CANARA

Engineering College
Benjanapadavu-574219

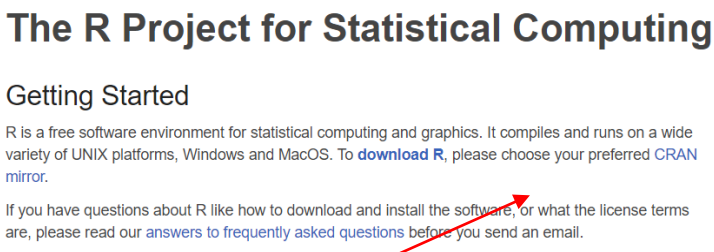
R - PROGRAMMING

Sub Code: **BCS358B**

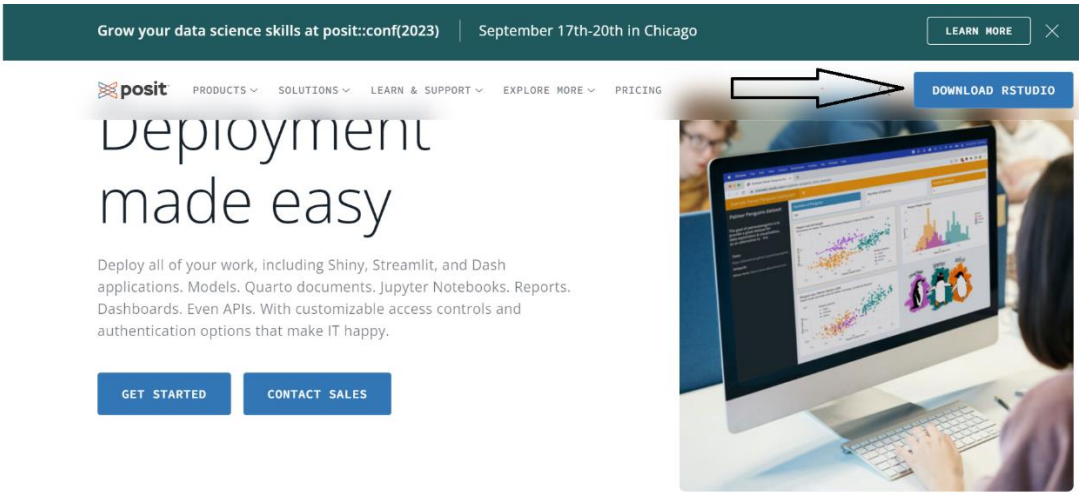
Academic Year:2023-24

Installing R:

1. Go to http://www.r-project.org
2. Click the link that says download R in the "Getting Started" section.



3. Select a mirror closest to you and choose a link in the "Download and Install R" section that is appropriate for your operating system.
4. R Studio is an R specific IDE that can be downloaded from http://www.rstudio.org which gets redirected to https://posit.co/.
5. Click on Download RStudio



6. To get any help in RStudio DO the following:
    a. To get help about a function or dataset that you know, type ? followed by name of the function. ?function_name
    b. To find functions type ?? followed by a related keyword. ??keyword
    c. To find functions type ?? followed by a related special characters, reserved words and multiword search terms use single or double quotes to enclose the search terms. ??"search terms" or ??'keyword'
    d. Alternately you can use help() and help.search() functions but you always have to specify search terms in quotes. help("keyword") , help.search("keyword")
7. # Symbol denotes a comment to document the code.
8. = and <- can both be used as assignment operator.

Packages in R can be installed and loaded or installed but not loaded or not installed. New Packages can be installed using install.packages("Package_Name"). After a package is installed, it can be loaded using library(Package_name).

