R PROGRAMMING

UNIT - 1

Introduction To R Programming

1. What is R Programming?

R is a programming language and software environment for statistical analysis, graphics representation and reporting.

2. List Features of R?

- R is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.
- R has an effective data handling and storage facility,
- R provides a suite of operators for calculations on arrays, lists, vectors and matrices.
- R provides a large, coherent and integrated collection of tools for data analysis.
- R provides graphical facilities for data analysis.

3. List rules to declare R variables?

- A variable name in R can be created using letters, digits, periods, and underscores.
- You can start a variable name with a letter or a period, but not with digits.

- If a variable name starts with a dot, you can't follow it with digits.
- R is case sensitive. This means that age and Age are treated as different variables.
- We have some reserved words that cannot be used as variable names

4. List Types of R variables?

1. Boolean Variables - It stores single bit data which is either TRUE or FALSE

2. Integer Variables - It stores numeric data without any decimal values

3. Floating Point Variables - It stores numeric data with decimal values

4. Character Variables - It stores a single character data

5. String Variables - It stores data that is composed of more than one character.

5. List Type of constants in R?

- Numeric, integer, complex, logical, string.
- Null, NA, Inf, NaN.

6. List special R constants?

- **LETTERS -** list of uppercase letters
- **letters -** list of lowercase letters
- **month.abb -** letters abbrviation of english months
- **pi -** numerical value of constant pi

7. List ways of variable assignment?

1. First way -

> x <- 6 # assignment operator: a less-than character (<) and a hyphen (-) with no space

2. Second way -

```
> y = 3 # assignment operator = is used.
```

3. Third way -

```
> z <<- 9 # assignment to a global variable rather than a local variable.
```

4. Fourth way -

```
> 5 -> fun #A rightward assignment operator (->) can be used anywhere
```

5. Fifth way -

```
> a <- b <- 7 # Multiple values can be assigned simultaneously.
```

6. Sixth way -

```
> assign("k",12) # assign function can be used.
```

8. List types of data types in R?

1. logical -

The logical data type in R is also known as boolean data type. It can only have two values: TRUE and FALSE.

2. numeric -

In R, the numeric data type represents all real numbers with or without decimal values.

3. integer -

The integer data type specifies real values without decimal points. We use the suffix L to specify integer data.

4. complex -

The complex data type is used to specify purely imaginary values in R. We use the suffix i to specify the imaginary part.

5. character -

The character data type is used to specify character or string values in a variable.

6. raw -

A raw data type specifies values as raw bytes.

- charToRaw() - converts character data to raw data
- rawToChar() - converts raw data to character data

9. What is coercion?

The process of altering the data type of an object to another type is referred to as **coercion** or data type conversion.

Types of Coercion -

Implicit coercion occurs automatically when elements need to be converted to another type in order for an operation to complete.

This explicit coercion can be achieved with the **as-dot functions.**

Syntax:-

```
as.data_type(object)
```

Example:-

```
print(as.numeric(TRUE))
```

10. List types of operators in R language?

- Arithmetic Operators
- Logical Operators

- Relational Operators
- Assignment Operators
- Miscellaneous Operator

11. List common data structures in R?

1. Vector -

A vector is an ordered collection of basic data types of a given length. The only key thing here is all the elements of a vector must be of the identical data type. Vector is a one dimensional data structure.

The function for creating a vector is the single **letter c**

2. List -

Lists are the R objects which contain elements of different types like — numbers,strings, vectors and another list inside it.

In R, we use the **list()** function to create a list.

mylist <- list(object1, object2, ...)

3. Array -

An Array is a data structure which can store data of the same type in more than two dimensions.

In R, we use the array() function to create an array.

Syntax -

array(vector, dim = c(nrow, ncol, nmat))

Here,

vector - the data items of same type

nrow - number of rows

ncol - number of columns

nmat - the number of matrices of nrow * ncol dimension

The only difference between vectors, matrices, and arrays are -

- Vectors are uni-dimensional arrays
- Matrices are two-dimensional arrays
- Arrays can have more than two dimensions

4. Matrices -

A matrix is simply several vectors stored together. Whereas the size of a vector is described by its length, the size of a matrix is specified by a number of rows and a number of columns.

