Unit - 1

Overview in R - R Studio: R command Prompt, R script file, comments –Handling Packages in R: Installing a R Package, Few commands to get started: installedpackages(), packageDescription(), help(), find.package(), library() - Input and Output, Special Values functions- NA, Inf and –inf.

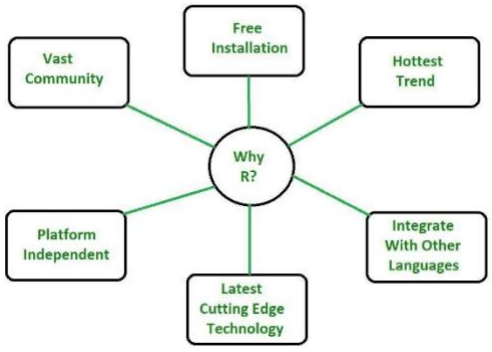
Why R?

R is not a programming language like C or Java. It was not created by software engineers for software development. Instead, it was developed by statisticians as an interactive environment for data analysis. You can read the full history in the paper A Brief History of S1. The interactivity is an indispensable feature in data science because, as you will soon learn, the ability to quickly explore data is a necessity for success in this field. However, like in other programming languages, you can save your work as scripts that can be easily executed at any moment. These scripts serve as a record of the analysis you performed, a key feature that facilitates reproducible work. If you are an expert programmer, you should not expect R to follow the conventions you are used to since you will be disappointed. If you are patient, you will come to appreciate the unequal power of R when it comes to data analysis and, specifically, data visualization.

R is an open-source programming language that is widely used as a statistical software and data analysis tool. R generally comes with the Command-line interface. R is available across widely used platforms like Windows, Linux, and macOS. Also, the R programming language is the latest cutting-edge tool.

It was designed by **Ross Ihaka and Robert Gentleman** at the University of Auckland, New Zealand, and is currently being developed by the R Development Core Team.

R programming language is an implementation of the S programming language. It also combines with lexical scoping semantics inspired by Scheme. Moreover, the project was conceived in 1992, with an initial version released in 1995 and a stable beta version in 2000.



R Variable Names (Identifiers)

Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for R variables are:

- A variable name must start with a letter and can be a combination of letters, digits, period(.)
- and underscore(_). If it starts with period(.), it cannot be followed by a digit.
- A variable name cannot start with a number or underscore (_)
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- Reserved words cannot be used as variables (TRUE, FALSE, NULL, if...)

Other attractive features of R are:

R is free and open source2.

It runs on all major platforms: Windows, Mac Os, UNIX/Linux.

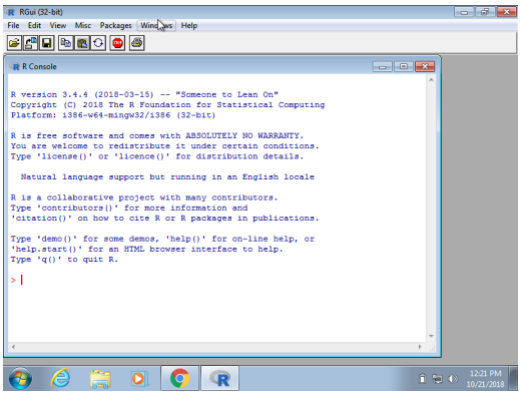
Scripts and data objects can be shared seamlessly across platforms.

There is a large, growing, and active community of R users and, as a result, there are numerous resources for learning and asking questions3 4 5.

It is easy for others to contribute add-ons which enables developers to share software implementations of new data science methodologies. This gives R users early access to the latest methods and to tools which are developed for a wide variety of disciplines, including ecology, molecular biology, social sciences, and geography, just to name a few examples.

The R console

Interactive data analysis usually occurs on the R console that executes commands as you type them. There are several ways to gain access to an R console. One way is to simply start R on your computer. The console looks something like this:

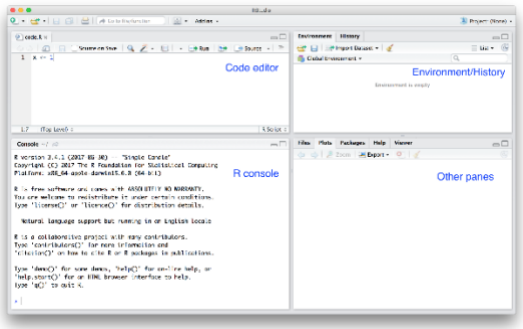


0.15 * 19.71

#> [1] 2.96

Scripts

One of the great advantages of R over point-and-click analysis software is that you can save your work as scripts. You can edit and save these scripts using a text editor. The material in this book was developed using the interactive integrated development environment (IDE) RStudio6. RStudio includes an editor with many R specific features, a console to execute your code, and other useful panes, including one to show figures.



