

E-Content on Data Manipulation and Visualization through Statistical Software R (Open Source Software)

Dr Gurpreet Singh Tuteja and Dr Dhiraj Kumar Singh
Department of Mathematics
Zakir Husain Delhi College
(University of Delhi)
Jawaharlal Nehru Marg, Delhi - 110002

Day 1: July 27, 2020

1 About the College

Zakir Husain Delhi College (accredited A grade by NAAC) is one of the leading colleges of University of Delhi imparting higher education in Science, Commerce, Humanities and Social Sciences. The college holds the distinction of being in existence well before the establishment of the University of Delhi. Today, the college offers undergraduate courses in 21 disciplines and 17 post-graduate courses. The strong academic environment is complemented by various societies giving a unique platform for students to nurture their talents in different domains like dramatics, music, dance, art, fashion and photography. The college is to organize online workshop on the topic "Data manipulation and visualization through statistical software 'R'" from July 27-31, 2020 **under the aegis of DBT-STAR college scheme.**

Organizing Team

Prof. (Dr.) Masroor Ahmad Beg, Principal
Dr. P.K. Shishodia, Coordinator DBT-STAR College Scheme
Dr. Dhiraj Kumar Singh, Course Convenor
Dr. Sunil Kumar, Co-Convenor
Dr. Pankaj Sharma, Tutor
Dr. Pratima Negi, Tutor
Dr. Gurpreet Singh Tuteja, Resource Person

2 About this Course

Title: Data Manipulation and Visualization through Statistical Software R (Open Source Software)

What is R? Why R is becoming so important? What is the role of R software in statistics, mathematics and other basic sciences? The statistical software R has positioned itself as an important tool for statistical and visual analysis. So, whether you are a student of mathematics, statistics, engineering, physics, chemistry, pharmacy, fashion, medicine, social science or a researcher, definitely this workshop will be a starter for you. This workshop will cover the basics of R, right from the installation to data visualization and exploratory data analysis. The entire course will be delivered online. Classes shall be held every day for 2 hours including practical from July, 27-31, 2020 either on Google Meet or Zoom. Being practical based course, candidates must have their working reliable laptops in every online class.

Eligibility: Participant must have studied at least one paper of Mathematics at UG/Senior Secondary level.

Who can apply? Faculty, Research Scholars, UG and PG Students (including First Year) are eligible.

Contact: *dbtstarcollege@zh.du.ac.in*

3 Programme Planner

Introduction and Installation of R software, using R- studio effectively, introductory R language fundamentals and basic syntax, understanding critical programming language concepts for performing data analysis, import data into R from external files, create summary statistics, create new variables, informative data visualization and tables, exploratory data analysis. The working data sets include Covid live data.

Each session will be followed by a quiz, and a final project to be submitted. The attendance will be marked through the quiz only which will be conducted either before or after each session.


Duration: July, 27-31, 2020 (10 Hrs)

Award of Certificate: Certificate will be awarded only to those participants who will complete 80% of attendance subject to passing of a final test (overall 60%).

4 Session 1

Welcome to one Week workshop on Data Manipulation and Visualization using R. I am happy that you have joined this course and at the end of this course you will be ready to take a deep dive into R. Before we have our introductory session, I expect that you will install R in your laptop or desktop.

4.1 What is R?

 is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems like Linux, Windows and Mac.

4.2 Why it is called R ?

R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is developed by the RDevelopment Core Team (of which, as of August 2018, R is named partly after the first names of the first two R authors and partly as a play on the name of S.

S was created by John Chambers in 1976, while at Bell Labs

4.3 Statistical Features of R

R and its libraries implement a wide variety of statistical and graphical techniques, including linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, and others.

R is easily extensible through functions and extensions, and the R community is noted for its active contributions in terms of packages. Another strength of R is static graphics, which can produce publication-quality graphs, including mathematical symbols. Dynamic and interactive graphics are available through additional packages.

4.4 Programming Features of R

R is an interpreted language; users typically access it through a command-line interpreter. Like other similar languages such as APL and MATLAB, R supports matrix arithmetic. R's data structures include vectors, matrices, arrays, data frames (similar to tables in a relational database) and lists. Arrays are stored in column-major order. R supports procedural programming with functions and, for some functions, object-oriented programming with generic functions.

