

Chapter Three: The R and RStudio Environment

Simona Simona

19/02/2019

0.1 What is R and why is it important?

R is a statistical programming language and computing environment which has become a *lingua franca* for the statistical community worldwide and it is increasingly becoming popular in the social sciences too. There are many reasons why this is the case. Probably most importantly, R is free. Licenses for commercial software can sometimes be outside the reach of many students and academic staff. A free software, especially one which is also funny and powerful can come in handy for that demographic. R comes with a variety of statistical and visualisation capabilities but with high extensibility because it is open source. What that means is that R users have access to the source code and have the freedom to use, copy, change, study and improve it. Users have the choice to use an integrated suite of software facilities for data manipulation, analysis and graphics or use add-on packages written by other users and available for free or create their own packages if what is already available isn't satisfactory enough. Of course, one requires some programming know how to do that.

At the time of writing this chapter, there were 13,741 available packages on the [cran](#) package repository. The availability of free add-on packages make R more powerful than other proprietary software packages. Proprietary packages like SPSS, Stata, SAS etc, are owned by organisations which developed them. SPSS is owned by IBM, Stata by stataCorp while SAS is owned by the SAS Institute Inc. These organisations hold the property rights for the source code which is very closely guarded and only accessed by a few people or a team of individuals in those organisations. As such, any innovation and improvement to the software can only be done by or in agreement with that core team and this of course limits the extent of freedom and innovation.

Contrast that with an open source software where any nerdy person out there can access the source code and has the freedom to add a package with specific functionality. That's why it is often said that R can do anything—from the simplest to the most advanced statistical procedures. Because if you can imagine something, most likely someone has already created a package on it. You just need to browse through the cran repository and look for it. If it doesn't exist, you can create your own functionality which enables you perform any task specific to your needs.

There are several other reasons that make R very attractive. Many software packages have limits in terms of the number of variables in the dataset you import into the software, R is only limited by the memory capacity of your machine. This quality makes it a go-to statistical environment in the era of speedy, big volume and high variety data popularly known as big data. While most packages would only handle one dataset at a time, R is very happy to hold plenty of datasets in its environment simultaneously and a user is free to alternative between different datasets at the same time.

R is also very good with graphics. The `ggplot2` package makes it very attractive to create graphs in R and that reason alone has made a lot of people switch to using R. We shall detail this aspect in the graphics chapter. It is important to note also that knowing R makes learning both closed packages like SPSS and STATA and full object-oriented programming languages like Python and

C++ much easier. Beginning with R, which has a much steeper learning curve makes the hustle of learning closed packages more tolerable and on the other hand, beginning with R introduces you to object-oriented programming used by the majority of programming languages. As such, if you chose to throw yourself in the mix of programming, you may just find yourself in familiar grounds before long and live happily thereafter.

The R community is expansive and It attracts many users who share ideas online through books, online courses, R users groups, conferences and Q&A websites. Having access to the internet is almost the sole prerequisite of being an expert in R. If you are stuck with something, there is almost a 100% chance that someone else faced a similar problem in the past and they posted a question about the same online and received helpful answers from the community. Make Google your friend and search for solutions to any problem you may be experiencing and you will be surprised how much information there is out there. If for some reason you don't find what you are looking for, you are also free to ask your own questions. [Stack overflow](#) is the most popular site for this purpose. Even if you may feel terrible in the beginning, just know that you are not alone, you have a large community of like-minded people who are eager to help you probably even more than you are eager to ask.

I will introduce R Markdown later but I have mention that it has been a game changer especially for me. It provides an environment which makes R useful even beyond data analysis and visualisation. It allows researchers to combine the statistical and text components of their projects to create beautifully looking documents in the form of research reports, teaching documents, journal articles and presentations. Additional packages like bookdown and blogdown allows you to create books and websites respectively within the R environment. I don't think it can get any cooler than that. You might want to know that this very document you are reading, my entire PhD thesis and my personal [website](#) were created using R Markdown. I have never written any HTML, CSS or JavaScript code in my life¹. This means you don't only have to use R for data analysis or graphics but for a host of other non-statistical purposes too. Even if you are a qualitative researcher, you may desire to distinguish yourself by creating cool presentations, journal articles or blog posts. In which case R is your good friend and learning it will take you a few steps ahead of your peers.

Yes indeed, I am going to talk about jobs especially for those who are still in college! Data science has been described as the sexiest job of the 21st century and R is one of the prominent tools in the data science community. Learning R puts you in the bracket of people with highly valuable skills especially now when we literally have an explosion of data infrastructure worldwide. I know that for social scientists like me, this may sound far fetched and intimidating. It is not. But I would say that even if it is not your intention to be a high flying data scientists, learning R puts you at an advantage in comparison with other graduates in the job market. At the very least you will be able to distill information from Quantitative studies and that increases your scope and exposes you to more literature in your field.

