**Chapter 1.  Quick Start**

Data analysis is difficult without the proper tools. It is almost impossible to extract patterns directly from a large set of numbers aligned in rows and columns and draw any conclusion, even for experts. A suitable tool, such as R, will remarkably boost your productivity in working with data. From my experience, learning a programming language is somehow like learning a human language. It is probably not a good idea to jump right into the details of vocabulary and grammar before looking at the big picture, getting motivated, and starting small. This chapter gives you a quick start by taking an overview of the R programming language in depth.

In this chapter, we will cover the following topics:

* Introducing R
* The need for R
* Installing R
* Tools required to write R code

As soon as the software and tools are ready to go, you will write a simple R program to experience how it basically works. Once this is done, the R journey will unfold from the basics to advanced techniques and applications.

**Introducing R**

R is a powerful programming language and environment for statistical computing, data exploration, analysis, and visualization. It is free, open source, and has a strong, rapidly growing community where users and developers share their experience and actively contribute to the development of more than 7,500 packages, so that R can deal with problems in a wide range of fields (refer to <https://cran.r-project.org/web/views/>).

Although the origin of the R programming language dates back to 1993, its general adoption in R programming language data-related research industry has grown rapidly in the last decade and has become the lingua franca of data science.

In general, R should be viewed as more than just a programming language; it is a comprehensive computing environment, a strong and active community, and a rapidly growing and expanding ecosystem.

**R as a programming language**

R, as a programming language, has been evolving and developing over the last 20 years. Its goal is quite clear to make it easy and flexible to perform comprehensive statistical computing, data exploration, and visualization.

However, ease of use and flexibility usually create conflicts. It can be very easy to click a few buttons to finish a variety of tasks in statistical analysis, but it won't be flexible if you need customization, automation, and your work needs to be reproducible. It can be very flexible to use tens of functions to transform data and make complicated graphics, but it won't be easy to learn and combine these functions correctly. R stands out for its well-positioned balance.

**R as a computing environment**

R, as a computing environment, is lightweight and ready to use. Compared to some other famous statistical software, for example, Matlab and SAS, R is much smaller and easier to deploy.

In this book, we will use RStudio to handle almost all our work in R. This integrated development environment provides rich features such as syntax-highlighting, auto-completion, package management, graphics viewer, help viewer, environment viewer, and debugging. These features hugely boost your productivity.

**R as a community**

R, as a community, is strong and active. You can visit Try R (<http://tryr.codeschool.com/>) immediately and get a first impression of R basics through an interactive tutorial. In practice, when you are coding, you probably won't solve every problem by yourself. You may google an R question and find that it almost always has answers in StackOverflow (<http://stackoverflow.com/questions/tagged/r>). If your question is not fully addressed, you can ask it and probably get an answer in a couple of minutes.

If you need to use a package but also want to see how it works in detail, you can visit the source code at its online repository (or repo). Many repos are hosted by GitHub ([https://www.github.com](https://www.github.com/)). In GitHub, you can do much more. When you find that a package is not working correctly, you can report a bug by filing an issue on the problem. If you need a feature that fits the purpose of the package, you can request a feature also by filing an issue for your demand. If you are interested in contributing to the package by resolving bugs and implementing features, you can fork the project, edit the code, and send merge requests so that your changes can be accepted by the owner. If your changes are accepted, congratulations, you have become a contributor to the package! Amazingly, R and its thousands of packages are built by contributors all over the world.

**R as an ecosystem**

R, as an ecosystem, is rapidly growing and expanding in all data-related areas beyond the IT industry. The majority of its users are not professional developers but data analysts and statisticians. These users may not write the best-quality code, but they may contribute cutting-edge tools to the ecosystem in R language, and everyone else has free access to these tools without having to reinvent the wheels.

For example, let's say an econometrician writes an extension package that includes a new method to detect a category of time series patterns; it may attract several users who find it interesting and useful. Some professional users may improve the original code to make it faster and more general-purpose. A while later, a quantitative investor may find it helpful to incorporate this method into a trading strategy because it can detect patterns that usually causes risks in his/her portfolio. At the end of the day, the econometrician's tool is applied in a real-world industry, and the investor finds the portfolio less risky.

That is how the ecosystem works. And that is one of the reasons why R rocks in these areas: it has the ability to quickly adapt cutting-edge knowledge outside the IT industry (usually data science, Academia, and Industry) to generally available and applicable tools in the ecosystem. In other words, it facilitates conversion from the field knowledge and data science to productivity and value.