1. Demonstrate the steps for installation of R and R Studio.

Perform the following:

a) Assign different type of values to variables and display the type of variable. Assign different types such as Double, Integer, Logical, Complex and Character and understand the difference between each data type.

b) Demonstrate Arithmetic and Logical Operations with simple examples.

c) Demonstrate generation of sequences and creation of vectors.

d) Demonstrate Creation of Matrices

e) Demonstrate the Creation of Matrices from Vectors using Binding Function.

f) Demonstrate element extraction from vectors, matrices and arrays

```
rm(list=ls())#Clears the Environment
var1<-123
typeof(var1)
var2<-123L
typeof(var2)
var3<-"apple"
typeof(var2)
var3<-TRUE
var4<-FALSE
var5<-T
var6<-F
typeof(var3)
typeof(var4)
typeof(var5)
typeof(var6)
var7<- 2+3i
typeof(var7)

#Arithmetic Operations - Inspect the Variables in the workspace
a<-1
b<-3
c<-1
sum<-a+b
difference<-b-a
difference1<-a-b
e<-2^3
e2<-2**3
x <- 10/3 #This will store 3.33 in x - FLoating Point Division
10%/%3#Integer DIvision
10%%3#Remainder after division - Modulus
x1 <- 2L
x2 <- 2.5
```

```
x <- x1+x2 #x will have 4.5 as result

#Logical Operations
status1<-a=>b
status2<-a<=b
status3<-!FALSE
status4<-!0
a=6
b=7
a==b
#exponentiaTION
exp(2)

#ARithmetic Operation on Vectors
vec<-seq( from=1, to=9, by=2)
vec1<-1:5
vec2<-6:10
sum_vec<-vec1+vec2
dif_vec<-vec2-vec1
prod_vec<-vec1*vec2#element by element multiplication

#Algebraic operations on vectors act element-wise
x <- c(1,2,3)
y <- c(4,5,6)
x*y # [1] 4 10 18
x+y # [1] 5 7 9
y^x # [1] 4 25 216

#Recycling of values-
#When algebraic expressions are applied
#on two vectors of unequal lengths, R
#automatically repeats the shorter vector
#until it has something that has the same
#length as the longer vector.

c(1,2,3,4) + c(1,2) # [1] 2 4 4 6
(1:10)^c(1,2) # [1] 1 4 3 16 5 36 7 64 9 100
2+c(1,2,3) # [1] 3 4 5
2*c(1,2,3) # [1] 2 4 6
(1:10)^2 # [1] 1 4 9 16 25 36 49 64 81 100

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

```
print(v&t)  #[1]  TRUE  TRUE FALSE  TRUE

#Example:Find all integers between 1 & 20 that are divisible by 4.
x<-1:20
(y <- x[x %% 4 == 0]) #[1]  4  8 12 16 20

#Using concatenate c function to create vectors
a<-c(1,2,3,4,5)
b<-c(6,7,8,9,10)
d<-a+b

#Create a matrix by binding vectors as rows
a<-c(1,2,3,4,5)
b<-c(6,7,8,9,10)
rb<-rbind(a,b)
rb
#Create a matrix by binding vectors as columns
cb<-cbind(a,b)
cb
#Accessing Vector elements
a[1]
a[1:3]

#CReating Matrices & accessing Values
A <- matrix(1:6, nrow=2, ncol=3,byrow=TRUE)
B <- matrix(2:7, nrow=2, ncol=3,byrow=TRUE)
A
A[1,3]<-0#CHanges the 1st row 3rd column value to 0
A[, 2:3] #References all the rows and columns from 2 to 3

#CReating an Array out of a vector or matrix
arr_a=as.array(a)
arr_a
d<-c(1,2,"Apple",5,6)
list_d=as.list(d)
list_d

arr_a[1]
arr_a[1:3]

list_d[1]
list_d[3]
list_d[3]
```

2. Assess the Financial Statement of an Organization being supplied with 2 vectors of data: Monthly Revenue and Monthly Expenses for the Financial Year. You can create your own sample data vector for this experiment) Calculate the following financial metrics:

a. Profit for each month.
b. Profit after tax for each month (Tax Rate is 30%).
c. Profit margin for each month equals to profit after tax divided by revenue.
d. Good Months – where the profit after tax was greater than the mean for the year.
e. Bad Months – where the profit after tax was less than the mean for the year.
f. The best month – where the profit after tax was max for the year.
g. The worst month – where the profit after tax was min for the year.
Note:
a. All Results need to be presented as vectors
b. Results for Dollar values need to be calculated with $0.01 precision, but need to be presented in Units of $1000 (i.e 1k) with no decimal points
c. Results for the profit margin ratio need to be presented in units of % with no decimal point.
d. It is okay for tax to be negative for any given month (deferred tax asset)
e. Generate CSV file for the data.

