

Introduction to R and RStudio

Getting Started with R

Henry Athiany

2022-04-12

Gentle Introduction

- ▶ The materials presented in this training is a result of work done by several researchers, and colleagues who have either presented or trained in similar workshops.
- ▶ We do acknowledge the various sources at the end of the training materials
- ▶ However, these slides, do not provide every detail about R programming, but serves as a good start to learning the software and its applications.
- ▶ We therefore recommend that you take time to practice using various resources provided at the end of each session.

Basic concepts of statistics

This training assumes that participants have some background of the basic concepts of statistics. Specifically, we assume that you are familiar with the following concepts:

- ▶ Descriptive statistics
 - ▶ Central Tendency and Dispersion
 - ▶ The shape of distributions
- ▶ Basic inferential Statistics

We also assume that you have a background knowledge on the Types of Variables & Measurement Scales. However, we shall try to give details on the different concepts as we use them for illustrations.

For instance, there are four measurement scales (or types of data): **nominal**, **ordinal**, **interval** and **ratio**. These are simply ways to categorize different types of variables.

Let's Talk R!

- ▶ An open source programming language and software environment for statistical computing and graphics. Hence, serves as a data analysis and storage facility.
- ▶ Created by **R**oss Ihaka and **R**obert **G**entleman in 1992 (University of Auckland, New Zealand)
- ▶ Yes, **R** stems from the first letter of their first names, inspired by another programming language called **S**
- ▶ Allows for rapid development of new tools according to user demand

Let's Talk R!

- ▶ These tools are distributed as packages, which any user can download to customize the R environment.
- ▶ Currently supported by the R Foundation for Statistical Computing (Vienna, Austria)
- ▶ Has an active Core Team, very large and active community of R- Users

Benefits of R Programming

- ▶ **Open Source:** anyone can work with R without any need for a license or a fee and can customize its packages, develop new ones and resolve issues
- ▶ **Exemplary Support for Data Wrangling:** good support for data wrangling. Packages like dplyr, readr are capable of transforming messy data into a structured form.
- ▶ **The Array of Packages:** Over 18,000 packages in the **CRAN** repository, the number is constantly growing
- ▶ **Quality Plotting and Graphing:** popular libraries like ggplot2 and plotly advocate for aesthetic and visually appealing graphs

Benefits of R Programming

- ▶ **Highly Compatible:** can be paired with languages like C, C++, Java, and Python
- ▶ **Platform Independent:** run quite easily on Windows, Linux, and Mac
- ▶ **Eye-Catching Reports:** Shiny and Markdown, reporting the results is easy with R
- ▶ **Machine Learning Operations:** machine learning operations like classification, regression
- ▶ **Statistics:** for developing statistical tools

R drawbacks:

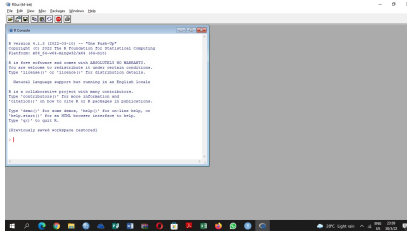
- ▶ **Weak Origin:** shares origin with old languages like **S**. Base package does not have support for dynamic or 3D graphics
- ▶ **Data Handling:** physical memory stores the objects, hence utilising more memory
- ▶ **Complicated Language:** R is not an easy language to learn. It has a steep learning curve
- ▶ **Lesser Speed:** R packages and the R programming language is much slower than other languages like MATLAB and Python
- ▶ **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

Base R

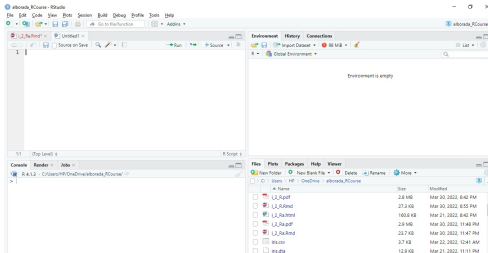
Ways to use R

There are two main ways to use R;

► Base R



► RStudio



Installing R

- ▶ To install R, first you need to download the executable file from the repository
- ▶ R **repositories** contain package tar files and are the primary vehicle for organizing and distributing R packages
- ▶ To do this, go to <https://cloud.r-project.org/>.
- ▶ This is the cloud mirror for the Comprehensive R Archive Network (CRAN), which is the online home for R and its packages.
- ▶ At the top of the page, there are three links for downloading R.