0.2 Origins

I'm really not a fan of history, but I am a fan of acknowledging people who have helped me in my life. R was created by Ross Ihaka and Robert Gentleman. Every R course you will attend, the instructor is likely to tell you that the R name came from the first letter of the first names of the two nerds. Well, this is partly true and since I like this reason better, I didn't want to bother finding

¹I am told these are the programming languages you need to learn to create a website

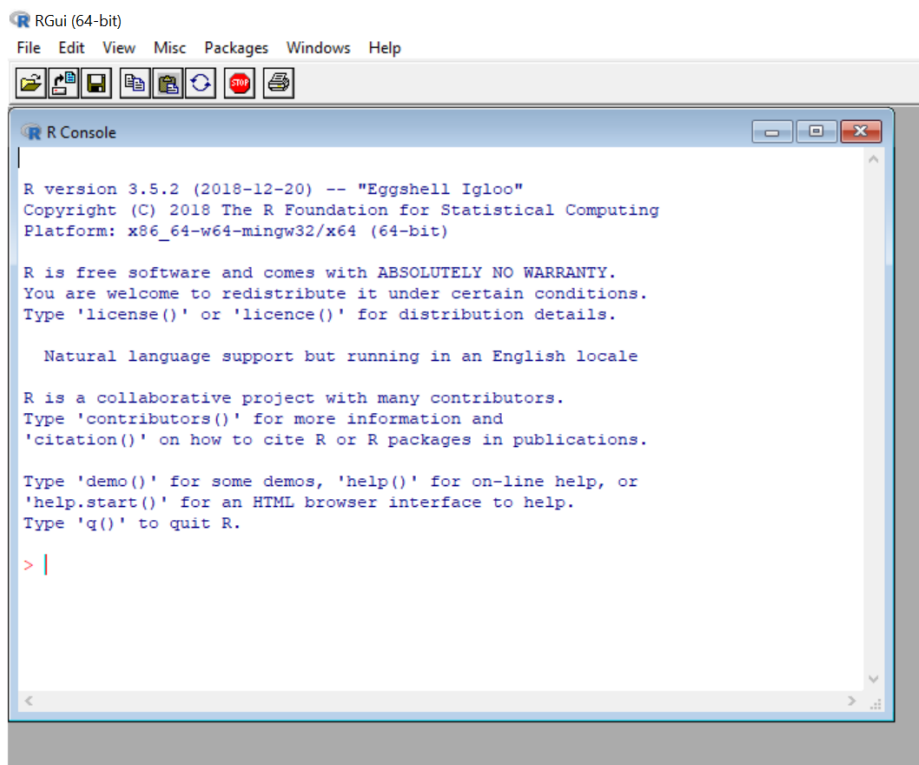


Figure 1: The R interface

out what the other part of the reason was. They were at the University of Auckland at the time. R is an implementation of the S language² which was created by John Chambers at the AT&T's Bell Labs in Texas in the 1970s. Now it is maintained and coordinated by the R Development Core Team, which is a committed group of volunteers across the world.

Fig 1. is the R interface and as you can see it is not very appealing and this has been partly why not so many social scientists have been enthusiastic about the software until quite recently. One of the major reasons for the surge in the use of R is the development of RStudio which had its initial release in 2011. RStudio is an open source integrated development environment (IDE). It provides a much more user friendly environment than the original R interface. From now onward when we refer to R, we will exchangeably be referring to R and RStudio unless so specified. RStudio continues to provide powerful tools that aid the usability and operationalisation of R not only for data manipulation, data analysis and visualisation but also for non-statistical purposes like creating reports, journal articles and presentations through R Markdown.

We shall learn how to download RStudio in this chapter and inspect it's environment and then move on explore basic functionality including basic arithmetic, objects, vectors, data frames, functions and arguments. But we need to note before that, that you always need to install the original R first before RStudio and that RStudio will not work without R. I always compare the relationship between R and RStudio to that of a an engine and car body. The body of a car provides a user friendly environment for you to comfortably drive a car. You don't even have to know what is

²Don't worry if you don't understand the meaning of that statement. I don't either, so it makes the two of us, probably even more of us



[\[Home\]](#)

Download

[CRAN](#)

R Project

[About R](#)

[Logo](#)

[Contributors](#)

[What's New?](#)

[Reporting Bugs](#)

[Conferences](#)

[Search](#)

[Get Involved: Mailing Lists](#)

[Developer Pages](#)

[R Blog](#)

R Foundation

[Foundation](#)

[Board](#)

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To [download R](#), please choose your preferred [CRAN mirror](#).

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.