Example -

```
# create a 2 by 3 matrix

matrix1 <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3, byrow = TRUE)

print(matrix1)
```

5. Data Frame -

Data frames are tabular data objects. Has both rows and columns analogous to excel spreadsheet. Unlike a matrix in data frame each column can contain different modes of data.

A data frame is created with the **data.frame()** function -

```
mydata <- data.frame(col1, col2, col3,…)
```

6. Factors-

Factors are data structures used to categorize and store data on multiple levels.The main advantage is that it can store both Integer and Character types of data.

We can create a factor using the function **factor().**

12. Explain Classes in R?

A class is a blueprint to create objects. An object is simply a collection of data and methods or an object is an instance of a class.

Class System in R -

1. S3 Class -

S3 class is somewhat primitive in nature. It lacks a formal definition and objects of this class can be created simply by adding a class attribute to it.

With the help of the S3 class, we can take advantage of the ability to implement the generic function OO.

2. S4 Class -

S4 class is an improvement over the S3 class. They have a formally defined structure which helps in making objects of the same class look more or less similar.

In R, we use the setClass() function to define a class.

3. Reference Class -

Reference classes are more similar to the object oriented programming we are used to seeing in other major programming languages.

Defining a reference class is similar to defining a S4 class. Instead of setClass() we use the setRefClass() function.

Comparison Between S3, S4, and Reference Class -

S3 Class	S4 Class	Reference Class
Lacks formal definition	Class defined using setClass()	Class defined using setRefClass()
Objects are created by setting the class attribute	Objects are created using new()	Objects are created using generator functions
Attributes are accessed using $	Attributes are accessed using @	Attributes are accessed using $
Methods belong to generic function	Methods belong to generic function	Methods belong to the class

13. Explain creating matrix using rbind() and cbind()?

You can either treat each vector as a row *(by using the command rbind)* or treat each vector as a column *(using the command cbind).*

rbind() -

R > rbind(1:3,4:6)

Output:-

[1,] 1 2 3

[2,] 4 5 6

cbind() -

R > cbind(c(1,4),c(2,5),c(3,6))

[1,] 1 2 3

[2,] 4 5 6

14. Explain Graphical Parameters in plot function?

1. type - Tells R how to plot the supplied coordinates.

Ex: type = "l" draws a line to connect all the points.

2. main, xlab, ylab - Options to include plot title, the horizontal axi label, and the vertical axis label, respectively

3. col - Color (or colors) to use for plotting points and lines.

4. pch - Stands for point character. This selects which character to use for plotting individual points.

5. cex - Stands for character expansion. This controls the size of plotted point characters.

6. lty - Stands for line type. This specifies the type of line to use to connect the points (for example, solid, dotted, or dashed).

7. lwd - Stands for line width. This controls the thickness of plotted lines.

8. xlim, ylim - This provides limits for the horizontal range and vertical range (respectively) of the plotting region.

UNIT - 2

R Programming Structures

1. List and explain loops in R?

1. For loop -

A for loop is used to iterate over a list, vector or any other object of elements.

Syntax -

```
for (value in sequence) {
  # block of code
  }
```

Here, **sequence** is an object of elements and **value** takes in each of those elements.

2. While loop -

The while loop in R is used to execute a set of statements in a loop as long as a condition is TRUE. It is a control statement.

while loops are used when you don't know the exact number of times a block of code is to be repeated.

Syntax -

```
while ( condition ) {
        statement
}
```

3. Repeat loop -

We use the R repeat loop to execute a code block multiple times. However, the repeat loop doesn't have any condition to terminate. You need to put an exit condition implicitly with a break statement inside the loop.

Syntax -

```
repeat {
        # statements
        if(stop_condition) {
                break
        }
```

}

2. Explain break and next statement in R?

1. Break -

A break statement is used inside a loop (repeat, for, while) to stop the iterations and flow the control outside of the loop.

2. Next -

A next statement is useful when we want to skip the current iteration of a loop without terminating it. On encountering next, the R parser skips further evaluation and starts next iteration of the loop

3. Explain looping over non-vector sets?

R does not directly support iteration over nonvector sets, but there are a couple of indirect yet easy ways to accomplish it.

