What is R and Why use R for statistical work?

R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team, of which Chambers is a member. R is named partly after the first names of the first two R authors and partly as a play on the name of S.

R's wide popularity is because of its ability to perform simple and complex mathematical and statistical calculations. It is also used for analyzing data in many industries.

The official R software environment is an open-source free software environment released as part of the GNU Project and available under the GNU General Public License. It is written primarily in C, Fortran, and R itself

Good for data visualization

R has a wide range of powerful libraries and tools for creating high-quality, interactive visualizations of complex data sets. R's ggplot2 library is a popular choice for creating sophisticated, customizable data visualizations, and it has a large and active community of users and contributors.

Data Cleaning in R with the Janitor Package. it's an R package that has simple functions for examining and cleaning dirty data. It can format data frame column names, isolate duplicate and partially duplicate records, isolate empty and constant data, and much more!

Installation of R

R Studio is an integrated development environment (IDE) for R. IDE is a GUI, where you can write your quotes, see the results and also see the variables that are generated during the course of programming.

- **R Studio is available as both Open source and Commercial software.**
- **R Studio is also available as both Desktop and Server versions.**
- **R Studio is also available for various platforms such as Windows, Linux, and macOS.**

Rstudio is an open-source tool that provides Ide to use R language, and enterprise-ready professional software for data science teams to develop share the work with their team.

Install R in Windows

1. Download the R setup from https://cloud.r-project.org/bin/windows/base/.
2. In the next step, we have to select either customized startup or accept the default, and then we proceed to Next.
3. In the last, we will click on finish to successfully install R in our system.

R Advantages and Disadvantages

R is the most popular programming language for statistical modelling and analysis. Like other programming languages, R also has some advantages and disadvantages.

Advantages

1) Open Source

An open-source language is a language on which we can work without any need for a license or a fee. R is an open-source language. We can contribute to the development of R by optimizing our packages, developing new ones, and resolving issues.

2) Platform Independent

R is a platform-independent language or cross-platform programming language which means its code can run on all operating systems. R enables programmers to develop software for several competing platforms by writing a program only once. R can run quite easily on Windows, Linux, and Mac.

3/Machine Learning Operation

R allows us to do various machine learning operations such as classification and regression. For this purpose, R provides various packages and features for developing the artificial neural network. R is used by the best data scientists in the world.

4) Exemplary support for data wrangling

R allows us to perform data wrangling. R provides packages such as dplyr, readr which are capable of transforming messy data into a structured form.

5) Quality plotting and graphing

R simplifies quality plotting and graphing. R libraries such as ggplot2 and plotly advocates for visually appealing and aesthetic graphs which set R apart from other programming languages.

6) The array of packages

R has a rich set of packages. R has over 10,000 packages in the CRAN repository which are constantly growing. R provides packages for data science and machine learning operations.

7) Statistics

R is mainly known as the language of statistics. It is the main reason why R is predominant than other programming languages for the development of statistical tools.

8) Continuously Growing

R is a constantly evolving programming language. Constantly evolving means when something evolves, it changes or develops over time, like our taste in music and clothes, which evolve as we get older. R is a state of the art which provides updates whenever any new feature is added.

Disadvantage

1) Data Handling

In R, objects are stored in physical memory. It is in contrast with other programming languages like Python. R utilizes more memory as compared to Python. It requires the entire data in one single place which is in the memory. It is not an ideal option when we deal with Big Data.

2) Basic Security

R lacks basic security. It is an essential part of most programming languages such as Python. Because of this, there are many restrictions with R as it cannot be embedded in a web-application.

3) Complicated Language

R is a very complicated language, and it has a steep learning curve. The people who don't have prior knowledge or programming experience may find it difficult to learn R.

4) Weak Origin

The main disadvantage of R is, it does not have support for dynamic or 3D graphics. The reason behind this is its origin. It shares its origin with a much older programming language "S."

5) Lesser Speed

R programming language is much slower than other programming languages such as MATLAB and Python. In comparison to other programming language, R packages are much slower.

1.Print first line in R

```
> myString <- "Hello, Delhi!"
> print ( myString)
[1] "Hello, Delhi!"
```

2. Comments

Comments are like helping text in your R program and they are ignored by the interpreter while executing your actual program.