```
rm(list=ls())

#Data
revenue <- c(14574.49, 7606.46, 8611.41, 9175.41, 8058.65, 8105.44, 11496.28, 9766.09, 10305.32,
14379.96, 10713.97, 15433.50)
expenses <- c(12051.82, 5695.07, 12319.20, 12089.72, 8658.57, 840.20, 3285.73, 5821.12, 6976.93,
16618.61, 10054.37, 3803.96)
months<-
c("January","February","March","April","May","June","July","August","September","October","Novemb
er","December")

#Profit for each month.
profit <- revenue - expenses
profit


#Profit after tax for each month
tax<-round(profit*0.3)
tax

profit.after.tax<-(profit-tax)
profit.after.tax

#Profit margin for each month
profit.margin<-round(profit.after.tax/revenue,2)*100
```

```
profit.margin

#Good Months
mean_pat<- mean(profit.after.tax)
mean_pat

good.months<-profit.after.tax>mean_pat
good.months

#Bad Months
bad.months<-!good.months
bad.months

#The best month
best.month<-profit.after.tax == max(profit.after.tax)
best.month

#The worst month
worst.month<-profit.after.tax == min(profit.after.tax)
worst.month

#Result Presentation according to requirement
revenue.1000<-round(revenue/1000,0)
revenue.1000

expenses.1000<-round(expenses/1000,0)
expenses.1000

profit.1000<-round(profit/1000,0)
profit.1000

profit.after.tax.1000<-round(profit.after.tax/1000,0)
profit.after.tax.1000

#Output
revenue.1000
expenses.1000
profit.1000
profit.after.tax.1000
profit.margin
good.months
bad.months
best.month
```

```
financials <- rbind(
  revenue.1000,
  expenses.1000,
  profit.1000,
  profit.after.tax.1000,
  profit.margin,
  good.months,
  bad.months,
  best.month,
  worst.month
)

colnames(financials) <- months
View(financials)
write.csv(financials, file= "Financial Statement.csv", append=FALSE,  row.names=TRUE, col.names=TRUE)
```

3. Develop a program to create two 3 X 3 matrices A and B and perform the following operations a) Transpose of the matrix b) addition c) subtraction d) multiplication

```
A <- matrix(1:9, nrow=3, ncol=3,byrow=TRUE)#Create MAtrix A
B <- matrix(2:10, nrow=3, ncol=3,byrow=TRUE)#Create MAtrix B
AT<-t(A)#Transpose of Matrix using t()
mat_sum<-A+B#Add MAtrix
mat_dif<-A-B #get difference of Matrix A and B
mat_dif1<-B-A
A
B
mat_sum
mat_prod<-A*B#Vector MUltiplication
mat_prod1<-A%*%B#Matrix Multiplication
mat_prod1
```

4. Develop a program to find the factorial of given number using recursive function calls.

```
#Method 1:
nfact2<-function(n){
 if(n==1){
  cat("Called nfact2(1)\n")
  return(1)
 }else{
  cat("called nfact2(",n,")\n",sep="")
  return(n*nfact2(n-1))
 }
}
nfact2(6)

#Method 2:
n_factorial<-function(n){
#Calculate n Factorial
 n_fact<-prod(1:n)
 return(n_fact)
}

n_factorial(6)
```

5. Develop an R Program using functions to find all the prime numbers up to a specified number by the method of Sieve of Eratosthenes.