News

- [R version 3.5.3 \(Great Truth\) prerelease versions](#) will appear starting Friday 2019-03-01. Final release is scheduled for Monday 2019-03-11.
- [R version 3.5.2 \(Eggshell Igloo\)](#) has been released on 2018-12-20.
- The R Foundation Conference Committee has released a [call for proposals](#) to host useR! 2020 in North America.
- You can now support the R Foundation with a renewable subscription as a [supporting member](#)
- The R Foundation has been awarded the Personality/Organization of the year 2018 award by the professional association of German market and social researchers.

Figure 2: The R project website

going on under the hood for you to enjoy the ride but of course, the car would never run without the engine unless you are witnessing an apocalyptic miracle. We will thus start by finding R and downloading it and then move on to RStudio.


0.3 Finding and Installing R

I will spare you from languages like distributed binaries which are usually thrown around when people talk about installing R because they may be alien to you and quite frankly I don't understand them myself and so even if I wanted to, I really can't. Downloading and installing R is not very different from any software. R can be downloaded from the Comprehensive R Archive Network (CRAN). You can Google the R project or use this website: <https://www.r-project.org/>. The first 2 lines under the getting started subtitle, click on the highlighted **download R** or the highlighted **CRAN mirror** at the end of the paragraph. See Fig. 2 for this. When you open the CRAN mirror page, you should see that it resembles Fig. 3 and browse through the countries on left hand side of the screen and click on the city nearest to where you are, currently. You don't have to be completely accurate here. Essentially, whatever city you choose should work just fine. After this process you should now see Fig 4. Choose your operating system accordingly. If you are using a Windows please don't choose Mac. It is that simple! It should be pretty straightforward from here. On the next page click on **install R for the first time** if you are using a Windows machine. Then you should see the latest version of R on the next page **Download R 3.5.2 for Windows**(this is true at the time of writing the chapter. The latest version is expected to change with time).

If you are using Mac, after fig.4 and you have selected appropriately you should see a window resembling figure 5. The latest version of R should appear on your left hand side marked **R-3.5.2pkg**. If you click on that, you should get your installer and can launch it accordingly as you would with

CRAN Mirrors	
The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: main page , windows release , windows old release .	
If you want to host a new mirror at your institution, please have a look at the CRAN Mirror HOWTO .	
0-Cloud	Automatic redirection to servers worldwide, currently sponsored by Rstudio
https://cloud.r-project.org/ http://cloud.r-project.org/	Automatic redirection to servers worldwide, currently sponsored by Rstudio
Algeria	
https://cran.usthb.dz/ http://cran.usthb.dz/	University of Science and Technology Houari Boumediene University of Science and Technology Houari Boumediene
Argentina	
http://mirror.fcaglp.unlp.edu.ar/CRAN/	Universidad Nacional de La Plata
Australia	
https://cran.csiro.au/ http://cran.csiro.au/ https://mirror.aarnet.edu.au/pub/CRAN/ https://cran.ms.unimelb.edu.au/ https://cran.curtin.edu.au/	CSIRO CSIRO AARNET School of Mathematics and Statistics, University of Melbourne Curtin University of Technology
Austria	
https://cran.wu.ac.at/ http://cran.wu.ac.at/	Wirtschaftsuniversität Wien Wirtschaftsuniversität Wien

Figure 3: The R CRAN Mirror Page



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

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

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

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 (2018-12-20, Eggshell Igloo) [R-3.5.2.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.

Figure 4: The R Platform distributions

any other software.

I don't have instructions for Linux. I don't know why but somehow something is telling me that if you are using the Linux system, you are probably smart enough to figure the whole downloading business yourself. I know that I am wrong here and so, I will endeavour to provide instructions on my website for this.

0.4 Installing RStudio

It is worthy repeating that RStudio is the platform we shall be using in this book but it should be installed only after you have followed the instructions above and have installed R accordingly. RStudio is found on this website: <https://www.rstudio.com/>. Or you can google it which should eventually take you to the same website. When it opens, you will see a few RStudio products listed, mainly RStudio, Shiny and R Packages. Under R Studio, click the Download button which should take you to the RStudio download page as shown in fig. 6. There are different licences that RStudio offers, some of which are commercial. They have got to make some money because they are a company. The open source versions of RStudio are more than enough for us and since we have



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

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

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

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

R for Mac OS X

This directory contains binaries for a base distribution and packages to run on Mac OS X (release 10.6 and above). Mac OS 8.6 to 9.2 (and Mac OS X 10.1) are no longer supported but you can find the last supported release of R for these systems (which is R 1.7.1) [here](#). Releases for old Mac OS X systems (through Mac OS X 10.5) and PowerPC Macs can be found in the [old](#) directory.