**The need for R**

R stands out from a wide variety of statistical software for the following reasons:

* **Free of charge**: R is totally free. You don't need to buy a license, so there is no financial entry barrier to use it and most of its extension packages.
* **Open-source**: R and most of its packages are fully open source. Thousands of developers are constantly reviewing the source code of the packages to check whether there are bugs to fix or things to improve. If you encounter exceptions, you can even dig into the source code, find where the problem is, and contribute to fixing it.
* **Popular**: R is a very popular, if not the most popular, statistical programming language and platform to perform data mining, analysis, and visualization. High popularity often means easier communication between you and other users because you "speak" the same language.
* **Flexible**: R is a dynamic script language. It is highly flexible to allow programming styles in multiple paradigms, including functionality programming and object-oriented programming. It also supports flexible metaprogramming. Its flexibility enables you to perform highly customized and comprehensive data transformation and visualization.
* **Reproducible**: When using software based on a graphical user interface, you only need to choose from menus and click buttons. However, it is hard to accurately reproduce what you have done automatically without writing scripts.

In most scientific research areas and many industrial applications, reproducibility is necessary for many reasons. R scripts can precisely describe what you do with the computing environment and data so that it is fully reproducible from scratch.

* **Rich resources**: R has a huge, rapidly increasing number of online resources. One type of resource is extension packages. There are, at the time of writing this, more than 7,500 packages available at **CRAN** (short for **Comprehensive R Archive Network**), a world-wide network of mirror servers from which you can get identical, up-to-date, R distributions and packages.

These packages are created and maintained by more than 4,500 package developers in almost all data-related areas, such as multivariate analysis, time series analysis, econometrics, Bayesian inference, optimization, finance, genetics, chemometrics, computational physics, and many others. Take a look at CRAN Task View (<https://cran.r-project.org/web/views/>) for a good summary.

In addition to the enormous number of packages, there are also a great number of authors who regularly write personal blogs and Stack Overflow answers and share their thoughts, experiences, and recommended practices. Plus, there are a lot of websites specializing in R, such as R-bloggers (<http://www.r-bloggers.com/>), R documentation (<http://www.rdocumentation.org/>), and METACRAN (<http://www.r-pkg.org/>).

* **Strong community**: The community of R consists of not only R developers but also, (the majority), R users from a wide range of backgrounds such as statistics, econometrics, finance, bioinformatics, mechanical engineering, physics, medicine, and so on.

A great number of R developers actively contribute to open source projects or packages written in R. The goal of the community is to make data analysis, exploration, and visualization easier and more interesting.

If you are stuck in a problem in R, just google what puzzles you; probably, there are already some answers to your question. If not, just ask a question on Stack Overflow and you will get a response in a very short time.

* **Cutting-edge:** Many R users are professional researchers in statistics, econometrics, or other disciplines. Quite often, authors publish their new papers along with a new package that includes the cutting-edge techniques presented in the paper. Maybe it's a new statistical test, a pattern recognition method, or a better-tuned optimization algorithm.

No matter what it is, the R community has the privilege of applying cutting-edge data science knowledge in the real world  often ahead of everyone else, improving its functionality and revealing its potential.

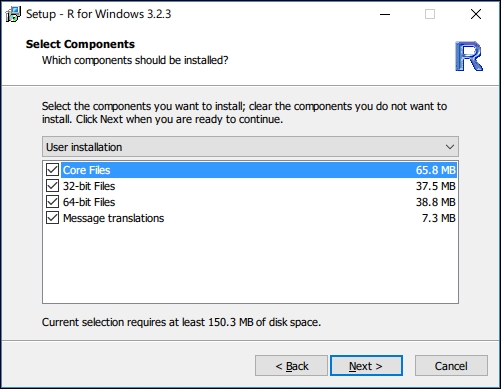
**Installing R**

To install R, you need to visit R's official website (<https://www.r-project.org/>), download R (<https://cran.r-project.org/mirrors.html>), choose a nearby mirror, and download a version for your operating system. At the time of writing, the latest version is 3.2.3. The examples in this book are created and run under this version in Windows and Linux, but there should not be significant differences between the output in previous versions or other supported operating systems.

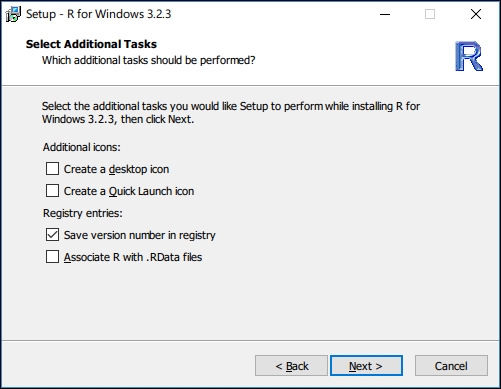
If you are using Windows, just download an installer for the latest version. To install R, run the Windows installer that you just downloaded. The installation process is easy to handle, but many users may still face problems with several steps.

In the Windows drop-down, when choosing the components to install, the installer lists four components. Here, **Core files** means the core libraries of R, and the **Message translations**component provides many versions of translations of warning and error messages in a list of supported languages. However, what may confuse you is the **32-bit files** and 64-bit files options. Just don't worry; you only need to know that 64-bit R can handle much more data in a single process than its 32-bit counterpart. If you are using a modern computer purchased in recent years, it is most likely to support 64-bit programs and should be running a 64-bit operating system, so the default option will be 64-bit files. If you are using a 32-bit operating system, unfortunately, you cannot use 64-bit R unless you install a 64-bit system if your hardware supports it.