```
primesieve<- function(sieved,unsieved){
  p<-unsieved[1]
  n<-unsieved[length(unsieved)]
  if(p^2 >n){
    return(c(sieved, unsieved))
  }else{
    unsieved<-unsieved[unsieved%%p!=0]
    sieved<-c(sieved,p)
    return(primesieve(sieved,unsieved))
  }
}
primesieve(c(),2:200)
```

6. The built-in data set mammals contain data on body weight versus brain weight. Develop R commands to:

a) Find the Pearson and Spearman correlation coefficients. Are they similar?

b) Plot the data using the plot command.

c) Plot the logarithm (log) of each variable and see if that makes a difference.

```
#Click on the Functions Tab on the Right Hand Side of the IDE. Select MASS.
# Part a: Find the Pearson and Spearman correlation coefficients. Are they similar?
data(mammals)
pearson_corr <- cor(mammals$brain, mammals$body, method = "pearson")
spearman_corr <- cor(mammals$brain, mammals$body, method = "spearman")
print(paste("Pearson correlation coefficient:", pearson_corr))
print(paste("Spearman correlation coefficient:", spearman_corr))

# Part b: Plot the data using the plot command
plot(mammals$body, mammals$brain, xlab = "Body Weight", ylab = "Brain Weight", main = "Body
Weight vs. Brain Weight")

# Part c: Plot the logarithm (log) of each variable and see if that makes a difference
plot(log(mammals$body), log(mammals$brain), xlab = "log(Body Weight)", ylab = "log(Brain Weight)",
main = "log(Body Weight) vs. log(Brain Weight)")
```

7. Write an R program to create a Data Frame with following details and do the following operations.

itemCode	itemCategory	itemPrice
1001	Electronics	700
1002	Desktop Supplies	300
1003	Office Supplies	350
1004	USB	400
1005	CD Drive	800

a) Subset the Data frame and display the details of only those items whose price is greater than or equal to 350.

b) Subset the Data frame and display only the items where the category is either "Office Supplies" or "Desktop Supplies"

c) Create another Data Frame called "item-details" with three different fields itemCode, ItemQtyonHand and ItemReorderLvl and merge the two frames

```
rm(list=ls())
item1<- c('1001','Electronics','700')
item2<- c('1002','Desktop Supplies','300')
item3<- c('1003','Office Supplies','350')
item4<- c('1004','USB','400')
item5<- c('1005','CD Drive','800')

# Add rows
itemList = list(item1,item2,item3,item4,item5)
#Set Column Names
columns<-c('itemCode','itemCategory','itemPrice')
#Create Empty DataFrame
itemData = data.frame(matrix(nrow = 0, ncol = length(columns)))
# assign column names
colnames(itemData) = columns
View(itemData)
#Append Rows to Dataframe
for(item in itemList){
  itemData[nrow(itemData)+1,]<-item
}
View(itemData)
#Create Required subsets
sub_itemData_gt350<-subset(itemData, itemData$itemPrice >= 350)
View(sub_itemData_gt350)

sub_itemData_supplies<-subset(itemData, (itemData$itemCategory=='Office Supplies')
|(itemData$itemCategory=='Desktop Supplies'))
View(sub_itemData_supplies)
```

```
#CReate Seond Dataframe
itemDetailsColumn<-c('itemCode','ItemQtyonHand','ItemReorderLvl')
itemDetails = data.frame(matrix(nrow = 0, ncol = length(itemDetailsColumn)))
colnames(itemDetails) = itemDetailsColumn
View(itemDetails)
itemDetails1<- c('1001','15','2')
itemDetails2<- c('1002','10','2')
itemDetails3<- c('1003','20','4')
itemDetails4<- c('1004','12','5')
itemDetails5<- c('1005','5','1')
itemDetailsList = list(itemDetails1,itemDetails2,itemDetails3,itemDetails4,itemDetails5)

for(itemDetail in itemDetailsList){
  itemDetails[nrow(itemDetails)+1,]<-itemDetail
}

View(itemDetails)

#Performing an Inner Join merge
merged <- merge(itemData, itemDetails, by = "itemCode")
merged