Installing R

- Choose the link that describes your operating system; e.g Linux, Mac or Windows as shown below



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

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



The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** these versions of R:

- [Download R for Linux](#) ([Debian](#), [Fedora/Redhat](#), [Ubuntu](#))
- [Download R for macOS](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper


- I am a Windows user, so the examples given are for Windows Operating System

Installing R

- ▶ Once you have clicked on *Download for Windows*, the following screen appears.
- ▶ It indicates the most current version of R. New versions of R are produced approximately once a year. There are also smaller updates in between the year.

R FOR WINDOWS


Subdirectories:

	base	Binaries for base distribution. This is what you want to install R for the first time .
	contrib	Binaries of contributed CRAN packages (for R >= 3.4.x).
	old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 3.4.x).
	Rtools	Tools to build R and R packages. This is what you want to build your own packages on W

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of ques

You may also want to read the [R FAQ](#) and [R for Windows FAQ](#).

R-4.1.3 for Windows (32/64 bit)

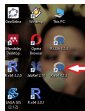
	Download R 4.1.3 for Windows (87 megabytes, 32/64 bit)
	Installation and other instructions
	New features in this version

Installing R

- ▶ Once downloaded, there are instructions on how to install R in your computer.
- ▶ Follow these instructions to install R.
- ▶ In addition, there are new features of R in the current version.
- ▶ There are two possible flavours of R: 32 - bit and 64 - bit R. Simply select both when using the installer

Launching R

- ▶ Once you have installed R, double click on the icon shown with the arrow to launch R



- ▶ Once launched, the following screen appears indicating R version, nickname and launching date e.g “One Push-Up” for R version 4.1.3 (2022-03-10)

Launching R

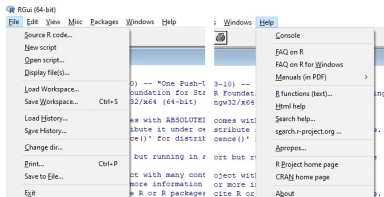
- ▶ Now that we have launched R, we can then use R to perform various operations.
- ▶ To start using R, we can just type at the console as shown with the arrow. Typing `version` and pressing enter, gives details of the current installed version of R (see below).

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> version

```
platform      x86_64-w64-mingw32
arch           x86_64
os             mingw32
system        x86_64, mingw32
status
major          4
minor          1.3
year           2022
month          03
day            10
svn rev        81860
language       R
version.string R version 4.1.3 (2022-03-10)
nickname       One Push-Up
```



- ▶ We also see that we can use the file menu to launch a new script file (commonly used in R) new Workspace and save files. Similarly, the help menu has several options.

Datasets in R

- ▶ The base R software installed comes with several datasets. Typing `data()` at the console and then pressing enter, gives a list of all datasets installed with base R.



```
R R datasets
Data sets in package 'datasets':

AirPassengers      Monthly Airline Passenger Numbers 1949-1960
BJsales            Sales Data with Leading Indicator
BJsales.lead (BJsales) Sales Data with Leading Indicator
BOD                Biochemical Oxygen Demand
CO2                Carbon Dioxide Uptake in Grass Plants
ChickWeight        Weight versus age of chicks on different diets
DNase              Elisa assay of DNase
EuStockMarkets     Daily Closing Prices of Major European Stock
                  Indices, 1991-1998
Formaldehyde       Determination of Formaldehyde
HairEyeColor       Hair and Eye Color of Statistics Students
Herman23.cox       Harman Example 2.3
Herman74.cox       Harman Example 7.4
Indometh            Pharmacokinetics of Indomethacin
InsectSprays       Effectiveness of Insect Sprays
```

- ▶ Additional datasets are added as new packages are installed. Actually, these datasets are contained in a package called `dataset`. More details on packages at a later stage
- ▶ To get more information about each of the datasets, type `?dataset name`. For instance, if a dataset is called `trees`, then type `?trees` for more information about the dataset.
- ▶ We shall use a number of these datasets in our illustrations

The R command line and script editor

- ▶ In R, the command line comes at the bottom of the R console with the prompt `>`. Typing a command such as `3*4` will return a solution as shown below.