Apply() Family -

1. apply() -

This function is the most basic form of implicit looping—it takes a function and applies it to each margin of an array.

Syntax -

apply(x, MARGIN, FUN,…)

Where, X is a matrix, data frame, or array. MARGIN is a vector giving the subscripts which the function will be applied over. And FUN is the function to be applied.

2. lapply() -

The lapply() function is used to apply a function to each element of the list. It collects the returned values into a list, and then returns that list.

Syntax-

lapply(x,FUN,…)

4. Explain operators in R?

1. Arithmetic operators -

These operators are used to carry out mathematical operations like addition and multiplication.

Operators - + , - , * , / , %% , %/% , ^

2. Relational operators -

Relational operators are used to compare between values. Each element of the first vector is compared with the corresponding element of the second vector. The result of comparison is a Boolean value.

Operators - >, <, ==, <=, >=, !=

3. Logical operators -

Each element of the first vector is compared with the corresponding element of the second vector. The result of comparison is a Boolean value.

Element Wise Logical operators - &, I, !

Logical operators - &&, II

4. Assignment operators -

These operators are used to assign values to vectors.

1. Left Assignment operator -

<- or = or <<-

2. Right assignment operator -

-> or ->>

5. Define function with syntax in R?

A function is a block or chunck of code having a specific structure, which is often singular or atomic nature, and can be reused to accomplish a specific nature.

Syntax -

```
function_name <- function(arguments)    {
        #statements
}
```

6. What is lazy evaluation of functions?

Arguments to functions are evaluated lazily, which means so they are evaluated only when needed by the function body.

7. What are return values?

Functions are generally used for computing some value, so they need a mechanism to supply that value back to the caller. This is called **returning**.

8. Functions are objects, explain?

R functions are first-class objects (of the class "function"), meaning that they can be used for the most part just like other objects.

This is seen in the syntax of function creation:

```
g <- function(x) {
        return(x+1)
}
```

9. Explain No pointers in R with example?

R does not have variables corresponding to pointers or references like those of, say, the C language. This can make programming more difficult in some cases.

For Example -

```
>>> x = [13,5,12]

>>> x.sort()

>>> x

[5, 12, 13]
```

Here, the value of x, the argument to sort(), changed. By contrast, here's how it works in R:

```
> x <- c(13,5,12)

> sort(x)

[1] 5 12 13

> x

[1] 13 5 12
```

The argument to sort() does not change. If we do want x to change in this R code, the solution is to reassign the arguments:

```
> x <- sort(x)

> x

[1] 5 12 13
```

10. Define recursion?

Recursion is a programming technique in which, a function calls itself repeatedly for some input

11. Write program to implement Quick sort using recursion?

```
qs <- function(x) {
        if (length(x) <= 1) return(x)
        pivot <- x[1]
        therest <- x[-1]
        sv1 <- therest[therest < pivot]
        sv2 <- therest[therest >= pivot]
        sv1 <- qs(sv1)
        sv2 <- qs(sv2)
        return(c(sv1,pivot,sv2))
}
```

UNIT - 3

Doing Math And Simulation In R

1. List and explain built-in function for math operations in R?

1. **exp() -** Exponential function, base e
2. **log() -** Natural logarithm
3. **log10() -** Logarithm base 10
4. **sqrt() -** Square root

5. **abs() -** Absolute value
6. **min() -** Minimum value within a vector
7. **max() -** Maximum value within a vector
8. **which.min() -** Index of minimal element of the vector
9. **which.max() -** Index of maximal element of the vector
10. **pmin() -** Element-wise minima of several vectors
11. **pmax() -** Element-wise maxima of several vectors
12. **sum() -** Sum of the elements of the vector
13. **prod() -** Product of the elements of the vector
14. **cumsum() -** Cumulative sum of the elements of a vector
15. **cumprod() -** Cumulative product of the elements of a vector
16. **round() -** Round of the closest integers
17. **floor() -** Round of the closest integer below
18. **ceiling() -** Round of the closes integer above
19. **factorial() -** Factorial function

2. What is cumulative sum and product ? Give example

1. Cumulative Product -

Cumulative product is a sequence of partial products of a given sequence.
Example -

```
> x <- c(2,4,3)

> cumprod(x)

[1] 2 8 24
```

2. Cumulative Sum -

A cumulative sum is a sequence of partial sum of a given sequence.