#Example to Explain concept of Joins
df1 <- data.frame(Id = c(333, 444, 555, 777, 999),
                  Product = c("laptop", "printer", "tablet", "desk", "chair"))

df2 <- data.frame(Id = c(333, 444, 666, 777, 888),
                  Price = c(1200, 150, 300, 450, 200))

inner_merged <- merge(df1, df2, by = "Id")
print(inner_merged)

left_merged <- merge(df1, df2, by = "Id", all.x = TRUE)
print(left_merged)


right_merged <- merge(df1, df2, by = "Id", all.y = TRUE)
print(right_merged)

outer_merged <- merge(df1, df2, by = "Id", all = TRUE)
print(outer_merged)
```

8. Let us use the built-in dataset air quality which has Daily air quality measurements in New York, May to September 1973. Create a histogram by using appropriate arguments for the following statements.

a) Assigning names, using the air quality data set.
b) Change colors of the Histogram
c) Remove Axis and Add labels to Histogram
d) Change Axis limits of a Histogram
e) Create a Histogram with density and Add Density curve to the histogram

```
View(airquality)
air<-airquality
library(ggplot2)
library("gridExtra")
air$Month <- factor(air$Month,labels=c("May","June","July","August","September"))
gg1 <- ggplot(air,aes(x=1:nrow(air),y=Temp)) +
 geom_line(aes(col=Month)) +
 geom_point(aes(col=Month,size=Wind)) +
 geom_smooth(method="loess",col="black") +
 labs(x="Time (days)",y="Temperature (F)")

gg2 <- ggplot(air,aes(x=Solar.R,fill=Month)) +
 geom_density(alpha=0.4) +
 labs(x=expression(paste("Solar radiation (",ring(A),")")),
    y="Kernel estimate")
gg3 <- ggplot(air,aes(x=Wind,y=Temp,color=Month)) +
 geom_point(aes(size=Ozone)) +
 geom_smooth(method="lm",level=0.9,fullrange=FALSE,alpha=0.2) +
 labs(x="Wind speed (MPH)",y="Temperature (F)")

grid.arrange(gg1,gg2,gg3)

s<-ggplot(data=air,aes(x=Ozone, y=..density..))
s+geom_histogram(binwidth=10,aes(fill=Month),colour="Black")+
 geom_density(lwd = 1, colour = 4,
        fill = 4, alpha = 0.25)

s<-ggplot(data=air,aes(x=Temp, y=..density..))
s+geom_histogram(binwidth=10,aes(fill=Month),colour="Black")+
 geom_density(lwd = 1, colour = 4,
        fill = 4, alpha = 0.25)
```

9. Design a data frame in R for storing about 20 employee details. Create a CSV file named "input.csv" that defines all the required information about the employee such as id, name, salary, start_date, dept. Import into R and do the following analysis.

a) Find the total number rows & columns

b) Find the maximum salary

c) Retrieve the details of the employee with maximum salary

d) Retrieve all the employees working in the IT Department.

e) Retrieve the employees in the IT Department whose salary is greater than 20000 and write these details into another file "output.csv"

Sl.No	id	name	salary	start_date	department
1	2005001	Clark Kent	15000	01-01-2005	Media
2	2005002	Bruce Wayne	80000	01-01-2005	Executive
3	2005003	Hal Jordan	10000	01-01-2005	Defence
4	2005004	Barry Allen	15000	01-04-2005	Forensics
5	2005005	Wally West	15000	01-04-2005	Media
6	2006001	Arthur Curry	30000	03-05-2006	Catering
7	2007001	Oliver Queen	80000	01-10-2007	Executive
8	2007002	Richard Grayson	5000	01-10-2007	IT
9	2007003	Tim Drake	15000	01-10-2007	IT
10	2007004	Jason Todd	20000	01-10-2007	IT
11	2007005	Tony Stark	90000	15-11-2007	Executive
12	2007006	Steve Rogers	25000	15-11-2007	Sports
13	2007007	Peter Parker	20000	15-11-2007	Media
14	2008001	Clint Barton	40000	02-02-2008	IT
15	2008002	Bruce Banner	30000	02-02-2008	R&D
16	2008003	Reed Richards	35000	02-02-2008	R&D
17	2008004	Ellie Satler	40000	15-02-2008	R&D
18	2008005	John Hammond	90000	01-05-2008	Executive
19	2008006	Ian Malcolm	60000	01-05-2008	Accounts
20	2008007	Alan Grant	60000	01-05-2008	R&D