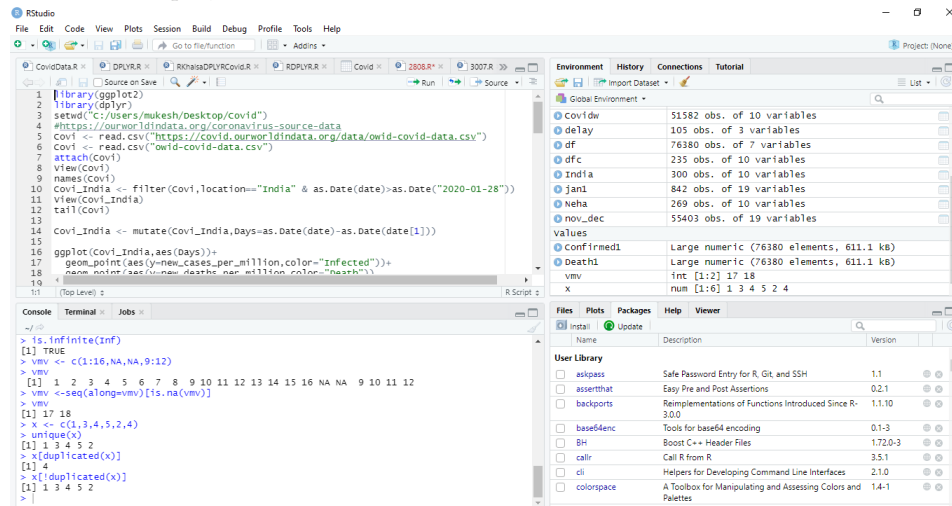
4.5 Packages

The capabilities of R are extended through user-created packages, which allow specialised statistical techniques, graphical devices, import/export capabilities, reporting tools (Rmarkdown, knitr, Sweave), etc.

A group of packages called the Tidyverse, which can be considered a "dialect of the R language", is increasingly popular in the R ecosystem. A group of packages strives to provide a cohesive collection of functions to deal with common data science tasks, including data import, cleaning, transformation and visualisation (notably with the ggplot2 package).

4.6 R Interface

The most specialized integrated development environment (IDE) for R is RStudio. A similar development interface is R Tools for Visual Studio. Some generic IDEs like Eclipse, also offer features to work with R.




RStudio is the premier integrated development environment for R. It is available in open source and commercial editions on the desktop (Windows, Mac, and Linux) It is available in two formats: RStudio Desktop is a regular desktop application while RStudio Server runs on a remote server and allows accessing RStudio using a web browser.

4.7 Comparison with R, SPSS and Stata

R is comparable to popular commercial statistical packages such as SAS, SPSS, and Stata, but R is available to users at no charge under a free software license. R is more procedural-code oriented than either SAS or SPSS, both of which make heavy use of pre-programmed procedures (called "procs") that are built-in to the language environment and customized by parameters of each call. R generally processes data in-memory, which limits its usefulness in processing extremely large files.

4.8 Installation

You will first need to download and install both R and RStudio (Desktop version) on your computer. It is important that you install R first and then RStudio. The step-wise procedure is as follows: You must do this first: Download and install R by visiting the URL: <https://cloud.r-project.org/>



CRAN
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

About R
[R Homepage](#)
[The R Journal](#)

Software
[R Binaries](#)
[Packages](#)
[Other](#)

Documentation
[Manuals](#)
[FAQs](#)
[Contributed](#)

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2020-10-10, Bunny-Wunnies Freak Out) [R-4.0.3.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#)

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

If you are a Windows user: Click on "Download R for Windows", then click on "base", then click on the Download link.

If you are Mac OS user: Click on "Download R for (Mac) OS X", then under

"Latest release:" click on R-X.X.X.pkg, where R-X.X.X is the version number.

If you are a Linux user: Click on "Download R for Linux" and choose your distribution for more information on installing R for your setup.

This will complete your setup of R. Now you need a GUI as a workspace. Download and install RStudio at:

https: //www.rstudio.com/products/rstudio/download/



RStudio DOWNLOAD SUPPORT DOCS COMMUNITY

Products Solutions Customers Resources About Pricing

Download the RStudio IDE

Choose Your Version

The RStudio IDE is a set of integrated tools designed to help you be more productive with R and Python. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace.

[LEARN MORE ABOUT RSTUDIO FEATURES](#)

RStudio Team

RStudio's recommended professional data science solution for every team. RStudio Team is a bundle of RStudio's popular professional software for data analysis, package management, and sharing data products.

[Learn more about RStudio Team](#)

RStudio Desktop	RStudio Desktop Pro	RStudio Server	RStudio Server Pro
Open Source License	Commercial License	Open Source License	Commercial License
Free	\$995	Free	\$4,975

Scroll down to "Installers for Supported Platforms" near the bottom of the page. Click on the download link corresponding to your computer's operating system. Once you finish the downloading, install R studio in your machine. After the successful completion of the installation, you are ready to explore the world of R.