Example -

```
> x <- c(2,4,3)
> cumsum(x)
[1] 2 6 9
```

3. Explain functions for statistical distributions with their prefix names?

Prefix names as follows -

- With d for the density or probability mass function (pmf)
- With p for the cumulative distribution function (cdf)
- With q for quantiles
- With r for random number generation

Function for statistical distribution -

Distribution	Density/pmf	cdf	Quantiles	Random Numbers
Normal	dnorm()	pnorm()	qnorm()	rnorm()
Chi square	dchisq()	pchisq()	qchisq()	rchisq()
Binomial	dbinom()	pbinom()	qbinom()	rbinom()

4. List sorting functions in R?

1. sort() - Ordinary numerical sorting of a vector is done with the sort() function.

2. order() - If you want the indices of the sorted values in the original vector, use order() function.

3. rank() - This function specifies the rank of every single element present in a vector.

5. List and explain the linear algebra functions in R?

1. t() -

It used to calculate transpose of a matrix or Data Frame.

Syntax - t(x)

2. det() -

It is used to calculate the determinant of the specified matrix.

Syntax - det(x, ...)

3. diag() -

Extracts the diagonal of a square matrix (useful for obtaining variances from a covariance matrix and for constructing a diagonal matrix).

Syntax - diag(x, nrow, ncol)

4. sweep() -

It is used to apply the operation "+ or -" to the row or column in data matrix.

Syntax - sweep(x, MARGIN, STATS, FUN)

5. qr() - QR decomposition

6. chol() - Cholesky decomposition

6. Explain the set operations in R?

1. union(x,y) -

The union of two sets is defined as the set of all the elements that are members of set A, set B or both and is denoted by AUB read as A union B.

AUB = {1,2,3,4,5,a,b} U {a,b,c,d,e}

= {1,2,3,4,5,a,b,c,d,e}

2. intersect(x,y) -

The intersection of any two sets A and B is the set containing of all the elements that belong to both A and B is denoted by A∩B read as A intersection B.

A ∩ B = {1,2,3,4,5,a,b} ∩ {a,b,c,d,e}

= {a,b}

3. setdiff(x,y) -

The set difference of any two sets A and B is the set of elements that belongs to A but not B. It is denoted by A-B and read as A difference B.

A = {1,2,3,4,5,6} B = {3,5,7,9}

A-B = {1,2,4,6}

B-A = {7,9}

4. setequal(x,y) -

Test for equality between x and y. If both x and y are equal it returns TRUE otherwise returns FALSE.

Syntax: setequal(x, y)

Parameters: x and y - Objects with sequence of items

Example -

```
x1 <- c(1, 2, 3, 4, 5, 6)

x2 <- c(1:6)

x3 <- c(2, 3, 4, 5, 6)

setequal(x1, x2)

setequal(x1, x3)
```

5. c %in% y -

Membership, testing whether c is an element of the set y

6. choose(n,k) -

This function that can compute the nCr value without writing the whole code for computing nCr value.

Syntax: choose(n, r)

Parameters: n - Number of elements, r - Number of combinations

7. Explain combn() function?

The function combn() generates combinations.

Let‘s find the subsets of {1,2,3} of size 2.

```
> c32 <- combn(1:3,2)

> c32

     [,1] [,2] [,3]

[1,] 1 1 2

[2,] 2 3 3
```

UNIT - 4

Probability Distributions

1. Define probability?

Probability denotes the possibility of something happening. It is a mathematical concept that predicts how likely events are to occur.

2. What is probability distribution?

A probability distribution describes how a random variable is distributed; it tells us which values a random variable is most likely to take on and which values are less likely.

3. Explain probability distribution function with name convention?

The **probability density** function always begins with 'd'.