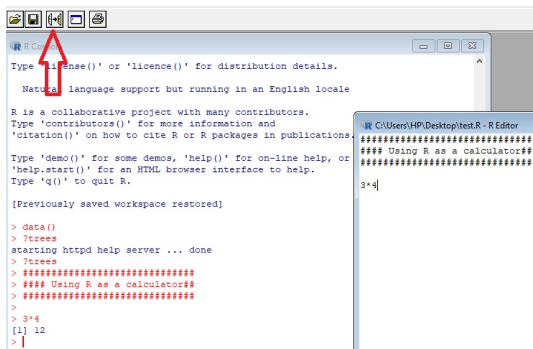
```
# Multiplying 3 by 4  
3 * 4
```

```
## [1] 12
```

- ▶ However, since this will disappear after quitting R, it is always good to have a better way of running the commands.
- ▶ We use the R scripts to save the commands.
- ▶ An R script is just a text file that contains R code and comments. When saved, the file extension is `.R`, e.g. `r_mod1.R`.
- ▶ To quit R, type `q()` at the command prompt. A message will appear asking if you would like to save the workspace image. Choose one applicable option!

Runing R Script

- ▶ The following figure shows how an R script can be used to obtain the same solution arrived at from the command line
- ▶ Click on the icon shown by the arrow to run the script.



- ▶ We shall learn more about the command line (console) in details once we have installed RStudio and ready for use in our computers.

Other script editors

- ▶ If you have worked with any other statistical program before, then you may be familiar with the idea of writing, modifying, saving and sharing code scripts. SAS calls these code scripts SAS programs, Stata calls them DO files, SPSS calls them SPSS syntax files.
- ▶ You may have noticed that this installation of base R comes with a simple Graphical User Interface (GUI). However, we will not be using this interface anymore during this training.
- ▶ The remaining sections of this training will now be using R via RStudio platform/environment.

R Workspace

- ▶ It is good practice to always start by setting your working directory
- ▶ To do this, go to the menu and click Session -> change dir (Set Working Directory) -> Choose directory.
- ▶ This will allow you to choose the folder you prefer to work from.
- ▶ You can then confirm the current working directory by typing `getwd()` at the command line/console

R Workspace

- ▶ Alternatively, you can set your working directory as follows:

```
# Get the path of your current working directory  
getwd()
```

```
## [1] "C:/Users/HP/OneDrive/alborada_RCourse"
```

```
# set working directory  
setwd("C:/Users/HP/OneDrive/alborada_RCourse")  
# similar to above but see the backslash  
setwd("C:\\Users\\HP\\OneDrive\\alborada_RCourse")  
getwd()
```

```
## [1] "C:/Users/HP/OneDrive/alborada_RCourse"
```

R Workspace

- ▶ R stores both data and output from data analysis in objects.
- ▶ Data are assigned to and stored in objects using the two assignment operators `<-` or `=`
- ▶ Using the script editor, let's start by creating some few objects in R. Print the contents of an object to the console, by specifying the object's name

```
# creating our first few objects  
x <- 10 # Assigning the value 10 to the variable x  
x # Calling the value assigned to x
```

```
## [1] 10
```

```
y = 2 # Assigning the value 2 to the variable x  
z <- x/y # assigning the value z to x/y  
z # calling z
```

```
## [1] 5
```

R Workspace

- ▶ So far, we have created few objects in R that are contained within your current working environment R workspace.
- ▶ This is internal to R and not a file stored on your computer.
- ▶ To see a list of the objects within your R workspace, or remove objects from your working space, we use the following codes

```
# lets assign an object first
```

```
tmp <- 1:4
```

```
ls() #list objects in R
```

```
## [1] "tmp" "x"   "y"   "z"
```

```
rm(x) # Remove a specific object :rm(object)
```

```
# You will get no warning, so don't do this unless you are
```

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


Coding in R

- ▶ Before we get into using RStudio, we would like to learn something about coding in R.
- ▶ Previously, we assigned data to objects using the operators `<-` or `=`. This is applicable for numerical data
- ▶ For character data (strings), we use the double quotes `" "` for assignment. e.g `b="James"`
- ▶ New R commands are usually placed in a new line or separated by the operator `;` (semicolon)
- ▶ Commands can extend beyond one line of text. Put operators like `+` at the end of lines for multi-line commands.
- ▶ The character (`#`) used at the beginning or side of lines means a comment. This is not executed by the program
- ▶ We have arithmetic and logical operators in R. Both are described in the next slide
- ▶ We note that the assignment operator `"="` is not the same as `"=="`.