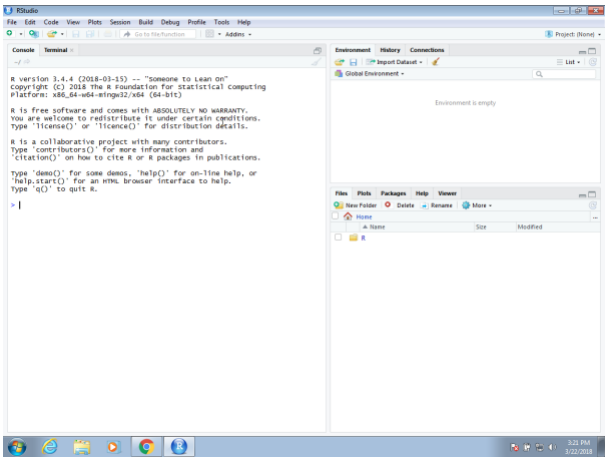
Most web-based R consoles also provide a pane to edit scripts, but not all permit you to save the scripts for later use.

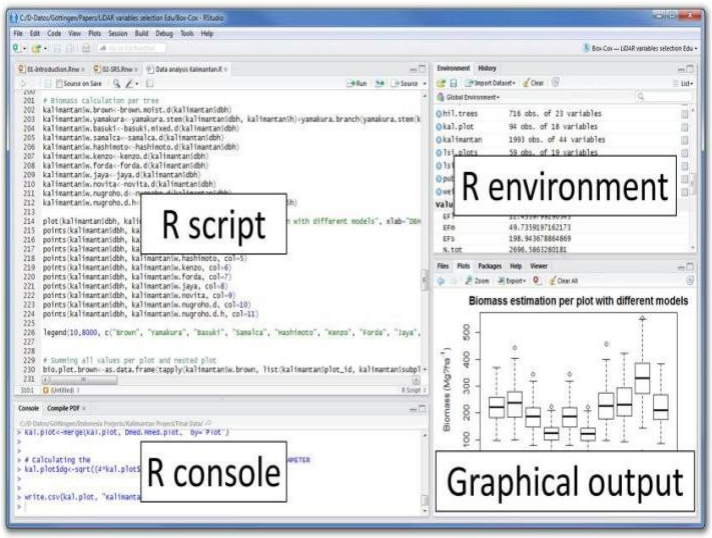
RStudio

RStudio will be our launching pad for data science projects. It not only provides an editor for us to create and edit our scripts but also provides many other useful tools. In this section, we go over some of the basics.

The panes

When you start RStudio for the first time, you will see three panes. The left pane shows the R console. On the right, the top pane includes tabs such as Environment and History, while the bottom pane shows five tabs: File, Plots, Packages, Help, and Viewer (these tabs may change in new versions). You can click on each tab to move across the different features.