The **cumulative distribution** function always begins with 'p'.

The inverse **cumulative distribution (or quantile function)** always beings with 'q'.

The function that produces **random variables** always begins with 'r'.

Name	Probability Density	Cumulative Distribution	Quantile
Normal	dnorm(Z,mean,sd)	pnorm(Z,mean,sd)	qnorm(Q,mean,sd)
Poisson	dnorm(N,lambda)	pnorm(N,lambda)	qnorm(Q,lambda)
Binomial	dbinom(N,size,prob)	pbinom(N,size,prob)	qbinom(Q,size,prob)
Exponential	dexp(N,rate)	pexp(N,rate)	qexp(Q,rate)
$\chi 2$	dchisq(X,df)	pchisq(X.df)	qchisq(X,df)

4. Define Binomial distribution? List functions to generate binomial distribution

The binomial distribution is a discrete probability distribution. It describes the outcome of n independent trials in an experiment. Each trial is assumed to have only two outcomes, either success or failure.

Functions to generate binomial distribution -

dbinom(x, size, prob) :- This function gives the probability density distribution at each point.

pbinom(x, size, prob) :- This function gives the cumulative probability of an event. It is a single value representing the probability.

qbinom(p, size, prob) :- This function takes the probability value and gives a number whose cumulative value matches the probability value.

rbinom(n, size, prob) :- This function generates required number of random values of given probability from a given sample.

Following is the description of the parameters used —

1. x is a vector of numbers.
2. p is a vector of probabilities.
3. n is number of observations.
4. size is the number of trials.
5. prob is the probability of success of each trial.

5. Define cumulative binomial probability?

A cumulative binomial probability refers to the probability that the binomial random variable falls within a specified range.

6. Define poisson distribution? List functions to work with poisson distribution

A Poisson distribution is a probability distribution used in statistics to show how many times an event is likely to happen over a given period of time.

Functions to work with poisson distribution -

1. dpois(x, lambda) - Calculates the Probability Mass Function (PMF) of the Poisson distribution at a specific value of x, given a Poisson parameter lambda.

2. ppois(q, lambda) - Calculates the Cumulative Distribution Function (CDF) of the Poisson distribution at a specific value of q, given a Poisson parameter lambda.

3. qpois(p, lambda) - Calculates the Inverse Cumulative Distribution Function (quantile function) of the Poisson distribution at a specific probability value p, given a Poisson parameter lambda.

4. rpois(n, lambda) - Generates n random samples from a Poisson distribution with a Poisson parameter lambda.

7. Define normal distribution? Explain its functions

Normal Distribution is a probability function used in statistics that tells about how the data values are distributed.

Functions -

1. dnorm(x, mean = 0, sd = 1, log = FALSE) :- This function gives the probability fdensity distribution at each point.

2. pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE):- This function gives the cumulative probability of an event. It is a single value representing the probability.

3. qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE):- This function takes the probability value and gives a number whose cumulative value matches the probability value.

4. rnorm(n, mean = 0, sd = 1) :- This function generates required number of random values of given probability from a given sample.

8. Define standard normal distribution?

The simplest case of a normal distribution is known as the standard normal distribution.

9. Define Numeric variable?

A numeric variable is one whose observations are naturally recorded as numbers. There are two types of numeric variables: **continuous** and **discrete**.

10. Define continuous variable?

A continuous variable can be recorded as any value in some interval, up to any number of decimals.

11. Define discrete variable?

A discrete variable, on the other hand, may take on only distinct numeric values—and if the range is restricted, then the number of possible values is finite.

12. Define univariate and multivariate data?

1. Univariate data -

When discussing or analyzing data related to only one dimension, you're dealing with univariate data.

2. Multivariate data -

When it's necessary to consider data with respect to variables that exist in more than one dimension, they are considered multivariate.

13. Define mean?

It's considered to be the central "balance point" of a collection of observations.

Syntax:

mean(x)

14. Define median?

It is the middle value of the data set. It splits the data into two halves. If the number of elements in the data set is odd then the center element is median and if it is even then the median would be the average of two central elements.