Arithmetic and logical Operators in R

Operator	Description
+	Addition
-	Subtraction
/	Division
*	Multiplication
^ or **	Exponentiation
<	Less Than
>	Greater Than
<=	Less Than or Equal To
>=	Greater Than or Equal To
==	Exactly Equal To
!=	Not Equal To
!=	Not Equal To
!a	Not a
a&b	a AND b

Logic in R: Example

We can then see an example of this as follows:

```
a <- c(1:12)  
# a>9 OR a<4 Gives us 1 2 3 10 11 12 Having R do this  
a
```

```
## [1] 1 2 3 4 5 6 7 8 9 10 11 12
```

```
a > 9
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FAI
```

```
a < 4
```

```
## [1] TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FAI
```

```
a > 9 | a < 4
```

```
## [1] TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FAI
```

```
a[a > 9 | a < 4]
```

```
## [1] 1 2 3 10 11 12
```

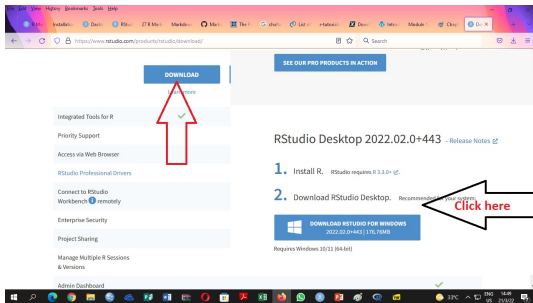
RStudio

Installing RStudio

- ▶ As previously mentioned, you can always work directly in R, but most users prefer a graphical interface.
- ▶ For this, we highly recommend using RStudio, an Integrated Development Environment (IDE) that features:
 - ▶ a console
 - ▶ a powerful code/script editor featuring
 - ▶ special tools for plotting, viewing R objects and code history
 - ▶ cheatsheets for R programming
 - ▶ tab-completion for object names and function arguments

Installing RStudio

- ▶ To install RStudio, go to <https://www.rstudio.com/products/rstudio/download/>
- ▶ This leads to the current download page for RStudio IDE.
- ▶ On the page, find the download button for the RStudio Desktop (Open Source Licence)
- ▶ Click the button to download and get the version of RStudio that you would like to use, preferably the latest.



Using R via RStudio

- ▶ At its simplest, **R** is like a car's engine while **RStudio** is like a car's dashboard as illustrated in Figure

R: Engine



RStudio: Dashboard



- ▶ Whereas runs computations, RStudio is an IDE that provides an interface by adding many convenient features and tools
- ▶ Having installed R and RStudio on your computer, you have two new programs (also called *applications*) you can open