```
setwd('C:/Users/Praahas/Projects/R-Lab/dataset')
# Importing the dataset
dataset = read.csv('input.csv')
View (dataset)
typeof(dataset)
dataset
nrow(dataset)
ncol(dataset)
```

```
max_salary = max(dataset$salary)
max_salary

max_salaries <- subset(dataset, dataset$salary == max(dataset$salary))
max_salaries

IT_Dept<-subset(dataset, dataset$department == 'IT')
IT_Dept

salary_gt_value<-subset(dataset, dataset$salary > 20000)
salary_gt_value
write.csv(salary_gt_value,"output.csv", row.names = FALSE)

#Append new line to dataframe
dataset[nrow(dataset)+1,]<-list(nrow(dataset)+1,'2023001','XYZ','5000','01-01-2023','IT')

View(dataset)
```

10. **Using the built in dataset mtcars which is a popular dataset consisting of the design and fuel consumption patterns of 32 different automobiles. The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models). Format A data frame with 32 observations on 11 variables : [1] mpg Miles/(US) gallon, [2] cyl Number of cylinders [3] disp Displacement (cu.in.), [4] hp Gross horsepower [5] drat Rear axle ratio,[6] wt Weight (lb/1000) [7] qsec 1/4 mile time, [8] vs V/S, [9] am Transmission (0 = automatic, 1 = manual), [10] gear Number of forward gears, [11] carb Number of carburetors**

Answer the following:

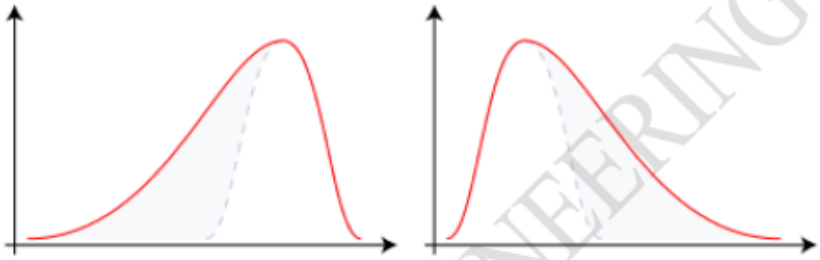
a) What is the total number of observations and variables in the dataset?

b) Find the car with the largest hp and the least hp using suitable functions

c) Plot histogram / density for each variable and determine whether continuous variables are normally distributed or not. If not, what is their skewness?

d) What is the average difference of gross horse power(hp) between automobiles with 4 and 8 number of cylinders(cyl)? Also determine the difference in their standard deviations.

e) Which pair of variables has the highest Pearson correlation?



Negative Skew Positive Skew

#https://www.freecodecamp.org/news/skewness-and-kurtosis-in-statistics-explained/
#https://community.gooddata.com/metrics-and-maql-kb-articles-43/normality-testing-skewness-and-kurtosis-241
#In statistics, skewness is a measure of the asymmetry of the probability distribution of a random
#variable about its mean. In other words, skewness tells you the amount and direction of skew
#(departure from horizontal symmetry). The skewness value can be positive or negative, or even
#undefined. If skewness is 0, the data are perfectly symmetrical, although it is quite unlikely for real-
#world data. As a general rule of thumb:
#If skewness is less than -1 or greater than 1, the distribution is highly skewed.
#If skewness is between -1 and -0.5 or between 0.5 and 1, the distribution is moderately skewed.
##If skewness is between -0.5 and 0.5, the distribution is approximately symmetric.

#Kurtosis tells you the height and sharpness of the central peak, relative to that of a standard bell curve.
#kurtosis is 3 for a normal distribution