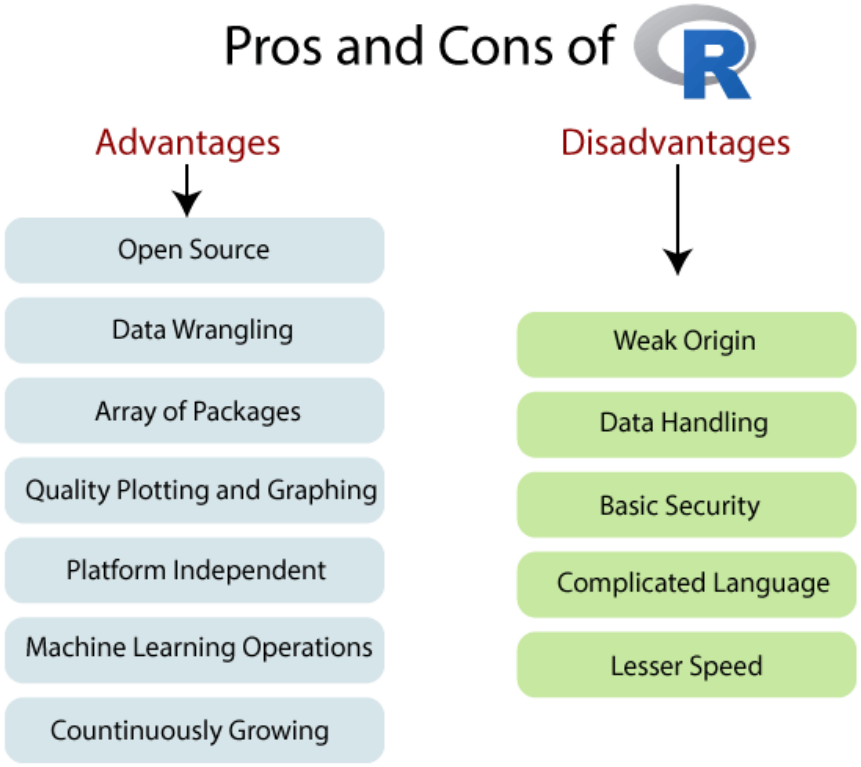




R Advantages and Disadvantages

R is the most popular programming language for statistical modeling and analysis. Like other programming languages, R also has some advantages and disadvantages. It is a continuously evolving language which means that many cons will slowly fade away with future updates to R.

There are the following pros and cons of R



Pros

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 Operations

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.

Cons

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.

In R, algorithms are spread across different packages. The programmers who have no prior knowledge of packages may find it difficult to implement algorithms.

What are the basic operations of R?

Arithmetic operations in R simulate various math operations, like addition, subtraction, multiplication, division, and modulo using the specified operator between operands, which may be either scalar values, complex numbers, or vectors.

Installing R packages

The functionality provided by a fresh install of R is only a small fraction of what is possible. In fact, we refer to what you get after your first install as base R. The extra functionality comes from add-ons available from developers. There are currently hundreds of these available from CRAN and many others shared via other repositories such as GitHub. However, because not everybody needs all available functionality, R instead makes different components available via packages. R makes it very easy to install packages from within R. For example, to install the dslabs package, which we use to share datasets and code related to this book, you would type:

```
install.packages("dslabs")
```

In RStudio, you can navigate to the Tools tab and select install packages. We can then load the package into our R sessions using the library function:

```
library(dslabs)
```

As you go through this book, you will see that we load packages without installing them. This is because once you install a package, it remains installed and only needs to be loaded with library. The package remains loaded until we quit the R session. If you try to load a package and get an error, it probably means you need to install it first.

We can install more than one package at once by feeding a character vector to this function:

```
install.packages(c("tidyverse", "dslabs"))
```

Note that installing tidyverse actually installs several packages. This commonly occurs when a package has dependencies, or uses functions from other packages. When you load a package using library, you also load its dependencies.

Once packages are installed, you can load them into R and you do not need to install them again, unless you install a fresh version of R. Remember packages are installed in R not RStudio.

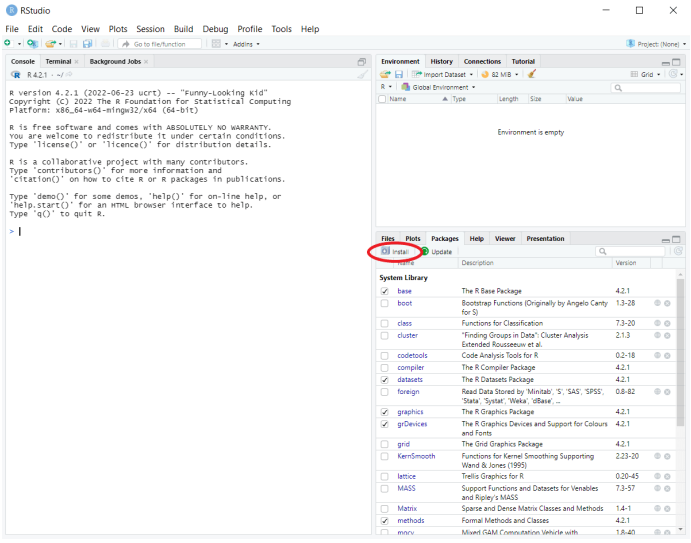
It is helpful to keep a list of all the packages you need for your work in a script because if you need to perform a fresh install of R, you can re-install all your packages by simply running a script.

You can see all the packages you have installed using the following function:

```
installed.packages()
```

RStudio:

- Remember to **install all packages in the console** rather than in a script file since they have to be installed on a computer's hard disk only once.
- You can install packages directly from the RStudio interface: open the Packages tab (the bottom-left area), click Install and select the necessary packages from CRAN separated with a space or comma, as follow



Getting Help on an R Package or any Built-in R Object

To get help on an installed and loaded package, or a function of an installed and loaded package, or any other built-in R object (such as a preloaded dataset), use one of the following syntaxes:

```
help(package_or_function_name)
```

```
help("package_or_function_name")
```

```
?package_or_function_name
```

Note: we need to pass in a function name to the help function without parentheses.

