BHARATHIDASAN UNIVERSITY

Tiruchirappalli- 620024

Tamil Nadu, India.

Programme: M.Sc. Statistics

Course Title: R Programming

Course Code: 23ST05CC

Unit-I

Introduction to R

Dr. T. Jai Sankar
Associate Professor and Head
Department of Statistics

Ms. I. Angel Agnes Mary
Guest Faculty
Department of Statistics

UNIT – I

Introduction to R

Introduction to R

- R is an open-source programming language and environment used for statistical analysis, data visualization, and data science.
- Being open-source, R has a massive community that continuously works to improve the environment as well as helps members worldwide to improve and innovate.
- R can be used for data analytics, statistical analysis, as well as machine learning purposes.
- R is compatible with a number of different technologies and is highly flexible.
- It has over 10,000 different libraries and packages to enhance and add on to its already significant capabilities. It also has graphics libraries for static as well as dynamic graphics.

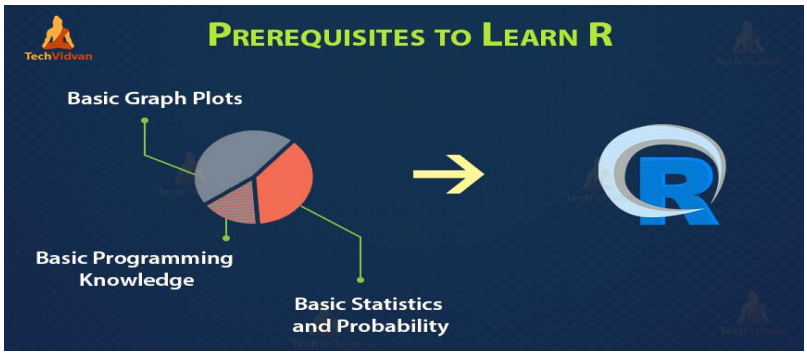
History of R:

- R is a programming language it was an implementation over S-Programming language. R was first designed by **Ross Ihaka and Robert Gentleman** at the **University of Auckland** in **1993**.
- A large group of individuals has contributed to R by sending code and bug reports.
- Since mid-1997 there has been a core group (the "R Core Team") who can modify the R source code archive.
- It was stable released on October 31st 2014 the 4 months ago, by R Development Core Team under GNU General Public License.

Before Start Learning R

There are no mandatory prerequisites to R. But before you jump into learning R, it is recommended to have some basic knowledge of a few topics. These include:

- Basic understanding of statistics, mathematics, and probability
- General understanding of data science and the processes involved.
- Basic Understanding of various types of graphs and data representation techniques.



Features of R

R is packed with many exciting features and limitless possibilities. Some key features of the R programming language:

- R has a massive community that works tirelessly to improve and add upon R's abilities. CRAN or Comprehensive R Archive Network has over 10,000 packages or extensions that can be used from producing high-definition graphics to creating interactive web-apps.
- R can perform complex mathematical and statistical operations on vectors, matrices, data frames, arrays, and other data objects of varying sizes.
- R is an interpreted language and does not need a compiler. It generates a machine-independent code that is easy to debug and is highly portable.
- R is a comprehensive programming language that supports object-oriented as well as procedural programming with generic and first-class functions.

- It supports matrix arithmetic.
- R can present data graphically. With static graphics, producing production quality visualizations and extended libraries providing interactive graphic capabilities, data visualization, and data representation becomes very easy. From concise charts to elaborate flow diagrams, all are well within R's repertoire.
- R can be used throughout the data analysis process. It helps to gather the data, to clean it, to investigate it, to model it, and finally, it helps you to compile the results in an eye-catching and easy to understand reports with R markdown. R can also help you build apps to show the results to the world.
- It can use distributed computing to process large datasets parallelly.
- R has packages that allow it to interact with multiple databases of different formats. It can also interact with various database management systems.
- R is compatible with many other programming languages like C, C++, Java, Python, etc.