```
rm(list=ls())
#library(moments)
library(e1071)
library(ggplot2)
mtcars
#Read description about the dataset
?mtcars
carData <- mtcars
#TOtal NUmber of Observations
# Use dim() to find the dimension of the data set
number.of.observations<-dim(carData)[1]
number.of.variables<-dim(carData)[2]

car.with.largest.hp<-row.names(subset(carData, carData$hp==max(carData$hp)))
car.with.largest.hp

car.with.least.hp<-row.names(subset(carData, carData$hp==min(carData$hp)))
car.with.least.hp

hist(carData$mpg,col="green", border="black",prob = TRUE)
lines(density(carData$mpg),col = "chocolate3")
skewness(carData$mpg,type=2)
kurtosis(carData$mpg,type=2)

hist(carData$disp,col="green",border="black",prob = TRUE)
lines(density(carData$disp),col = "chocolate3")
skewness(carData$disp,type=2)
kurtosis(carData$disp,type=2)

hist(carData$hp,col="green",border="black", prob = TRUE)
lines(density(carData$hp),col = "chocolate3")
skewness(carData$hp,type=2)
kurtosis(carData$hp,type=2)

hist(carData$drat,col="green",border="black",prob = TRUE)
lines(density(carData$drat),col = "chocolate3")
skewness(carData$drat,type=2)
kurtosis(carData$drat,type=2)

hist(carData$wt,col="green",border="black",prob = TRUE)
lines(density(carData$wt),col = "chocolate3")
skewness(carData$wt,type=2)
```

```
kurtosis(carData$wt,type=2)

hist(carData$qsec,col="green",border="black",prob = TRUE)
lines(density(carData$qsec),col = "chocolate3")
skewness(carData$qsec,type=2)
kurtosis(carData$qsec,type=2)

#https://www.freecodecamp.org/news/skewness-and-kurtosis-in-statistics-explained/
#https://community.gooddata.com/metrics-and-maql-kb-articles-43/normality-testing-skewness-and-kurtosis-241
#In statistics, skewness is a measure of the asymmetry of the probability distribution of a random variable about its mean. In other words, skewness tells you the amount and direction of skew (departure from horizontal symmetry). The skewness value can be positive or negative, or even undefined. If skewness is 0, the data are perfectly symmetrical, although it is quite unlikely for real-world data. As a general rule of thumb:
#If skewness is less than -1 or greater than 1, the distribution is highly skewed.
#If skewness is between -1 and -0.5 or between 0.5 and 1, the distribution is moderately skewed.
##If skewness is between -0.5 and 0.5, the distribution is approximately symmetric.

#Kurtosis tells you the height and sharpness of the central peak, relative to that of a standard bell curve.
#kurtosis is 3 for a normal distribution

carData.4cyl.hp<-subset(carData,carData$cyl==4)["hp"]
carData.8cyl.hp<-subset(carData,carData$cyl==8)["hp"]
diff.gross.hp<-round(mean(carData.8cyl.hp[,1])-mean(carData.4cyl.hp[,1]))
print(diff.gross.hp)
diff.sd<-round(sd(carData.8cyl.hp[,1])-sd(carData.4cyl.hp[,1]))
print(diff.sd)

print(cor(carData$mpg, carData$cyl, method = 'pearson'))
print(cor(carData$mpg, carData$disp, method = 'pearson'))
print(cor(carData$mpg, carData$hp, method = 'pearson'))
print(cor(carData$mpg, carData$drat, method = 'pearson'))
print(cor(carData$mpg, carData$wt, method = 'pearson'))
print(cor(carData$mpg, carData$qsec, method = 'pearson'))

print(cor(carData$cyl, carData$disp, method = 'pearson'))
print(cor(carData$cyl, carData$hp, method = 'pearson'))
print(cor(carData$cyl, carData$drat, method = 'pearson'))
print(cor(carData$cyl, carData$wt, method = 'pearson'))
print(cor(carData$cyl, carData$qsec, method = 'pearson'))