Note: CRAN does not have Mac OS X systems and cannot check these binaries for viruses. Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

As of 2016/03/01 package binaries for R versions older than 2.12.0 are only available from the [CRAN archive](#) so users of such versions should adjust the CRAN mirror setting accordingly.

R 3.5.2 "Eggshell Igloo" released on 2018/12/20

Important: since R 3.4.0 release we are now providing binaries for OS X 10.11 (El Capitan) and higher using non-Apple toolkit to provide support for OpenMP and C++17 standard features. To compile packages you may have to download tools from the [tools](#) directory and read the corresponding note below.

Please check the MD5 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type
md5 R-3.5.2.pkg
in the *Terminal* application to print the MD5 checksum for the R-3.5.2.pkg image. On Mac OS X 10.7 and later you can also validate the signature using
pkgutil --check-signature R-3.5.2.pkg

Latest release:

R-3.5.2.pkg
MD5-check: 0c4e0f980b179132b0d524545f63e
SHA1-hash: c7a0b00a0b06c345ac968872a0c2b0466a2c5b7
(ca. 74MB)

R 3.5.2 binary for OS X 10.11 (El Capitan) and higher, signed package. Contains R 3.5.2 framework, Rapp GUI 1.70 in 64-bit for Intel Macs, Tcl/Tk 8.6.6 X11 libraries and Texinfo 5.2. The latter two components are optional and can be omitted when choosing "custom install", they are only needed if

Figure 5: The R for MacOS

marketed R so well because of it being an open source software, we should equally emphasise that the free version of RStudio offers you literally everything you need. So, unless you are a business and you think you are likely to benefit more from the commercial licences, or you feel the urge to contribute something to RStudio for the wonderful job they are doing, you probably want to stay on the free side.

Under the free licence, you can either click on the download button or scrawl down to the installers for supported platforms section. Select the installers according to your operating system and follow the straightforward instructions to install the software. After this process we are basically good to go. R and RStudio should be appearing on the Start menu for Windows or the Applications folder for Mac. Don't ask me about Linux or Ubuntu operating systems, I have no idea. As I said, we are going to be using RStudio and so, at this point you can double-click on it's icon to open the environment, which takes us to the next section.

0.5 The RStudio Environment

When you open RStudio, we are essentially in a new R session and the window you get should look like Fig.7. The new session of RStudio has 4 windows, although only 3 will appear initially. We will get to the fourth one shortly. The main window on the left hand side, is the **console** and this is where the main outputs will be shown. The right hand side has two windows and each one of them has a few tabs. The top right window has 3 tabs. The **Environment** is a place where your data frames, values and functions that are imported or created within RStudio are shown. The **History** tab keeps a record of all the commands we have used during our current session. You may not use the **Connections** tab but it allows you to connect 2 databases including relational ones. It is definitely for advanced users. The bottom right window has multiple tabs. It is the area you should see your files that are in your working directory under the **Files** tab. **Plots** is the place where the plots executed in the console or the r script (to be discussed later) will appear. A list of all packages (we shall discuss this shortly) installed on our computer will appear in the **packages** tab. The **Help** tab displays documentations for any R functions or packages that you have sought further information on. RStudio allows you to seek help about a particular function or package by typing the relevant code in the R console or source script and the results will be shown in the help tab. Many people may not even do this as they have Google for friend. They would rather consult