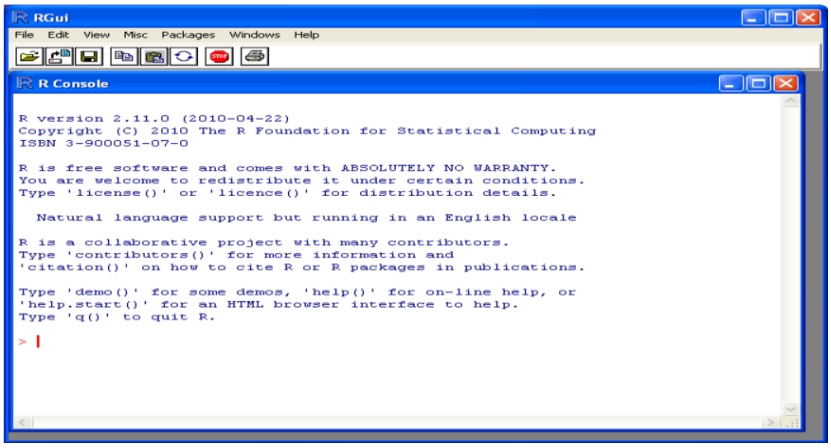
Installing R

Installing R on Windows is very straightforward. The easiest way is to install it through CRAN, which stands for The Comprehensive R Archive Network. Website: https://www.rcran.com/products/rcran/ download/

- Under "Download and Install R", click on the "Windows" link.
- Under "Subdirectories", click on the "base" link.
- On the next page, you should see a link saying something like "Download R 2.10.1 for Windows" (or R X.X.X, where X.X.X gives the version of R, eg. R 2.11.1). Click on this link.

- You may be asked if you want to save or run a file "R-2.10.1-win32.exe". Choose "Save" and save the file on the Desktop. Then double-click on the icon for the file to run it.
- You will be asked what language to install it in - choose English.
- The R Setup Wizard will appear in a window. Click "Next" at the bottom of the R Setup wizard window.
- Click Next.. Next.. Finish.
- Download Complete.

To Start R, click on its desktop icon or use 'search windows' to access the program.



Installing R packages

R comes with some standard packages that are installed when you install R. However, in this booklet I will also tell you how to use some additional R packages that are useful, for example, the "rmeta" package. These additional packages do not come with the standard installation of R, so you need to install them yourself.

How to install an R package

Once you have installed R on a Windows computer (following the steps above), you can install an additional package by following the steps below:

- To start R, follow either step 2 or 3:

- Check if there is an “R” icon on the desktop of the computer that you are using. If so, double-click on the “R” icon to start R. If you cannot find an “R” icon, try step 3 instead.

- Click on the “Start” button at the bottom left of your computer screen, and then choose “All programs”, and start R by selecting “R” (or R X.X.X, where X.X.X gives the version of R, eg. R 2.10.0) from the menu of programs.

- The R console (a rectangle) should pop up.

- Once you have started R, you can now install an R package (eg. the “rmeta” package) by choosing “Install package(s)” from the “Packages” menu at the top of the R console. This will ask you what website you want to download the package from, you should choose “Ireland” (or another country, if you prefer). It will also bring up a list of available packages that you can install, and you should choose the package that you want to install from that list (eg. “rmeta”).

- This will install the “rmeta” package.

- The “rmeta” package is now installed. Whenever you want to use the “rmeta” package after this, after starting R, you first have to load the package by typing into the R console: > **library**("rmeta")

Note that there are some additional R packages for bioinformatics that are part of a special set of R packages called Bioconductor (www.bioconductor.org) such as the “yeastExpData” R package, the “Biostrings” R package, etc.). These Bioconductor packages need to be installed using a different, Bioconductor-specific procedure (see How to install a Bioconductor R package below).

How to install a Bioconductor R package

The procedure above can be used to install the majority of R packages. However, the Bioconductor set of bioinformatics R packages need to be installed by a special procedure. Bioconductor (www.bioconductor.org) is a group of R packages that have been developed for bioinformatics. This includes R packages such as “yeastExpData”, “Biostrings”, etc.