 # My first program in R Programming

R does not support multi-line comments but you can perform a trick which is something as follows –

if(FALSE) {

"This is a demo for multi-line comments and it should be put inside either a single OR double quote"

}

myString <- "Hello, World!"

print (myString)

R - Data Types

Data Objects- Data Types & Data Structures (e.g. lists. Arrays, matrices, data frames)

like C and java in R, the variables are not declared as some data type. The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable. There are some key data types are:

- Vectors
- Lists
- Matrices
- Arrays
- Factors
- Data Frames

vector object is the simplest object type and there are six data types of these atomic vectors, also termed as six classes of vectors. The other R-Objects are built upon the atomic vectors.

to create vector with more than one element, you should use **c()** function which means to combine the elements into a vector.

 # Create a vector.

```
apple <- c('red','green',"yellow")
print(apple)
# Get the class of the vector.
print(class(apple))
```

2. Lists

A list is an R-object which can contain many different types of elements inside it like vectors, functions and even another list inside it.

```
# Create a list.
list1 <- list(c(2,5,3),21.3,sin)

# Print the list.
print(list1)
```

3.Matrices

A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

```
# Create a matrix.
M = matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE)
print(M)
```

4.Arrays

While matrices are confined to two dimensions, arrays can be of any number of dimensions. The array function takes a dim attribute which creates the required number of dimension.

```
# Create an array.
a <- array(c('green','yellow'),dim = c(3,3,2))
print(a)
```

5.Factors

Factors are the r-objects which are created using a vector. It stores the vector along with the distinct values of the elements in the vector as labels. The labels are always character irrespective of whether it is numeric or character or Boolean etc. in the input vector. They are useful in statistical modeling.

Factors are created using the **factor()** function.

```
# Create a vector.
apple_colors <- c('green','green','yellow','red','red','red','green')

# Create a factor object.
factor_apple <- factor(apple_colors)

# Print the factor.
print(factor_apple)
print(nlevels(factor_apple))
```

6. **Data Frames**

Data frames are tabular data objects. Unlike a matrix in data frame each column can contain different modes of data. The first column can be numeric while the second column can be character and third column can be logical. It is a list of vectors of equal length.

Data Frames are created using the **data.frame()** function.

```r
# Create the data frame.
BMI <-    data.frame(
  gender = c("Male", "Male","Female"),
  height = c(152, 171.5, 165),
  weight = c(81,93, 78),
  Age = c(42,38,26)
)
print(BMI)
```

R - Variables

A variable provides us with named storage that our programs can manipulate.

- **A valid variable name consists of letters, numbers and the dot or underline characters.**
- **The variable name starts with a letter or the dot not followed by a number.**

Variable Name	Validity	Reason
var_name2.	valid	Has letters, numbers, dot and underscore
var_name%	Invalid	Has the character '%'. Only dot(.) and underscore allowed.
2var_name	invalid	Starts with a number
.var_name, var.name	valid	Can start with a dot(.) but the dot(.)should not be followed by a number.
.2var_name	invalid	The starting dot is followed by a number making it invalid.
_var_name	invalid	Starts with _ which is not valid

Basic Mathematical & Arithmetic operations in R

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. R language is rich in built-in operators and provides following types of operators.

- **Arithmetic Operators**
- **Relational Operators**
- **Logical Operators**
- **Assignment Operators**
- **Miscellaneous Operators**

Miscellaneous Operators

These operators are used to for specific purpose and not general mathematical or logical computation.

Operator Description Example

Colon operator. It creates the series of numbers in sequence for a vector.

v <- 2:8

print(v)

%in% :This operator is used to identify if an element belongs to a vector.

v1 <- 8

v2 <- 12

t <- 1:10

print(v1 %in% t)

print(v2 %in% t)

%*%: This operator is used to multiply a matrix with its transpose.

M = matrix(c(2,6,5,1,10,4), nrow = 2,ncol = 3,byrow = TRUE)

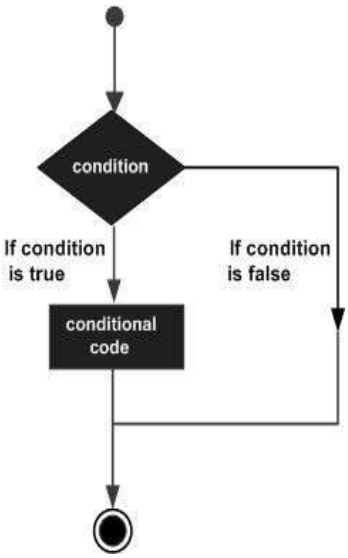
t = M %*% t(M)

print(t)

R - Decision making

Decision making structures require the programmer to specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be **true**, and optionally, other statements to be executed if the condition is determined to be **false**.

R provides the following types of decision making statements.