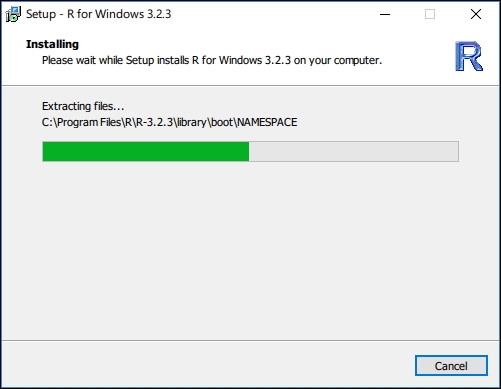
Anyway, I recommend that you install the default options, as shown in the following screenshot:



Another option you may feel confused about is whether to save the R version number in the registry. Checking these options makes it easier for other programs to detect which R version is installed. If you are sure you only use R in its own, just go ahead with the defaults.



Then, the installation starts copying files to your hard drive.



Finally, R is deployed to your computer. At the moment, you only have two ways to use R: In a command prompt (or terminal) or in the R GUI.

If you allowed the installer to create program shortcuts on your desktop, you will find two R shortcuts there. R runs in the Command Prompt and RGUI runs in an extremely simple GUI.

Although you can start to use R right now, it does not mean you have to use it in this way. I strongly recommend RStudio for editing and debugging R scripts. Actually, this book is also written in R Markdown in RStudio. Although RStudio is powerful, it does not work without a proper installation of R. In other words, R is the backend and RStudio is a frontend that helps you better work with the backend.

If you are using Windows, you may also install Rtools (<http://cran.rstudio.com/bin/windows/Rtools/>) so that you can write C++ code, compile it, and call it in R, and you can install and compile packages that contain C/C++ code from their sources.

**A quick example**

In this section, I will demonstrate a simple example of computing, model fitting, and producing graphics by typing in commands in the console.

First, let's create vector x of 100 normally distributed random numbers. Then, create another vector yof 100 numbers, each of which is 3 times the corresponding element in x plus 2 and some random noise. Note that <- is the assignment operator, which we will cover later. I use str() to print the structure of the vectors:

x <- rnorm(100) y <- 2 + 3 \* x + rnorm(100) \* 0.5 str(x)

## num [1:100] -0.4458 -1.2059 0.0411 0.6394 -0.7866 ...

str(y)

## num [1:100] -0.022 -1.536 2.067 4.348 -0.295 ...

Since we know that the true relationship between *X* and *Y* is A quick example, we can run a simple linear regression on the sample *X* and *Y* and see how the linear model recovers the linear parameters (that is, *2* and *3*) of the model. We call *lm(y ~ x)* to fit such a model:

model1 <- lm(y ~ x)

The result of the model fitting is stored in an object named model1. We can view the model fit by simply typing model1 or explicitly typing print(model1):

model1

##

## Call:

## lm(formula = y ~ x)

##

## Coefficients:

## (Intercept) x

## 2.051 2.973

If you want to see more details, call summary() with model1:

summary(model1)

##

## Call:

## lm(formula = y ~ x)

##

## Residuals:

## Min 1Q Median 3Q Max

## -1.14529 -0.30477 0.03154 0.30042 0.98045

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 2.05065 0.04533 45.24 <2e-16 \*\*\*

## x 2.97343 0.04525 65.71 <2e-16 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 0.4532 on 98 degrees of freedom

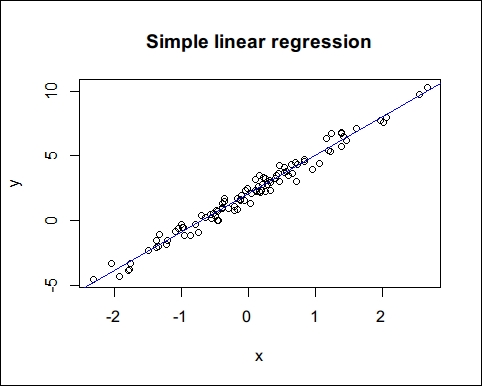
## Multiple R-squared: 0.9778, Adjusted R-squared: 0.9776

## F-statistic: 4318 on 1 and 98 DF, p-value: < 2.2e-16

We can plot the points and the fitted model together:

plot(x, y, main = "Simple linear regression")

abline(model1$coefficients, col = "blue")



The preceding screenshot demonstrates some simple functions so that you can get a first impression of working with R. If you are not familiar with the symbols and functions in the example, don't worry: the next few chapters will cover the basic objects and functions you need to know.

Code:

x <- rnorm(1:100)

y <- 2 + 3\*x + rnorm(100) \* 0.5

x

y

str(x)

str(y)

model <- lm(y~x)

model

summary(model)

#plot

plot(x,y, main = "Simple Linear Regression")

abline (model$coefficients,col = "blue")