To install the Bioconductor packages, follow these steps:

- To start R, follow either step 2 or 3:
- Check if there is an "R" icon on the desktop of the computer that you are using. If so, double-click on the "R" icon to start R. If you cannot find an "R" icon, try step 3 instead.
- Click on the "Start" button at the bottom left of your computer screen, and then choose "All programs", and start R by selecting "R" (or R X.X.X, where X.X.X gives the version of R, eg. R 2.10.0) from the menu of programs.
- The R console (a rectangle) should pop up.
- Once you have started R, now type in the R console:

  > **source**("http://bioconductor.org/biocLite.R")

  > biocLite()

- This will install a core set of Bioconductor packages ("affy", "affydata", "affyPLM", "annaffy", "annotate", "Biobase", "Biostrings", "DynDoc", "gcrma", "genefilter", "geneplotter", "hgu95av2.db", "limma", "marray", "matchprobes", "multtest", "ROC", "vsn", "xtable", "affyQCReport"). This takes a few minutes (eg. 10 minutes).
- At a later date, you may wish to install some extra Bioconductor packages that do not belong to the core set of Bioconductor packages. For example, to install the Bioconductor package called "yeastExpData", start R and type in the R console:

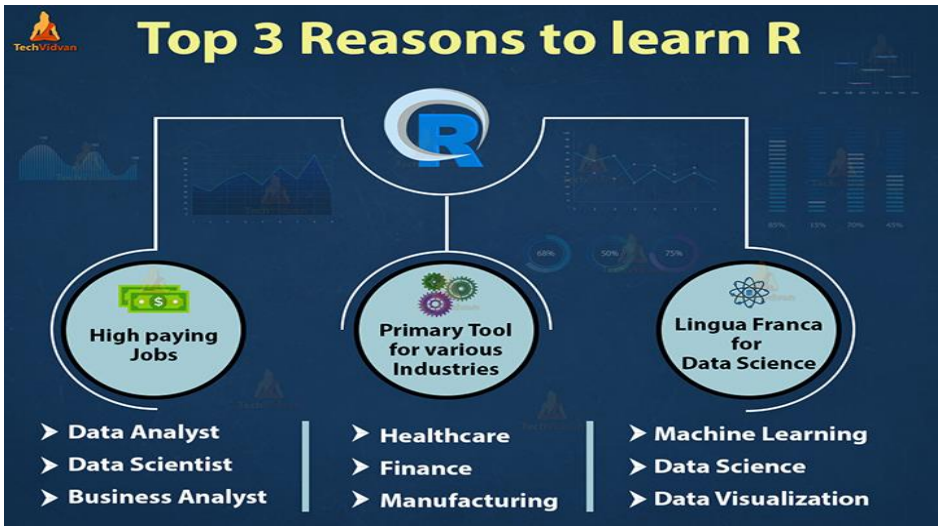
  > **source**("http://bioconductor.org/biocLite.R")

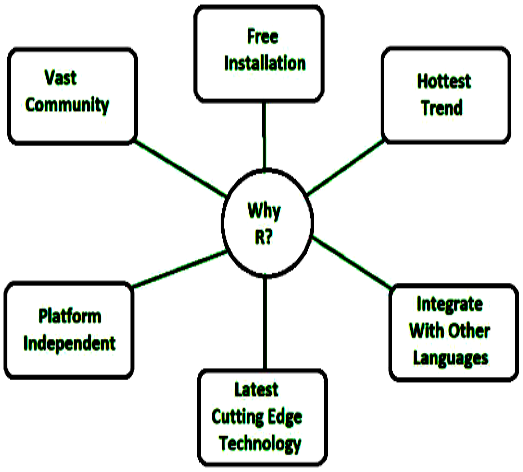
  > biocLite("yeastExpData")