R: Do not open this

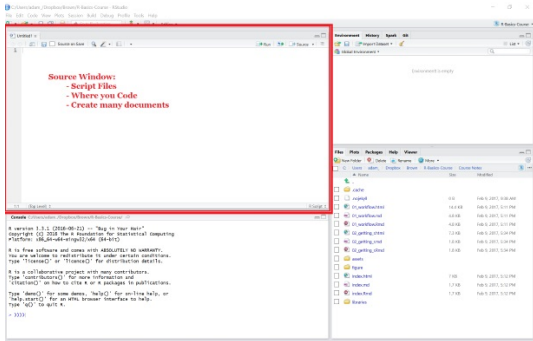


RStudio: Open this

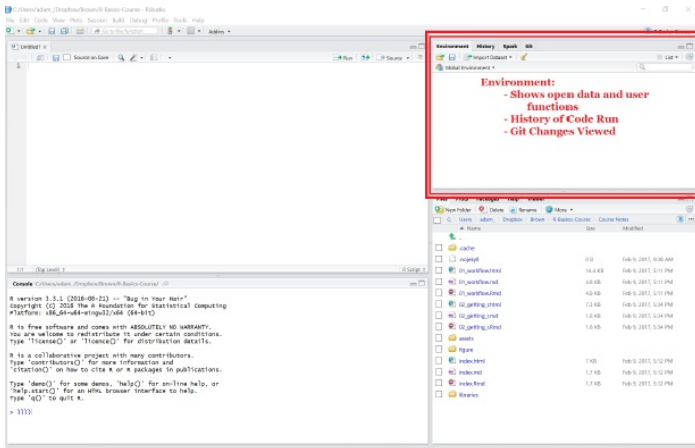


Using RStudio

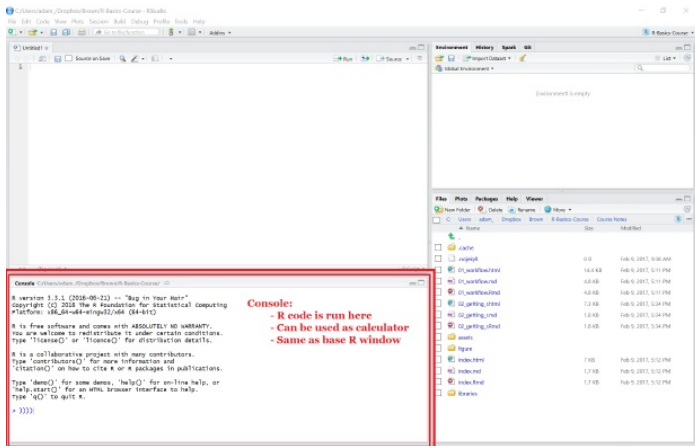
- ▶ To use RStudio, open it as you would open any program in your computer. You can even begin to use the console.
- ▶ If you are using it for the first time, by default you will have three panes (console pane, the files pane, and the environment pane), otherwise, it opens with the four panes as shown below



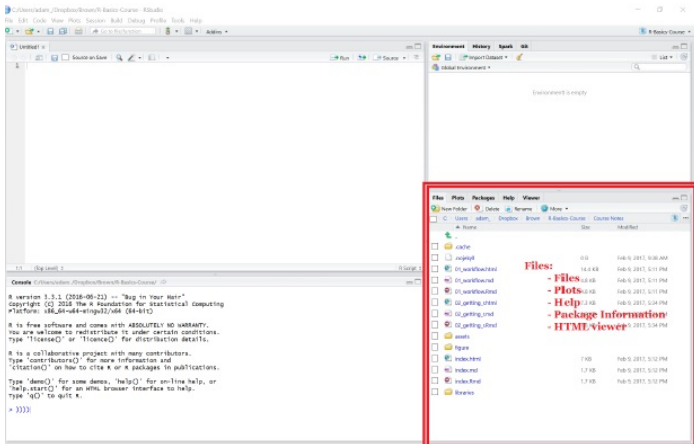
RStudio panes



RStudio panes



RStudio panes



RStudio Environment

- ▶ Now that we have installed RStudio, lets take a couple minutes to familiarize ourselves with the RStudio environment

Take note!

- ▶ R is case-sensitive: That is, the name **AMR** in R will not be the same as **amr** nor **Amr** nor **aMr** nor **amR**.

Explore the R console

- ▶ Generally, we don't work in the console, at least for writing important codes. We use R script instead
- ▶ However, let's take a few minutes to work with the console:
 - ▶ Click on the console, just to the right of one of the ">" prompts.
 - ▶ We can use the console like a calculator. For example, type `13+45` at the prompt and hit `enter`.
 - ▶ Now hit the "up" arrow. The R console remembers all previous commands that it has executed in this session, which allows you to re-run commands relatively easily.
 - ▶ Any command that is preceded by a pound sign (`#`) is ignored by the R console. Try typing `#2+2` and hitting "Enter". Nothing happens, right?