The Help tab will be opened with the package or object documentation. If we're checking a package, then we'll get the list of all its functions and the link to the documentation for each of them.

For example, run the following in the console (after making sure that the readr and dplyr packages are installed and loaded):

```
help("read.csv")
```

```
?readr
```

```
help(help)
```

```
help('CO2')
```

Importing Data

```
world_population <- read.csv("world_population.csv")
```

(To run the above piece of code, first, download the publicly available World Population Dataset from Kaggle and unzip it into the same folder where you store your R script.)

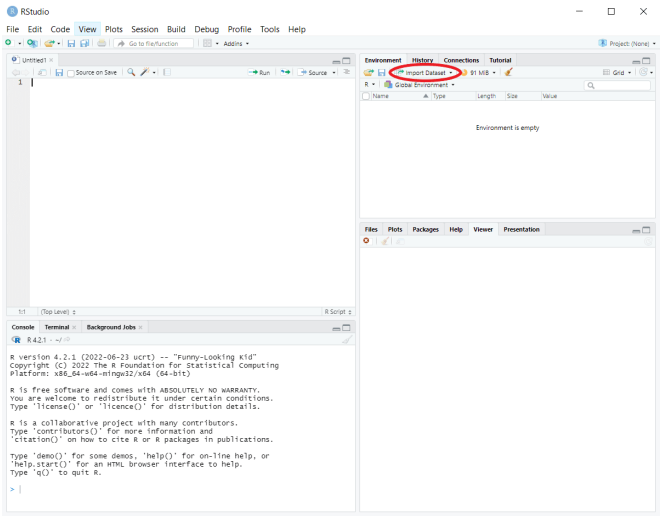
The result of running the above piece of code will be an R dataframe in your working folder.

In RStudio:

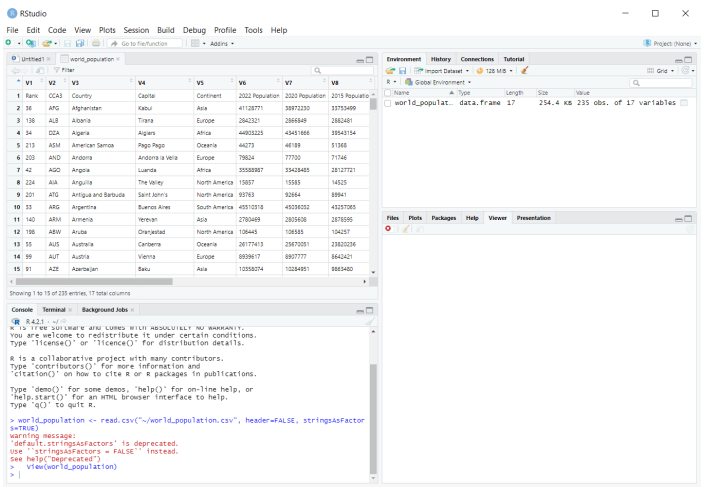
File – Import Dataset

OR

Click Import Dataset on the Environment tab:



You can now find and explore the imported dataset on the Environment tab and in a spreadsheet opened in a new tab:



Special Values

There are a few special values that are used in R.

NA

In R, the NA values are used to represent missing values. (NA stands for “not available.”) You may encounter NA values in text loaded into R (to represent missing values) or in data loaded from databases (to replace NULL values).

If you expand the size of a vector (or matrix or array) beyond the size where values were defined, the new spaces will have the value NA:

```
> v <- c(1,2,3)
> v
[1] 1 2 3
> length(v) <- 4
> v
[1]  1  2  3 NA
```

Inf and -Inf

If a computation results in a number that is too big, R will return Inf for a positive number and -Inf for a negative number (meaning positive and negative infinity, respectively):

```
> 2 ^ 1024
[1] Inf
> - 2 ^ 1024
[1] -Inf
```

This is also the value returned when you divide by 0:

```
> 1 / 0

[1] Inf
```

NaN

Sometimes, a computation will produce a result that makes little sense. In these cases, R will often return NaN (meaning “not a number”):

```
> Inf - Inf
[1] NaN
> 0 / 0
[1] NaN
```

NULL

Additionally, there is a null object in R, represented by the symbol NULL. (The symbol NULL always points to the same object.) NULL is often used as an argument in functions

to mean that no value was assigned to the argument. Additionally, some functions may return NULL. Note that NULL is not the same as NA, Inf, -Inf, or NaN.