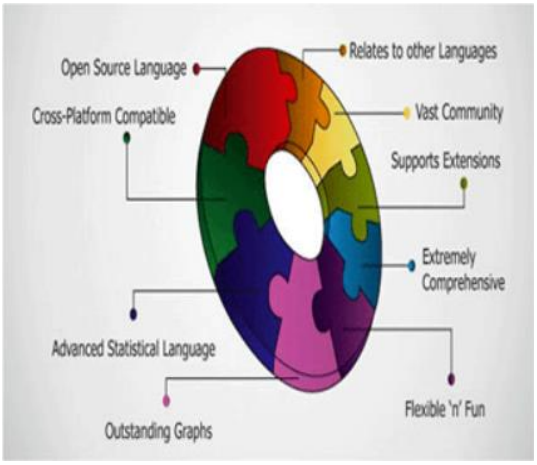
- Whenever you want to use a package after installing it, you need to load it into R by typing:

  > **library**("yeastExpData")

Why Learn R?







Here are a few **reasons** why learning R is a must for a data scientist:

- It is the **standard tool** for performing data analysis and statistical operations.
- R has **strong data visualization** capabilities. It can make production quality graphics, which makes presenting data in an **easy** to understand way possible.
- R makes data gathering processes like web scraping very easy.
- It is **platform-independent**, which means it can be used across all operating systems.
- Several industries like health, finance, banking, e-commerce, manufacturing, and much more use R as their **primary too**l for data modeling.

- It integrates seamlessly with other technologies like Hadoop, which makes an ideal combination for large scale data processing.
- R can be used for **data analysis**, **statistical analysis**, as well as **machine learning**.
- R is **free** due to its open-source GNU licensing and can be installed and used by anyone.
- With a large number of packages and libraries available for it, R is one of the most flexible programming languages in the world.
- Academic research, healthcare industry, government surveys, finance industry, banking industry, retail and manufacturing sectors, e-commerce companies as well as tech giants like **Facebook**, **Google**, and **Amazon** all use R for some purpose or the other and, therefore, **need R programmers**.

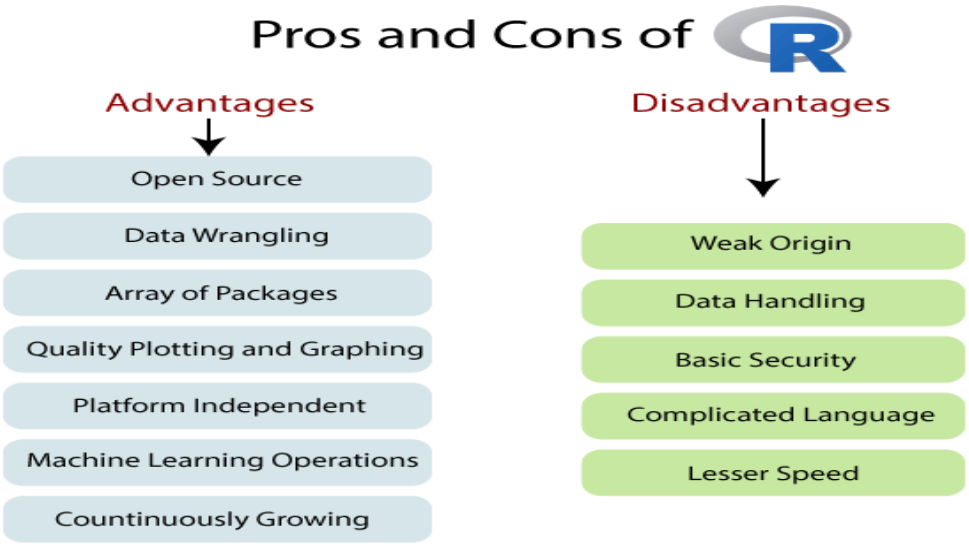
R vs the World

R is not the only programming language dealing with data science and machine learning. Here are a few advantages R has over the others like SAS:

- **Open source:** R is an open-source environment. It is cost-effective for projects of any size and is widely available.
- **Advanced graphics:** R has various libraries and packages available for plotting attractive and elegant graphs. These can also be used to create highly interactive graphics for data-driven storytelling, as well.
- **Data handling and storage:** R's various packages allow for very robust data handling and storage.
- **Options and alternatives:** Due to the large number of packages available for R, there are always several alternative ways to solve a problem.
- **Community support:** R has more than 2 million users worldwide. Due to its popularity, R users enjoy great community support.
- **R markdown:** Make attractive and concise reports of your findings and results using R markdown, which seamlessly combines plain text with code and visualization graphics.