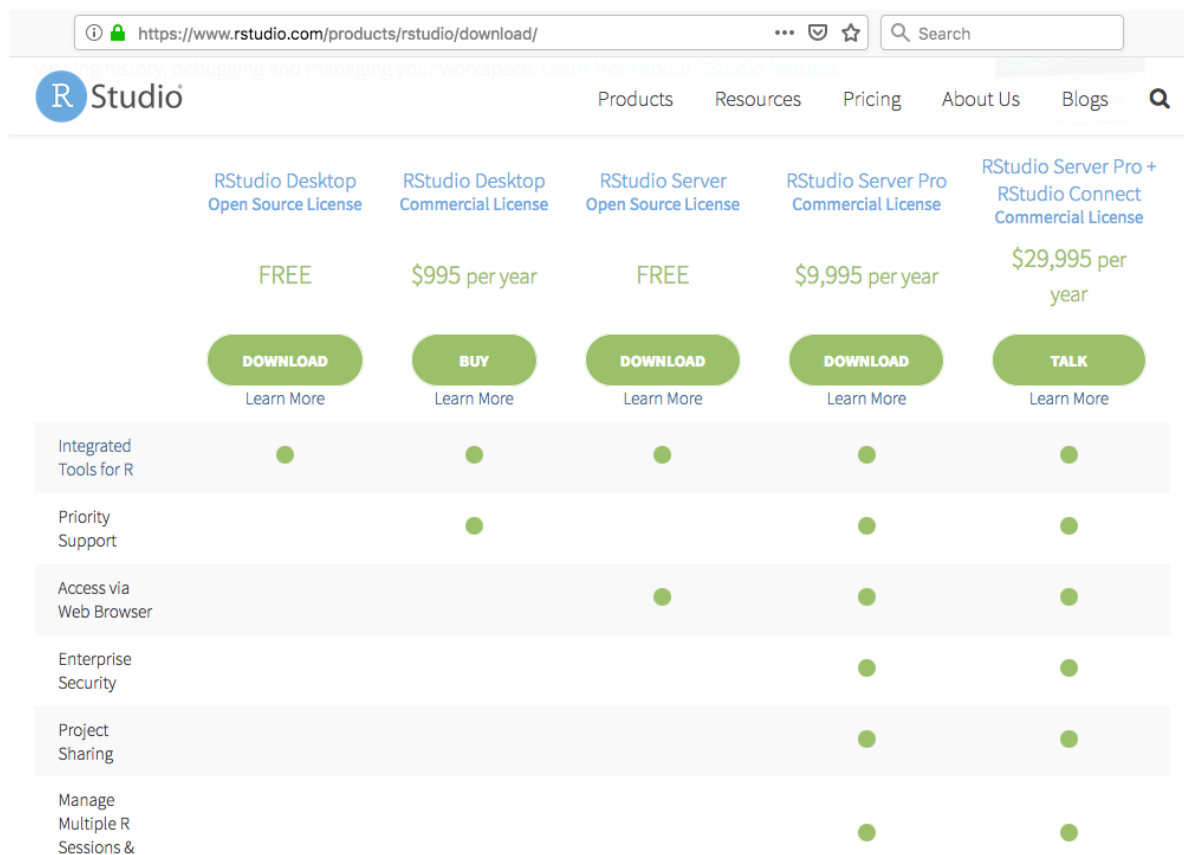


Figure 6: The RStudio Download Window

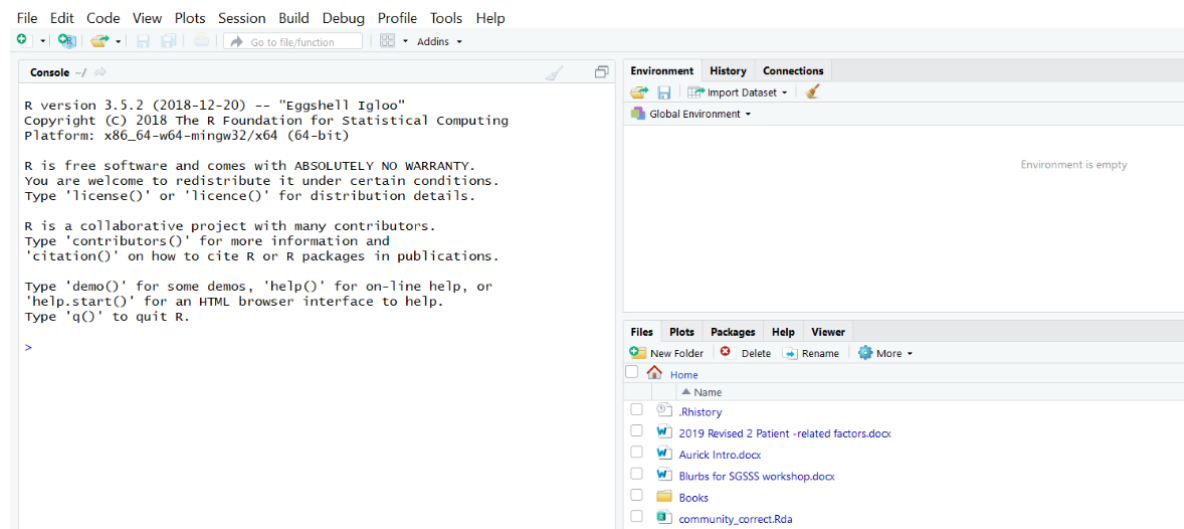


Figure 7: The RStudio Download Window

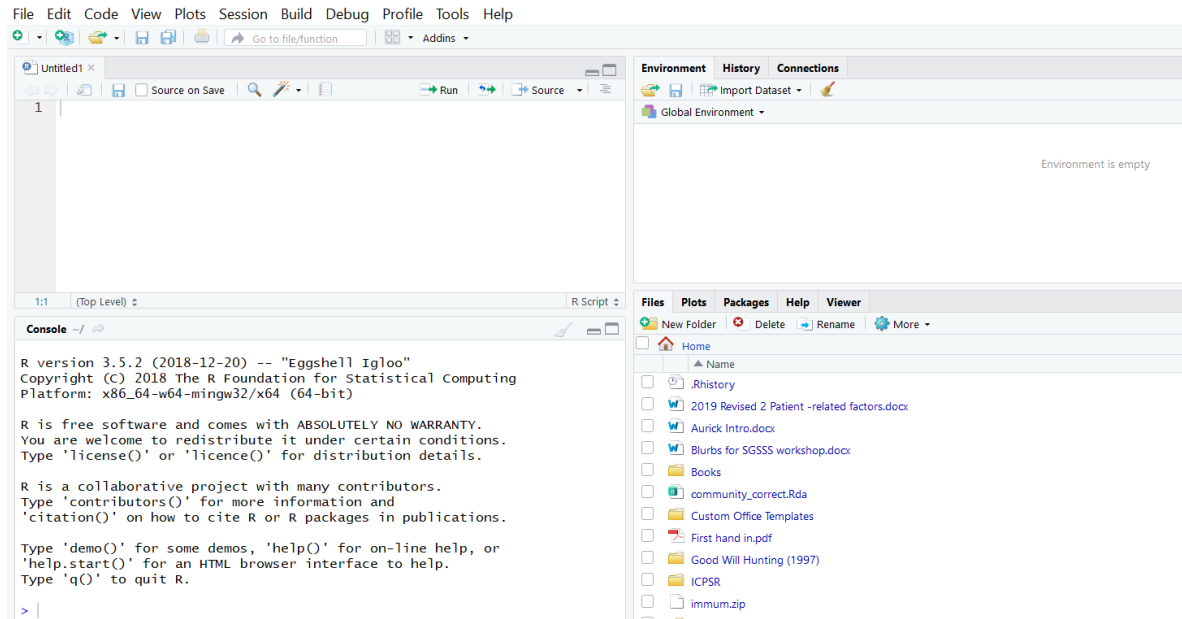


Figure 8: The RStudio Download Window

Google so that they have options in terms of which website to read the documentations from and also the webpage might have a better resolution compared to the RStudio tab. Finally we have the **Viewer** tab which displays the web content locally. This is useful when you want to use the more advanced facilities in RStudio to create websites using the **blogdown** package or interactive web applications using **RShiny** respectively. You will be able to preview them using this tab before launching on the internet. When I was creating my website in RStudio, I was able to see it in this tab before deploying it on the web.

Now it is time to go back to the fourth window and it is called the **source** or **script** editor. The R console can be used to write and execute code as highlighted earlier but the code written there cannot be saved or replicated. That's why we need the script window. It is the top left window in RStudio. It supports many forms of script files but here of course we are dealing with **.R** files. It does not always open when you launch RStudio for the first time. To open it, you use the **File** → **New File** → **R Script** menu. The File menu is located in the left hand corner of RStudio. The source script window is where we type our code to be executed in the console. We can most importantly document our code and save it for future use. For those of you familiar with **Stata**, this is similar to the do-file. The full RStudio interface looks like the figure above (Fig.8). This window is going to be home for any serious analysis. We can save the code we write here for our analysis or graphics and come back to it later. Not only that, we can easily correct errors in our code. The script editor comes with handy productivity enhancing features such as auto completion, syntax highlighting, multiple-file editing and find/replace. To run an R code, you just need to click anywhere on the line of code you want to run and then click the **Run** button on the top right corner of the **source script** window. Alternatively you can use **Ctrl + Enter** if you are using a windows machine or **Cmd + Enter** on a Mac.