R as a Calculator

- ▶ A basic but useful purpose of R is to perform simple calculations

```
# Addition and Subtraction
```

```
5 + 4
```

```
## [1] 9
```

```
124 - 26.82
```

```
## [1] 97.18
```

```
# Multiplication and Division
```

```
5 * 4
```

```
## [1] 20
```

```
35/8
```

```
## [1] 4.375
```

```
# Exponentials and Exponential Function: 3^(1/2) & exp(1.5)
```

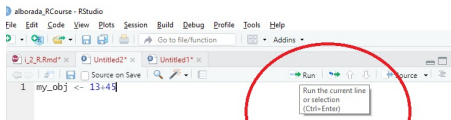
New script editor

- ▶ To create a new R script in RStudio, click on the “blank paper” icon at the far left of the RStudio toolbar (or File»New File»R script).
- ▶ The top left quadrant of the RStudio interface (at least by default) should now contain a blank R script.
- ▶ In your blank script, let's define a new object. For example:

```
my_obj <- 13 + 45  
my_obj
```

```
## [1] 58
```

- ▶ Place the cursor somewhere within the line of text, then hit ‘Ctrl+Enter’ (or ‘Command+Enter’) to run that line of code
Or Click on Run as shown here



Saving and Loading Script

- ▶ Take a few minutes to get comfortable running commands and defining new objects using your new R script
- ▶ Now that you have been able to write your first script in R, let us save it and then quite RStudio. Ensure you remember where you have saved the file.
- ▶ Open RStudio once more and try to locate the script file and then open it once more for use.

Comments in R

- ▶ Type the following line and run the code `#This is my first comment`
- ▶ What happens when you run the code? Any output?

```
my_obj <- 13 + 45  
my_obj
```

```
## [1] 58
```

```
# This is my first comment
```

- ▶ R ignores these lines (anything preceded by a hash (`#`) sign is ignored)- they are called comments, and they help to keep code organized and understandable
- ▶ use comments early and often- they are tremendously valuable

RStudio Cloud

- ▶ To prepare for the other sessions of this training, please:
- ▶ Sign up for an RStudio account at <https://rstudio.cloud/> (if you do not already have RStudio) before next class
- ▶ This will give you access to RStudio free of cost
- ▶ Provides 15 hours per month to use RStudio free of cost
- ▶ You will then be able to Watch the recordings of any sessions you missed during the training sessions.

R Packages: `install.packages()`

Working with R packages

- ▶ Base R and most R packages are available for download from the CRAN
 - ▶ base R comes with a number of basic data management, analysis, and graphical tools
 - ▶ However, R's power and flexibility lie in its array of packages
- ▶ To know which packages have been loaded in the R environment use the command

```
search()
```

```
## [1] ".GlobalEnv"          "package:knitr"        "package:st
## [4] "package:graphics"    "package:grDevices"    "package:ut
## [7] "package:datasets"    "package:methods"      "Autoloads"
## [10] "package:base"
```

- ▶ To learn more about a package eg. dplyr, one can check for the reference manual using Rseek or use the code;
help(package = dplyr)

Installing R packages

- ▶ To use packages in R, we must first install them using the *install.packages()* function, which typically downloads the package from CRAN and installs it for use.
- ▶ We may use the argument *dependencies = TRUE* to load all other packages required by the targeted package
- ▶ Example package installation is as follows;
 - ▶ `install.packages("pubh")`
 - ▶ `install.packages("pubh", quiet=TRUE)`
 - ▶ `install.packages("dplyr", dependencies=TRUE)`

```
# knowing the version of the package installed
packageVersion("pubh")
# packages installed on your computer/laptop
nrow(installed.packages())
```

- ▶ Ex: Try installing the package **gdata** using RStudio environment

Loading R Packages

- ▶ Once the packages have been installed, we can load them in the R environment using the functions *library()* or *require()*. The two functions do more or less the same task
- ▶ Functions and data structures within the package will then be available for use.
- ▶ The following is an example of how we can load a package for use in R

```
library(MASS)
```

```
library(dplyr, quiet = TRUE)
```

```
require(desc, quiet = TRUE)
```

- ▶ If a package has not been installed or not installed properly, then there will be a warning once its called
- ▶ Similarly, a warning may also be invoked if the package is not compatible with the version of R being used

Vignettes

- ▶ Many packages include vignettes – longer, tutorial style guides for a package.
- ▶ To see a list of available vignettes for the packages that are loaded, use `vignette()` with no arguments.

```
# list all available vignettes  
vignette()
```