- **Compatible with various other technologies:** R can integrate with a number of different technologies and programming languages.
- **Cross-platform compatible:** R can run on any OS, in any software environment, without any modifications and compatibility issues.

Advantages and Disadvantages of R Programming



Advantages of R Programming

1. Open Source

R is an open-source programming language. This means that anyone can work with R without any need for a license or a fee. Furthermore, you can contribute towards the development of R by customizing its packages, developing new ones and resolving issues.

2. Exemplary Support for Data Wrangling

R provides exemplary support for data wrangling. The packages like dplyr, readr are capable of transforming messy data into a structured form.

3. The Array of Packages

R has a vast array of packages. With over 10,000 packages in the CRAN repository, the number is constantly growing. These packages appeal to all the areas of industry.

4. Quality Plotting and Graphing

R facilitates quality plotting and graphing. The popular libraries like ggplot2 and plotly advocate for aesthetic and visually appealing graphs that set R apart from other programming languages.

5. Highly Compatible

R is highly compatible and can be paired with many other programming languages like C, C++, Java, and Python. It can also be integrated with technologies like Hadoop and various other database management systems as well.

6. Platform Independent

R is a platform-independent language. It is a cross-platform programming language, meaning that it can be run quite easily on Windows, Linux, and Mac.

7. Eye-Catching Reports

With packages like Shiny and Markdown, reporting the results of an analysis is extremely easy with R. You can make reports with the data, plots and R scripts embedded in them.

8. Machine Learning Operations

R provides various facilities for carrying out machine learning operations like classification, regression and also provides features for developing artificial neural networks.

9. Statistics

R is prominently known as the lingua franca of statistics. This is the main reason as to why R is dominant among other programming languages for developing statistical tools.

10. Continuously Growing

R is a constantly evolving programming language. It is a state of the art technology that provides updates whenever any new feature is added.

Disadvantages of R Programming

1. Weak Origin

R shares its origin with a much older programming language "S". This means that it's base package does not have support for dynamic or 3D graphics. With common packages of R like Ggplot2 and Plotly, it is possible to create dynamic, 3D as well as animated graphics.

2. Data Handling

In R, the physical memory stores the objects. This is in contrast to other languages like Python. Furthermore, R utilizes more memory as compared with Python. Also, R requires the entire data in one single place, that is, in the memory. Therefore, it is not an ideal option when dealing with Big Data. However, with data management packages and integration with Hadoop possible, this is easily covered.

3. Basic Security

R lacks basic security. This feature is an essential part of most programming languages like Python. Because of this, there are several restrictions with R as it cannot be embedded into a web-application.

4. Complicated Language

R is not an easy language to learn. It has a steep learning curve. Due to this, people who do not have prior programming experience may find it difficult to learn R.

