$Sepal.Length, decreasing = TRUE)[1:5], ]

#we want to select variable “Sepal.Length”

iris[, "Sepal.Length"]

#select two variables: “Sepal.Length”, “Sepal.Width”

iris[, c("Sepal.Length", "Sepal.Width")]

**Output:**