0.6 The Basics

0.6.1 Working directory

Again, for those of you familiar with Stata, this should be pretty straightforward. A working directory is a folder which contains all the R files we are working with in an R session. It doesn't have to contain only R files, it can have all sorts of other files and we don't necessarily have to use it but using it is quite handy. It is used when we want to read in (or import) data into R or save (export) data from R into our computer. Every new R session is connected to a working directory and all you need is to find out where it is and change it to your preferred directory if you need to. To check where your directory is located, you use the function `getwd()` which means Get Working Directory and to set it to your preferred directory you use the `setwd()` function, which means Set Working Directory. I am using a Mac computer and my working directory is the `R_zambia` folder on my computer. The first thing I want to do is to locate my current working directory and then since the current working directory that R is pointed to is not my preferred folder, I will direct it to my preferred directory in the next code chunk. The output from this function shows that my working directory is the folder named **Quantitative social science**. I know because it is the one which appears at the end of the path.

```
getwd()
```

```
## [1] "/Users/simonasimona/Book projects/Quantitative social sciences"
```

I can now change my working directory and to do so, I use the following code and specify the path:

```
setwd("/Users/simonasimona/R_zambia")
```

My new directory is now the `R_zambia`. Now if I run the `getwd()` function, it should be pointed to the `R_zambia` folder. On windows, you can easily pick the path to the working directory by right clicking on your preferred folder and choosing **properties** on the drop down window. You should be able to see it under the **Location** option. Copy and paste it in the `setwd()` function. Be sure to enclose it in parentheses and change the slashes from back to forward.