Syntax:

```
median(x, na.rm = FALSE)
```

15. Define mode?

The mode is the value that has highest number of occurrences in a set of data. Unike mean and median, mode can have both numeric and character data.

16. Define Quantile?

A quantile is a value computed from a collection of numeric measurements that indicates an observation's rank when compared to all the other present observations.

17. What percentile?

Quantiles can be expressed as a percentile—this is identical but on a "percent scale" of 0 to 100.

18. Define covariance?

The covariance expresses how much two numeric variables "change together" and the nature of that relationship, whether it is positive or negative.

19. Define ANOVA?

Analysis of variance (ANOVA), in its simplest form, is used to compare multiple means in a test for equivalence.

20. What is one-way ANOVA ?

The one-way ANOVA is used to test two or more means for equality. Those means are split by a categorical group or factor variable.

21. What is two-way ANOVA?

When there are two categorical independent variables (factors) and one continuous dependent variable, two-way ANOVA is used as an extension of one-way ANOVA.

UNIT - 5

Simpler Linear & Non-Linear Regression

1. What is regression?

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

2. Define Linear regression? Give general mathematical equation for a linear regression?

Linear regression is used to predict the value of an outcome variable y on the basis of one or more input predictor variables x.

y = ax + b

Where,

- y is the response variable.
- x is the predictor variable.

- a and b are constants which are called the coefficients

3. Define Linear regression line?

A linear line showing the relationship between the dependent and independent variables is called a regression line.

4. Define positive linear relationship?

If the dependent variable increases on the Y-axis and the independent variable increases on the X-axis, then such a relationship is termed as a Positive linear relationship.

5. Define negative linear relationship?

If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.

6. What is lm() function? Write syntax?

This function creates the relationship model between the predictor and the response variable.

Syntax -

lm(formula,data)

where,

- **formula** is a symbol presenting the relation between x and y.
- **data** is the vector on which the formula will be applied.

7. Explain predict() function?

This function is used to predict the value.

Syntax -

predict(object, newdata)

where,

- **object** is the formula which is already created using the lm() function.
- **newdata** is the vector containing the new value for predictor variable.

8. List advantages/Limitation of linear regression model?

- Linear regression implements a statistical model that, when relationships between the independent variables and the dependent variable are almost linear, shows optimal results.
- Linear regression is often inappropriately used to model non-linear relationships.
- Linear regression is limited to predicting numeric output.
- A lack of explanation about what has been learned can be a problem.

9. What is multiple linear regression?

Multiple Linear Regression basically describes how a single response variable Y depends linearly on a number of predictor variables.

10. How to manually open a new device?

The typical base R commands such as plot, hist, boxplot, and so on will automatically open a device for plotting and draw the desired plot, if nothing is currently open.

We can also open new device windows using **dev.new;**

11. How to switch between devices ?

By using dev.set() we can switch between devices.

12. How to close a device?

By using dev.off() we can close a device.

13. Write a R program for any visual representation of an object with creating graphs using?

```
# graphic functions: Plot(),Hist(),Linechart(),Pie(),Boxplot(),Scatterplots().

x <- c(1, 2, 3, 4, 5)

y <- c(3, 5, 7, 2, 8)

dev.new()


# Create a plot

plot(x, y, type = "o", main = "Line Chart", xlab = "X-axis", ylab = "Y-axis", col = "blue")

# Create a new graphics device

dev.new()


# Create a histogram

hist(x, main = "Histogram", xlab = "Value", ylab = "Frequency", col = "lightgreen")

# Create a new graphics device

dev.new()


# Create a line chart

lines(x, y, type = "o", col = "red")

title(main = "Line Chart", xlab = "X-axis", ylab = "Y-axis")

# Create a new graphics device

dev.new()


# Create a pie chart
```

```
pie(x, labels = c("A", "B", "C", "D", "E"), main = "Pie Chart")

# Create a new graphics device

dev.new()


# Create a boxplot

boxplot(x, main = "Boxplot", ylab = "Values", col = "orange")

# Create a new graphics device

dev.new()


# Create a scatterplot

plot(x, y, main = "Scatterplot", xlab = "X-axis", ylab = "Yaxis", col = "green", pch = 19)
```