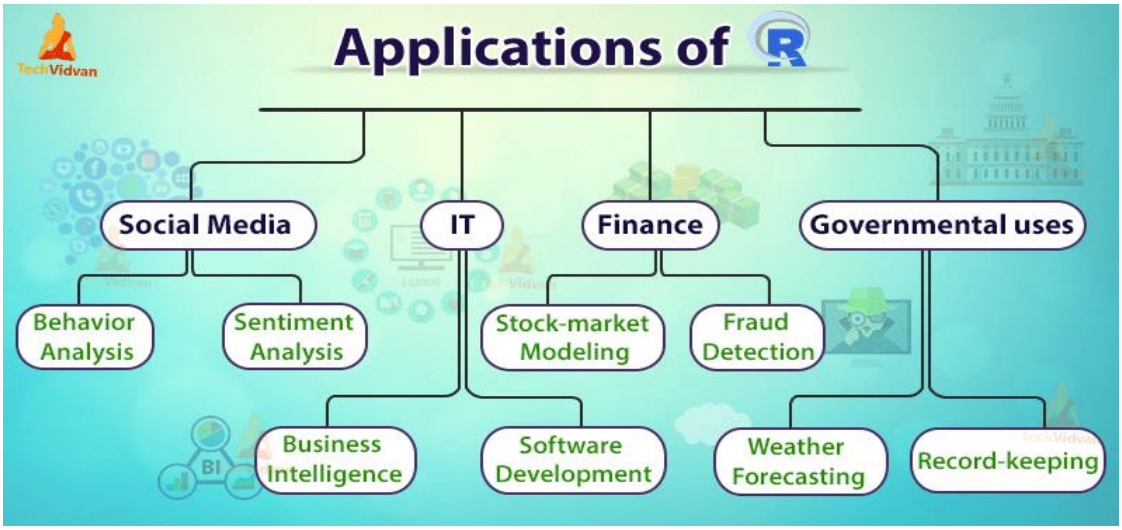
5. Lesser Speed

R packages and the R programming language is much slower than other languages like *MATLAB* and *Python*.

6. Spread across various Packages

The algorithms in R are spread across different packages. Programmers without prior knowledge of packages may find it difficult to implement algorithms.

Applications of R:



Some of the applications of R are given below:

- R is used for Data Science. It offers us a wide range of statistics-related libraries. Additionally, it offers a setting for statistical computation and design.
- Many quantitative analysts utilize R as a programming language. As a result, it aids in data import and cleansing.
- In environmental science, R is used to analyze and simulate environmental data, climate data, and ecological data.
- The most common language is R. It is used by a large number of data analysts and research programmers. As a result, it is employed as a fundamental financial instrument.

Job Profiles in R

There are around **1 million** job openings for R programmers world. Job portals like LinkedIn, Glassdoor, Indeed, Monster, etc. all have thousands of job listings for data scientists and R programmers.

R programmers can find jobs in various companies and firms in **roles** such as:

- **Data Scientist**: A data scientist's job entails collecting data, transforming it into the desired format, analyzing it and then drawing insights from it. They analyze customer behaviour and identify revenue opportunities. They also develop statistical models for data analysis. You need experience of 1-5yrs for a job as a data scientist. The average yearly income of data scientists in the United States is around **$375,000**
- **Data Analyst**: A data analyst works with IT and management teams to determine an organization's business goal. They gather data, clean it, and analyze it to pinpoint trends and patterns. On average, the experience required for a job as a data analyst is 1-3yrs. The annual income of an entry-level data analyst in the United States is **$95,000**
- **Business Analyst**: A business analyst has to come up with technical solutions for business problems. They identify business needs and develop strategies to meet them. They evaluate data from various sources and communicate complex data in easy-to-understand ways. To become a business analyst, you need 3+ yrs of experience in data science and strong business-knowledge. The average annual income of an entry-level business analyst in the United States is **$97,000**
- **Business Intelligence Expert**: A business intelligence expert extracts data, analyzes it, visualize it and then helps with the business strategy and business decision-making process. Experience of 2-4 yrs in data science, as well as knowledge of business strategy, is required for a job as a business intelligence expert. In the United States, the average yearly income of a business intelligence expert in **$65,000**
- **Data Visualization Expert**: A data visualization expert helps to convert the results of the analysis into easy-to-read reports with graphs and charts instead of raw data. They take data and turn it into insights. You need strong visualization skills and knowledge of designing. It takes the experience of 1-5 yrs to reach a good level at this job. The average yearly income of an entry-level data visualization expert in the United States is **$112,000**
- **Quantitative Analyst**: A quantitative analyst is someone who has a good knowledge of finance apart from the technical skills to be a data scientist. They develop and implement complex mathematical models to help organizations make decisions about risk management, investments, and pricing.

Comparison with other languages:

R	Python	JAVA
It was stably released in 2014.	It was stably released in 1996.	It was stably released in 1995.
It has more functions and packages.	It has less functions and packages.	It has large number of inbuilt functions and packages.
It is an interpreter base language	It is an interpreter base language	It is interpreter and compiled based language.
It is statistical design and graphics programming language.	It is general purpose language.	It is general purpose Programming language designed for web applications.
It is difficult to learn and understand.	It is easy to understand.	It is easy to learn and understand.
R is mostly use for data analysis.	Generic programming tasks such as design of software's or desktop applications.	Java is mostly used in design of windows applications and web applications.