4.9 Basics of R

4.9.1 Getting Help in R

The simplest way to get help in R is to click on the Help button on the toolbar of the RGUI window (this stands for R Graphical Interface. Alternatively, if you are connected to the internet, you can type CRAN into Google and search for the help you need at CRAN. However, if you know the name of the function you want help with, you just type a question mark ? at the command line prompt followed by the name of the function. So to get help on read.table, just type *?mean* or *help(mean)*

Sometimes you cannot remember the precise name of the function, but you know the subject on which you want help (e.g. data input in this case). Use the help.search function (without a question mark) with your query in double quotes like this: *help.search("median")*. The function help.start() presents documentation of R *help.start()*. The help for packages can be checked by *help(package="dplyr")*. Other useful functions are find and apropos. The find function tells you what package something is in: *find("cat") find("filter")*, while apropos returns a character vector giving the names of all objects in the search list that match your (potentially partial) enquiry: *apropos("lm")*. To see a worked example just type the function name (e.g. mean) *example(mean)*. Use args for a quick reminder of the function arguments: *args(matrix)*. Data sets in package 'datasets' can be accessed: *data()*. To quit R, type *q()*

4.9.2 Hello World!

Lets write the first R Program 'Hello World'

```
print("Hello World")
```

```
print("Dear", "Hello World !")
```

 This may not work!

For combining text or even individual string you can follow one of the following method:

```
print(paste("Dear", "Hello World"))
```

```
cat("Dear", "Hello World!")
```

4.10 Assignment Operator

At the R prompt we type expressions. The ← symbol is the assignment operator

```
x ← 5
```

```
x
```

```
print(x)
```

```

y = 7
y
4 → z
print(z)

```

4.10.1 Chaining Assignment

$$a \leftarrow b \leftarrow c \leftarrow 10$$

cat is useful for producing output in user-defined functions. It converts its arguments to character vectors, concatenates them to a single character vector, appends the given sep = string(s) to each element and then outputs them.

```
cat(a,b,c)
```

Note: The # character indicates a comment. Anything to the right of the # (including the # itself) is ignored. This is the only comment character in R. Unlike some other languages, R does not support multi-line comments or comment blocks.

```

x ← 7
print(x)
[1]7

```

shown in the output indicates that x is a vector and 7 is its first element. Typically with interactive work, we do not explicitly print objects with the print function; it is much easier to just auto-print them by typing the name of the object and hitting return/enter. However, when writing scripts, functions, or longer programs, there is sometimes a need to explicitly print objects because auto-printing does not work in those settings.

4.10.2 Variable Names and Assignment

There are three important things to remember when selecting names for your variables in R:

1. Variable names in R are case sensitive, so y is not the same as Y.
2. Variable names should not begin with numbers (e.g. 1x) or symbols (e.g. \$).
3. Variable names should not contain blank spaces (use back.pay not back pay).

4.10.3 Session Management

Find the current working directory (where inputs are found and outputs are sent).

```
getwd()
```

Setting up working directory

```
setwd("C : /Users/... /Desktop/R")
```

All variables created in R are stored in a common work space. You can access them by using ls()

```
ls()
```

In case you want to get rid of any variable use rm(remove)

```
rm(list of Objects)
```

```
rm(a)
```

```
rm(a,b)
```

You can wipe off or remove all objects or Clear Workspace

```
rm(list=ls())
```

To see what variables you have created in the current session, type:

```
objects()
```

To see which packages and dataframes are currently attached:

```
search()
```

4.10.4 R Objects

R has five basic or “atomic” classes of objects:

1. Character
2. Numeric (Real Numbers)
3. Integer
4. Complex Numbers
5. Logical (True/False)

The most basic type of R object is a vector. Empty vectors can be created with the vector() function. There is really only one rule about vectors in R, which is that A vector can only contain objects of the same class. A list is represented

as a vector but can contain objects of different classes, indeed, that's usually why we use them.

Numbers

Numbers in R are generally treated as numeric objects (i.e. double precision real numbers). This means that even if you see a number like "1" or "2" in R, which you might think of as integers, they are likely represented behind the scenes as numeric objects (so something like "1.00" or "2.00"). This isn't important most of the time, except when it is.

```
x ← 8L
```

```
class(x)
```

```
y = 9
```

```
class(y)
```

If you explicitly want an integer, you need to specify the L suffix. So entering 1 in R gives you a numeric object; entering 1L explicitly gives you an integer object. There is also a special number Inf which represents infinity. This allows us to represent entities like 1/0.

```
1/0
```

This way, Inf can be used in ordinary calculations; e.g. $1/\infty$ is 0. The value *NaN* represents an undefined value ("not a number"); e.g. 0/0; *NaN* can also be thought of as a missing value (more on that later).

```
1/∞
```

```
Find : 0/0, ∞ - ∞, ∞/∞
```