Sr.No.	Statement & Description
1	**if statement** An **if** statement consists of a Boolean expression followed by more statements.
2	**if...else statement** An **if** statement can be followed by an optional **else** statemer

which executes when the Boolean expression is false.

	switch statement	
3		A **switch** statement allows a variable to be tested for equality against a list of values.

R – Loops

A loop statement allows us to execute a statement or group of statements multiple times and the following is the general form of a loop statement in most of the programming languages. R programming language provides the following kinds of loop to handle looping requirements.

Sr.No.	**Loop Type & Description**
	repeat loop
1	Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
	while loop
2	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
	for loop
3	Like a while statement, except that it tests the condition at the end of the loop body.

R – Functions

A function is a set of statements organized together to perform a specific task. R has a large number of in-built functions and the user can create their own functions.

Function Definition

An R function is created by using the keyword **function**.

function_name <- function(arg_1, arg_2, ...) {

Function body

}

Function Components: The different parts of a function are −

Function Name − This is the actual name of the function. It is stored in R environment as an object with this name.

Arguments – An argument is a placeholder. When a function is invoked, you pass a value to the argument. Arguments are optional; that is, a function may contain no arguments. Also arguments can have default values.

Function Body – The function body contains a collection of statements that defines what the function does.

Return Value – The return value of a function is the last expression in the function body to be evaluated.

Built-in Function

in-built functions are seq(), mean(), max(), sum(x) and paste(...) etc. They are directly called by user written programs.

```
# Create a sequence of numbers from 32 to 44.
print(seq(32,44))
# Find mean of numbers from 25 to 82.
print(mean(25:82))
# Find sum of numbers frm 41 to 68.
print(sum(41:68))
```

User-defined Function

Create a function to print squares of numbers in sequence.

```
new.function <- function(a) {

  for(i in 1:a) {

    b <- i^2

    print(b)

  }

}
```

```
# Call the function new.function supplying 6 as an argument.
new.function(6)
```

Calling a Function with Default Argument

```
# Create a function with arguments.
new.function <- function(a = 3, b = 6) {
  result <- a * b
  print(result)
}# Call the function without giving any argument.
new.function()

# Call the function with giving new values of the argument.
new.function(9,5)
```

Packages in R

Packages in R Programming language are a set of R functions, compiled code, and sample data. These are stored under a directory called “library” within the R environment. By default, R installs a group of packages during installation. Once we start the R console, only the default packages are available by default.

Check Available R Packages

```
.libPaths()
library()
```

What are Repositories?

A repository is a place where packages are located and stored so you can install R packages from it. All the packages available in R language are listed at **R Packages.**

Organizations and Developers have a local repository, typically they are online and accessible to everyone. Some of the most popular repositories for R packages are:

CRAN: Comprehensive R Archive Network(CRAN) is the official repository, it is a network of FTP and web servers maintained by the R community around the world. The R community coordinates it, and for a package to be published in CRAN, the Package needs to pass several tests to ensure that the package is following CRAN policies.

Bioconductor: Bioconductor is a topic-specific repository, intended for open source software for bioinformatics. Similar to CRAN, it has its own submission and review processes, and its community is very active having several conferences and meetings per year in order to maintain quality.

Github: Github is the most popular repository for open-source projects. It’s popular as it comes from the unlimited space for open source, the integration with git, a version control software, and its ease to share and collaborate with others.

Install an R-Packages

Installing R Packages From CRAN: For installing R Package from CRAN we need the name of the package and use the following command:

installing Package from CRAN is the most common and easiest way as we just have to use only one command. In order to install more than a package at a time, we just have to write them as a character vector in the first argument of the **install.packages()** function:

install.packages(c("vioplot", "MASS"))

- **Installing BiocManager Packages:** In Bioconductor,using R version 3.5 or greater, which is not compatible with the **biocLite.R** script for installing Bioconductor packages.

  install.packages("BiocManager")

  install.packages("Package Name")

  # Install the package named "XML".

  install.packages("XML")

- This will install some basic functions which are needed to install bioconductor packages, such as the **biocLite()** function. To install the core packages of Bioconductor just type it without further arguments:

  BiocManager::install("edgeR")

-

Update, Remove and Check Installed Packages in R : To check what packages are installed on your computer, type this command:

installed.packages()

To update all the packages, type this command:

update.packages()

To update a specific package, type this command:

install.packages("PACKAGE NAME")

Difference Between a Package and a Library

There is always confusion between a package and a library, and we find people calling libraries as packages.

- **library():** It is the command used to load a package, and it refers to the place where the package is contained, usually a folder on our computer.
- **Package:** It is a collection of functions bundled conveniently. The package is an appropriate way to organize our own work and share it with others.