Basic Syntax of R

1. R Command Prompt

Once you have R environment setup, then it's easy to start your R command prompt by just typing the following command at your command prompt:

```
$ R
```

This will launch R interpreter and you will get a prompt > where you can start typing your program as follows:

```
> myString <- "Hello, World!"

> print ( myString)

[1] "Hello, World!"
```

Here first statement defines a string variable myString, where we assign a string "Hello, World!" and then next statement print() is being used to print the value stored in variable myString.

2. R Script File

Usually, you will do your programming by writing your programs in script files and then you execute those scripts at your command prompt with the help of R interpreter called Rscript. So let's start with writing following code in a text file called test.R as under:

```
# My first program in R Programming

myString <- "Hello, World!"

myString
```

When we run the above program, it produces the following result.

```
[1] "Hello, World!"
```

3. Comments

Comments are like helping text in your R program and they are ignored by the interpreter while executing your actual program. Single comment is written using # in the beginning of the statement as follows:

```
# 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 of double quote" }

myString <- "Hello, World!"

myString
```

Though above comments will be executed by R interpreter, they will not interfere with your actual program. You should put such comments inside, either single or double quote.

Data Workflow

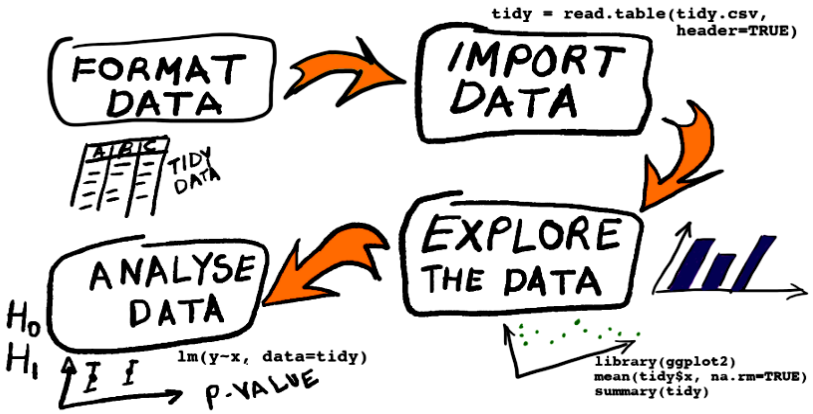
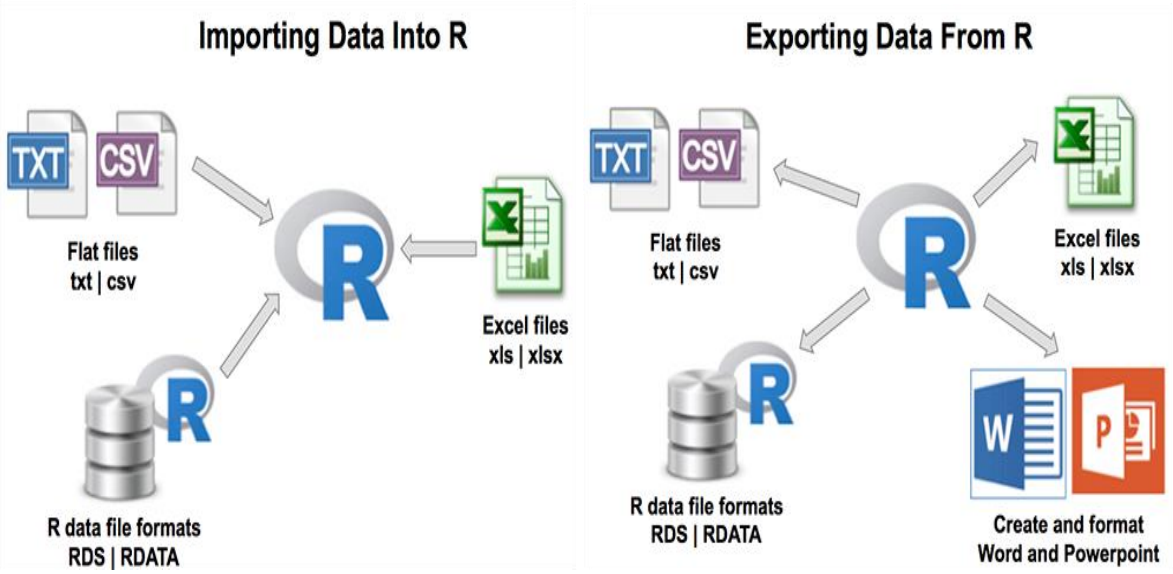