- ▶ Then to view a vignette, place its name inside `vignette()`:
- ▶ For example, to view the “Introduction to pubh” vignette by issuing the command `vignette(“pubh”)`

Common R Packages

- ▶ In this section, we list and give details on some important packages in R.
- ▶ Generally, these are some of the most popular and commonly used packages.
- ▶ These packages do not have rankings in any order but are provided here due to their functionalities and diverse operations.

tidyr

As the name suggests, we use `tidyr` to make the data 'tidy'. It works well with `dplyr`. This is basically an evolution of the `reshape2` package which is described later.

ggplot2

With `ggplot2`, you can create graphics declaratively. `ggplot2` is famous for its elegant and quality graphs that sets it apart from other visualization packages.

Common R Packages

ggraph

`ggraph` is an extension of *ggplot2*. It takes away the limitation of *ggplot2*, that is, its dependency on tabular data.

dplyr

We use this library for performing data wrangling and data analysis. The `dplyr` library facilitates several functions for the data frames in R.

tidyquant

`tidyquant` is a financial package that is used for carrying out quantitative financial analysis. It adds to the *tidyverse* universe as a financial package. We can use it for importing, analyzing and visualizing the data.

dygraphs

The `dygraphs` package in R provides an interface to the main JavaScript library that we can use for charting. It is especially used for plotting time-series data in R.

Common R Packages

leaflet

The leaflet is an open-source JavaScript library for creating interactive visualizations. Popular websites like New York Times, Flickr, Github, etc use leaflet. The R package of leaflet makes it easy to interact with it.

ggmap

This is a mapping package that is used for delineating spatial visualizations. It also consists of various tools for geolocating and routing.

glue

The developers made this package for performing the operation of data wrangling. We use this package for evaluating R expressions that are present within the string.

Common R Packages

shiny

With the help of shiny, you can develop interactive and aesthetically pleasing web apps using R. It also provides various extensions with Cascading Style Sheets (CSS), HTML widgets and JavaScript.

plotly

The R package 'plotly' provides online interactive and quality graphs. It extends upon the JavaScript library *–plotly.js*.

tidytext

This package provides various functions of text mining for word processing and carrying out sentiment analysis through 'dplyr', 'ggplot' and other miscellaneous tools.

MASS

MASS provides a large number of statistical functions. It provides datasets that are in conjunction with the book "Modern Applied Statistics with S".

Common R Packages

Tidyverse

The core tidyverse includes the packages that you're likely to use in **everyday data analyses**. As of tidyverse 1.3.0, the following packages are included in the core tidyverse: `ggplot2`, `dplyr`, `tidyr`, `readr`, `purrr`, `tibble`, `stringr` and `forcats`.

Ex.

-Give a short description of the packages *pubh* and *epicalc*.

Getting Help in R

Getting Help in R

The help() Function

- ▶ To get online help within an R session we use the help() function.

```
# help() function with seq as argument  
help(seq)  
# Shortcut for help() is ?  
`?`(seq)
```

The example() Function

- ▶ Many times we just need to see some examples rather than read the entire documentation of a function or command. Then use,

```
example(seq)
```

```
##  
## seq> seq(0, 1, length.out = 11)  
## [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0  
##  
## seq> seq(stats::rnorm(20)) # effectively 'along'
```

R Resources

Useful R Resources

The following are some of the online resources that can help you learn R. There are also several youtube videos available that tackles specific concepts in R, for instance use of particular packages.

- ▶ **[CEMA]**(https://cema-uonbi.github.io/cema_courses.github.io/index.html#what-can-r-do) Quick R materials for Epidemiological Modeling and analysis
- ▶ **[UCLA R resources]**(<https://stats.oarc.ucla.edu/r/>) provides links to various R resources
- ▶ **[Quick-R]**(<https://www.statmethods.net/>) website containing many short topics and codes to get you started on data import, manipulation and statistics as implemented in .
- ▶ **Cheat Sheets** makes it easy to use some of the packages:
`help <- Cheat Sheets`
- ▶ **[Ken Mwai's Github]**(<https://github.com/Keniajin/I-StaR>) : very rich in materials that one can use to learn R

Let's take 5 minutes break!