The number of decimal places is not the same as the number of significant digits. You can control the number of significant digits in a number using the function signif. Take a big number like 12345 678 (roughly 12.35 million). Here is what happens when we ask for 4, 5 or 6 significant digits:

```
signif(123456789,5)
```

The random number can be generated using runif(n,min=0,max=1)

```
runif(10,1,10)
```

```
floor(runif(10,1,10))
```

It can also be generated by using the sample command and rnorm:

```
sample(1:10)
```

```
rnorm(10,0,1)
```

Some other useful commands are :

Integer division

```
23%%5
```

Modulo division

```
23%%4
```

The output for the following will be either TRUE or FALSE :

```
23%%5 == 0
```

Complex Numbers

Following are few functions related to Complex number manipulations are ;

```
z ← 7 + 5i
```

Re(z) real part of a complex number z

Im(z) imaginary part of a complex number z

Arg(z) argument of a complex number z

Conj(z) conjugate of a complex number

One can find whether a number is complex or not using

`is.complex(z)`

The coercive conversion of a number into complex is done using :

`as.complex(3.5)`

4.10.5 Attributes

R objects can have attributes, which are like metadata for the object. These metadata can be very useful in that they help to describe the object. For example, column names on a data frame help to tell us what data are contained in each of the columns. Some examples of R object attributes are:

- names, dimnames
- dimensions (e.g. matrices, arrays)
- class (e.g. integer, numeric)
- length
- other user-defined attributes/metadata

Attributes of an object (if any) can be accessed using the `attributes()` function. Not all R objects contain attributes, in which case the `attributes()` function returns NULL.

```
x ← c(1, 2, 3, 4, 5, 6), 2, 3)
```

```
x
```

```
attributes(x)
```

```
dim(x)
```

4.10.6 Creating Vectors

The `c()` function can be used to create vectors of objects by concatenating things together.

```
x ← c(0.5, 0.6) #numeric
```

```
x ← c(TRUE, FALSE) # logical
```

```
x ← c(T, F) # logical(lazy)
```

```
x ← c("a", "b", "c") # character
x ← 9 : 29 # integer
x ← c(1 + 0i, 2 + 4i) # complex
```

You can also use the `vector()` function to initialize vectors:

```
vector("numeric", length = 5)
vector("logical", length=5)
?vector
```

4.10.7 Mixing Objects

There are occasions when different classes of R objects get mixed together. Sometimes this happens by accident but it can also happen on purpose. So what happens with the following code? `x ← c(1.7, "a")` # character `y ← c(TRUE, 2)` # numeric `z ← c("a", TRUE)` # character `class(x)`

In each case above, we are mixing objects of two different classes in a vector. But remember that the only rule about vectors says this is not allowed. When different objects are mixed in a vector, coercion occurs so that every element in the vector is of the same class.

For example, combining a numeric object with a character object will create a character vector, because numbers can usually be easily represented as strings.

4.10.8 Explicit Coercion

Objects can be explicitly coerced from one class to another using the `as.*` functions, if available

```
x ← 0L : 6L
x
class(x)
x ← 0 : 10
as.numeric(x)
as.character(x)
as.logical(x)
```

Sometimes, R can't figure out how to coerce an object and this can result in NAs being produced

```
u ← c("a", "b", "c")
as.integer(u)
```

4.10.9 Equality of floating point numbers using `all.equal`

The nature of floating point numbers used in computing is the cause of some initially perplexing features. You would imagine that since 0.3 minus 0.2 is 0.1, and the logic presented below would evaluate to TRUE. Not so:

```
x ← 0.3 - 0.2
```

```
y ← 0.1
```

```
x == y
```

 The function called `identical` gives the same result:

```
identical(x,y)
```

The solution is to use the function called `all.equal` which allows for insignificant differences:

```
all.equal(x,y)
```

Do not use `all.equal` directly in if expressions. Either use `isTRUE(all.equal(...))` or `identical` as appropriate.

5 Reading and Writing Data

In this section we will learn how to read data from a local file or a file from online resource. For small to moderately sized data-sets, usually call `read.table` without specifying any other arguments. This will read the table from the root directory.

```
data ← read.table("foo.txt")
```

You can also assign a path to the file as given in the following example:

```
data ← read.table("E:/Desktop/R/foo.txt")
```

Similarly, we can read any `.csv` file stored in the root directory or elsewhere.

```
Covid ← read.csv("https://covid.ourworldindata.org/data/owid-covid-data.csv")
```

This will fetch the updated file from the website `ourworldindata.org` on the fly.

For, writing the contents of a dataframe or table use:

```
write.csv("name of the dataframe")
```

This ends the first session. You are requested to appear in the quiz. Time allotted for the quiz is 15 mins.