Figure: The workflow to follow when handling data.

Import and Export Data

There are built-in functions in R for the data import and export of many common file types, such as read. table( ) for . txt files, read. csv( ) for .

- read.table( )
    - reads in data from an external file
- data.entry( )
    - create object first, then enter data
- c( )
    - concatenate
- scan( )
    - prompted data entry
- R has ODBC for connecting to other programs



Data Types:

Generally, while doing programming in any programming language, you need to use various variables to store various information. Variables are nothing but reserved memory locations to store values. This means that, when you create a variable you reserve some space in memory.

You may like to store information of various data types like character, wide character, integer, floating point, double floating point, Boolean etc. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.

In contrast to other programming languages 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 many types of R-objects. The frequently used ones are:

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

The simplest of these objects is the **vector object** 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.

Data Type	Example	Verify
Logical	TRUE , FALSE	v <- TRUE print(class(v)) it produces the following result: [1] "logical"
Numeric	12.3, 5, 999	v <- 23.5 print(class(v)) it produces the following result: [1] "numeric"
Integer	2L, 34L, 0L	v <- 2L print(class(v)) it produces the following result: [1] "integer"
Complex	3 + 2i	v <- 2+5i print(class(v)) it produces the following result: [1] "complex"
Character	'a' , '"good", "TRUE", '23.4'	v <- "TRUE" print(class(v)) it produces the following result: [1] "character"
Raw	"Hello" is stored as 48 65 6c 6c 6f	v <- charToRaw("Hello") print(class(v)) it produces the following result: [1] "raw"

Vectors

When you want 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))
```

When we execute the above code, it produces the following result:

```
[1] "red" "green" "yellow"

[1] "character"
```

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(list1)
```

When we execute the above code, it produces the following result:

```
[[1]]

[1] 2 5 3

[[2]]

[1] 21.3

[[3]]

function (x) .Primitive("sin")
```

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)
```

When we execute the above code, it produces the following result:

```
     [,1]  [,2]  [,3]
[1,] "a"   "a"   "b"
[2,] "c"   "b"   "a"
```

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. In the below example we create an array with two elements which are 3x3 matrices each.

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

When we execute the above code, it produces the following result:

```
, , 1

     [,1]     [,2]     [,3]
[1,] "green"  "yellow" "green"
[2,] "yellow" "green"  "yellow"
[3,] "green"  "yellow" "green"

, , 2

     [,1]     [,2]     [,3]
[1,] "yellow" "green"  "yellow"
[2,] "green"  "yellow" "green"
[3,] "yellow" "green"  "yellow"
```

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.The **nlevels** functions gives the count of levels.

```
# Create a vector

apple_colors <- c('green', 'green', 'yellow', 'red', 'red', 'red', 'green')

# Create a factor object

factor_apple <- factor(apple_colors)

print(factor_apple)

print(nlevels(factor_apple))
```

When we execute the above code, it produces the following result:

```
[1] green green yellow red red red yellow green

Levels: green red yellow
```

applying the nlevels function we can know the number of distinct values

```
[1] 3
```

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.

```
# 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)
```

When we execute the above code, it produces the following result:

	Gender	height	weight	Age
1	Male	152.0	81	42
2	Male	171.5	93	38
3	Female	165.0	78	26