print(cor(carData$disp, carData$hp, method = 'pearson'))
```

```
print(cor(carData$disp, carData$drat, method = 'pearson'))
print(cor(carData$disp, carData$wt, method = 'pearson'))
print(cor(carData$disp, carData$qsec, method = 'pearson'))

print(cor(carData$hp, carData$drat, method = 'pearson'))
print(cor(carData$hp, carData$wt, method = 'pearson'))
print(cor(carData$hp, carData$qsec, method = 'pearson'))

print(cor(carData$drat, carData$wt, method = 'pearson'))
print(cor(carData$drat, carData$qsec, method = 'pearson'))

print(cor(carData$wt, carData$qsec, method = 'pearson'))

#Interpretation of Pearson Correlation Coefficient
#The value of the correlation coefficient (r) lies between -1 to +1. When the value of –
#r=0; there is no relation between the variable.
#r=+1; perfectly positively correlated.
#r=-1; perfectly negatively correlated.
#r= 0 to 0.30; negligible correlation.
#r=0.30 to 0.50; moderate correlation.
#r=0.50 to 1 highly correlated.

cor(carData)

cor_df <- round(cor(carData), 2)

#melt the data frame
melted_cormat <- melt(cor_df)

# https://www.statology.org/correlation-heatmap-in-r/
#create correlation heatmap
ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
 geom_tile() +
 geom_text(aes(Var2, Var1, label = value), size = 5) +
 scale_fill_gradient2(low = "blue", high = "red",
            limit = c(-1,1), name="Correlation") +
 theme(axis.title.x = element_blank(),
     axis.title.y = element_blank(),
     panel.background = element_blank())
```

11. Demonstrate the progression of salary with years of experience using a suitable data set (You can create your own dataset). Plot the graph visualizing the best fit line on the plot of the given data points. Plot a curve of Actual Values vs. Predicted values to show their correlation and performance of the model. Interpret the meaning of the slope and y-intercept of the line with respect to the given data. Implement using lm function. Save the graphs and coefficients in files. Attach the predicted values of salaries as a new column to the original data set and save the data as a new CSV file.

Sample CSV:

YearsExperience	Salary
1.1	39343
1.3	46205
1.5	37731
2	43525
2.2	39891
2.9	56642
3	60150
3.2	54445
3.2	64445
3.7	57189
3.9	63218
4	55794
4	56957
4.1	57081
4.5	61111
4.9	67938
5.1	66029
5.3	83088
5.9	81363
6	93940
6.8	91738
7.1	98273
7.9	101302
8.2	113812
8.7	109431
9	105582
9.5	116969
9.6	112635
10.3	122391
10.5	121872

```
setwd('PATHNAME')
# Importing the dataset
dataset = read.csv('Salary_Data.csv')
View (dataset)
# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Salary, SplitRatio = 2/3)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
# Feature Scaling
# training_set = scale(training_set)
# test_set = scale(test_set)

# Fitting Simple Linear Regression to the Training set
regressor = lm(formula = Salary ~ YearsExperience,
               data = training_set)
regressor
?lm()
# Predicting the Test set results
y_pred = predict(regressor, newdata = test_set)
# Visualising the Training set results
library(ggplot2)
ggplot() +
  geom_point(aes(x = training_set$YearsExperience, y = training_set$Salary),
             colour = 'red') +
  geom_line(aes(x = training_set$YearsExperience, y = predict(regressor, newdata = training_set)),
            colour = 'blue') +
  ggtitle('Salary vs Experience (Training set)') +
  xlab('Years of experience') +
  ylab('Salary')

# Visualising the Test set results
library(ggplot2)
ggplot() +
  geom_point(aes(x = test_set$YearsExperience, y = test_set$Salary),
             colour = 'red') +
  geom_line(aes(x = training_set$YearsExperience, y = predict(regressor, newdata = training_set)),
            colour = 'blue') +
  ggtitle('Salary vs Experience (Validation set)') +
  xlab('Years of experience') +
  ylab('Salary')
```