There is another way to do this, which is what I actually use because I don't want to suffer. On the menu, pick **Session** and in the drop-down window, hover on 'Set Working Directory' and select 'Choose Directory', then you can navigate your way to your preferred folder accordingly and select it. This is simple and straightforward.

0.6.2 Arithmetic

We learn R because we want to do complicated things with it but like I said you can use R to really do anything you want. We will demonstrate this by using R as a calculator. This is also a good way to introduce the R syntax. Each line in R is a new command. I have already shown you how to run a code.

```
2+3
```

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

```
3+8
```

```
## [1] 11
```

```
10-7
```

```
## [1] 3
```

```
4*4
```

```
## [1] 16
```

```
3/3
```

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

0.6.3 Objects

R is part of languages which are often collectively called object-oriented, which basically means all data, values, models, outputs, etc, are all stored in memory as objects (Fogarty, 2018). The arithmetic calculations that we made earlier are not stored in memory and R will not remember them unless we run them again. Objects are containers where we can store any numbers, calculations, strings, models or datasets. We use the `<-` to do that. It is called an **assignment operator**. Once we assign any information in an object, the object will appear in the Global Environment pane. In order to display the contents of object we just need to call that object by running it. In the examples below, we assign the number 10 into the object *f*, the calculation $10 * 10$ into an object *x* and the name “Foster” into the object *y*. When we run the objects, we see the results.

```
f <- 10  
f
```

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

```
x <- 10*10  
x
```

```
## [1] 100
```

```
y <- "Foster"  
y
```

```
## [1] "Foster"
```

Notice in the example above that we put characters in parentheses. This is always a good practice in programming and often times even mandatory. Note also that R is case sensitive, meaning that if we saved the data in a lower-case *a*, then we can only retrieve it using the lower-case *a* and not upper-case A. If we exchange them, R is not going to like it very much. This is the case with everything in R whether we are talking about objects, functions or arguments.

We can also use object to do arithmetic. In the example below, we assign the operation $5 * 4$ into the object *a* and create the object *b* to *d* based on *a*.

```
a <- 5*4  
a
```

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

```
b <- a*2
b
```

```
## [1] 40
```

```
c <- a+b
c
```

```
## [1] 60
```

```
d <- c-b
d
```

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

0.6.4 Vectors and Data Frames

Vectors are the basic data structures in R. A vector is basically an array of elements of the same basic type. Elements of a vector can be of different types including numeric and strings. Elements of a vector are usually combined by the `c` function which means to concatenate or paste together. The following are examples of vectors:

```
x <- c(1,2,3,4,5)
x
```

```
## [1] 1 2 3 4 5
```

```
y <- c("January", "February", "March", "April", "May")
y
```

```
## [1] "January" "February" "March"    "April"    "May"
```

When we apply a mathematical operation to a vector like `a`, the operation will be applied to all the elements of the vector. For example, when we multiply `x` by 3, this applies to all the elements of `x` as follows:

```
x*3
```

```
## [1] 3 6 9 12 15
```

```
x
```

```
## [1] 1 2 3 4 5
```

We can also allocate a series of numbers using the `:` which is used when the numbers in question follow a certain sequence in which case you can begin from the smallest to the largest number or from the largest to the smallest

```
a <- 5:15
a
```

```
## [1] 5 6 7 8 9 10 11 12 13 14 15
```

```
b <- 5:-5
b
```

```
## [1] 5 4 3 2 1 0 -1 -2 -3 -4 -5
```

0.6.5 Data Frames

A data frame is a special kind of an object. It is a two dimensional matrix that R treats as a dataset. So when we speak about a data frame in R we basically mean a dataset. It has one or more columns with an equal number of rows. The columns are considered as variables while rows are observations. In this book, we shall deal more with data frames than any other data structures. Data frames can be created from vectors, created in R or imported from external sources. R is a very powerful platform but the only things it is clumsy about is entering data. You are encouraged to use other platforms like excel or SPSS to enter your data and only bring it in R for manipulation analysis and graphics. We shall learn about reading in (importing) data from different sources towards the end of this chapter. Turning a vector into a data frame, we use the `as.data.frame()`. We can do this on the object `a` above

```
as.data.frame(a)
```

```
##      a
## 1    5
## 2    6
## 3    7
## 4    8
## 5    9
## 6   10
## 7   11
## 8   12
## 9   13
## 10  14
## 11  15
```

Now we see that R has converted an R object from a vector to a data frame with 11 observations and corresponding values. The data frame `a` here has one variable and 11 observations.

0.6.6 Function and Arguments

In R, functions are objects which perform specific tasks. Functions are easily identifiable because they always end with an open and closed bracket `()`. Arguments are what goes inside the open and closed bracket and they control what functions do. For example the `seq()` function creates a sequence of numbers from a specific number to another specific number.

```
seq(20, 100)
```

```
## [1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
## [18] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
## [35] 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
## [52] 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87
## [69] 88 89 90 91 92 93 94 95 96 97 98 99 100
```

In the example above, the numbers 20 and 100 are examples of arguments and they tell the `seq()` function that those sequence of numbers you want to create should start from 20 up to 100. Functions do not always have to have arguments. There are functions without arguments and the best example here is the `getwd()` function when we have come across earlier on. Functions apply to other types of data structures including vectors, matrix and data frames. We shall mainly use functions on data frames in this book. Actually being an expert in R means knowing as much functions as possible. Of course it is not possible to know all the functions you need. It is for this reason that R has a facility to seek further information about functions and their arguments. To get further information about an R function you put `?` before a particular function. For example `?rep` will produce documentation on the `repeat` function. When you run this, documentations will pop up in the help tab on the bottom right hand window of the RStudio environment. But you can also do this in Google. It is even better if you ask me because it will provide you with plenty of options.

0.6.7 Packages in R

R packages are a collection of R functions, compiled code and sometimes sample data. R automatically installs a set of packages during installation. More packages are added later, when they are needed for some specific purpose. When we start the R/RStudio session, only the default packages are available. Other packages which are already installed have to be loaded explicitly to be used by the R program that is going to use them. The `library(package name)` or `require(package name)` function are used to load the packages into R. To see all the packages in your computer, you can use the `library()` function without any arguments.

If we want to use different functions not in the base packages—that is, certain methods, techniques and procedures, we need to download and install additional packages. New packages are installed from the CRAN (The Comprehensive R Archive Networks) website. We use `install.packages()` function for this purpose with the package name as the argument. The name of the package should always be in parentheses and the exact name of the package has to be supplied. Remember that every additional package we download, we need to load it in the current session for us to work with it. Other than the `library()` and `require()` functions, we can load the packages by browsing the packages tab on the bottom right-hand window and checking the box against the package we need.

```
install.packages(package names)
```

Sometimes when you install R packages for the time, RStudio may ask for the package repository or the CRAN mirror you want to download the package from in a pop-up window. Usually you choose the one physically closest to you in order to ensure faster installation. Then R will now install the package. R gives out information while it is processing your installation which you might find boring. You can just wait for it to finish. A red stop sign near your console indicates that R is processing the installation. You will know it has finished when you see a statement about where the package has been saved on your computer and the blue `>` prompt at the end of the installation. R often gives out red warnings about the R version the package was built under, it is safe to ignore this warning as the package will most likely work just fine with the new version.

0.6.8 Installing Packages in R

The `foreign` package is very popular because it helps us read in different formats of data including SPSS, SAS and Stata into R. It makes sense that we start with such a package. Because like I

said, I think that R is cool but it is not fun to enter the data into the platform. You are definitely going to have to rely on existing datasets, i.e already entered elsewhere and only use R for data manipulation, analysis and graphing. That's why the foreign package becomes useful because through it we can bring in datasets of different formats. There are a few more packages that do this in R such as the `haven` and `readr` packages. These use different syntax and so, I would recommend that you stick with one to avoid unnecessary confusion. This is one of the down sides of open source platforms that I have discovered, there can be many conflicting packages because they tend to have the same functionality.

```
install.packages("foreign")
```

Now you can check the **Packages** tab. If it is appearing, then it has been downloaded and installed successfully on our machine. It is appearing on my machine. Then the next thing is to load it in our current R session. Please note that the installation of packages is done once. As long as you will be using the same machine, you don't to do this process again but you have to load the package every time you have a new session. I will repeat this, please load the packages using the `library()` function every time you open your computer to work with R. Another thing, you need to use parentheses in the `install.packages()` function and you don not have to do so for the `library()` function. I am emphasising this because it is one of the mistakes that new R users make.

If you are given problems with mirror issues when you are downloading a package for the first time, you can check through the CRAN website again like you did when you were downloading the R software and specify the mirror near you. If you want to choose South Africa and University of Cape Town for example, the CRAN mirror is on <http://r.adu.org.za/> and we can specify this in our `install.packages()` function.

```
install.packages("foreign", repos = 'http://r.adu.org.za')
```

It is time to load our package:

```
library(foreign)
```

Now we have loaded our package and if you check the **Packages** tab, you will find the box against it has been checked. We are now ready to use it and bring in data into R but this is the subject for the next chapter.

References

Fogarty, B. J. (2018). *Quantitative Social Science Data with R: An Introduction*. SAGE Publications Limited.