

# Using R as a Research Tool.

Susan Johnston: Susan.Johnston@ed.ac.uk

Demonstrators: Gergana Dalaskova, John Godlee.  
Hat-Tips to Kyle Dexter and The Coding Club.

November 1, 2017

## 1 Introduction

### 1.1 What is R?

**R** began its life in New Zealand in 1993 as a language and environment for statistical computing and graphics. It is an interpreted programming language, meaning that rather than pointing and clicking, the user types in commands. It is **free** and works across all platforms.

### 1.2 Why use R?

“This is R. There is no if. Only how.”  
-- Simon ‘Yoda’ Blomberg, R-help (April 2005)

Almost anything is possible in **R**. It is fast becoming the *lingua franca* of academic research and data science. It is used for:

- Processing and tidying data
- Statistical analyses
- Data visualisation (ggplot)
- Creating interactive web applications (shiny)
- Generating reports and presentations (knitr, slidify)
- Creating portable projects (RStudio Projects)

The analytical power of **R** lies in its many packages (11,172 at the time of writing). At least 300 of these are written for ecologists and evolutionary biologists. A list of packages are hosted on the Comprehensive R Archive Network (known as **CRAN**): <https://cran.r-project.org/>.

## 2 Getting Started: R and the RStudio Environment.

### 2.1 Installing R and RStudio.

R can be downloaded from the CRAN website. Whilst the CRAN download version provides a simple user interface, we recommend that R is run through the software RStudio. This is open-source, free, and available at <http://www.rstudio.com/>. [NB. It is important to install R first and RStudio second.]

### 2.2 Creating an R Project.

Using R Projects (`.RProj`) is not necessary, but we strongly recommend that you use them. The advantage using R Projects is that it allows easier file imports, improved reproducibility and collaboration. This is primarily because it tells R where to look for data files and scripts, meaning that a script can be run on machine to another without any problems.

We have provided you with the project `Intro_to_R.RProj`. Opening this file will open RStudio. On the Files tab in the lower right corner, you will see the files in the current working directory. This will be useful later when we tell R to load files. You can check the working directory by typing `getwd()`.

If you would rather not use projects, you can set the working directory by using the command `setwd()` or by selecting Session > Set working directory > Choose directory.

Creating an R Project is straightforward: select File > New Project and follow the instructions.

### 2.3 Using RStudio.

Open `Intro_to_R.RProj` and open the example R Script (File > Open File... > `1_Example_Script`). RStudio should look something like Figure 1.

On the lower left is the *Console* pane - this is the engine of R. You can give instructions to R by directly typing at the prompt (`>`).

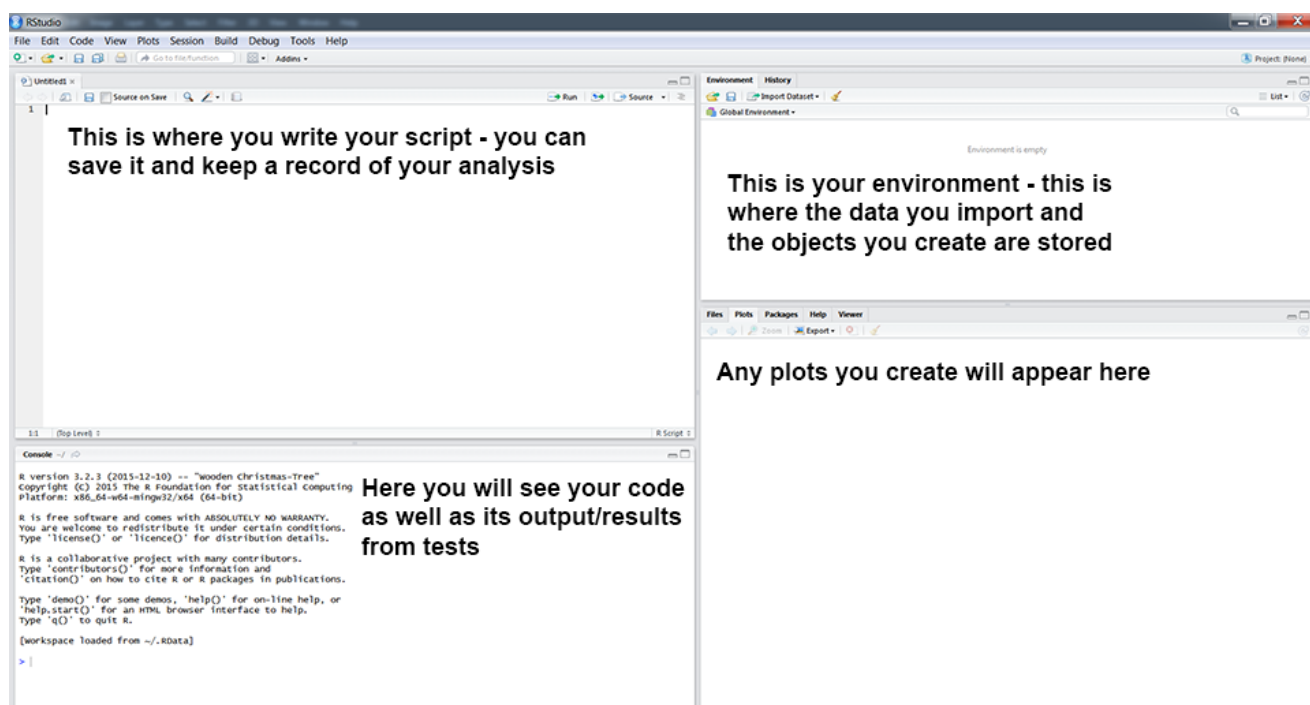


Figure 1: The RStudio Environment. Taken from OurCodingClub.io

On the upper left is your R Script - here, you can write commands and send them to the console by clicking “Run” or by typing **Ctrl-Enter** (or **Cmd-Enter**). Notice that all lines are preceded by **#** - this is the comment character in R.

On the lower right, you can browse the packages installed on your machine, open files and search R Help. This pane will also show plots when we run them later in the practical.

## Exercise 1.

Try running some basic commands directly in the console and from the R Script:

```
> 2+3
> 1:10
> seq(1, 20, 4)
> mean(c(3, 6, 9, 3, 6, 7))
```

Let's assign a sequence of numbers to an object, **x**:

```
> x <- 1:10
> x
> y <- seq(0, 4.5, 0.5)
> y
```

You can see that in the upper right pane, we can see this new objects **x** and **y** in the environment.

## 2.4 Finding Help within R.

The fastest way to find help in R is to search using **?**. For example:

```
> ?mean
```

should bring up a help page for the function **mean()** in the lower right corner. Typing two question marks:

```
> ??mean
> ??"standard error"
```

will search all help files and return a list of those that match.

### Exercise 2.

1. Using only **?** and/or **??**, find a function for calculating the standard deviation. What is the standard deviation of **x**?
2. Using **?**, find the help file for the **sort()** function. Sort **x** and **y** in reverse order.

## 2.5 Troubleshooting and finding help outside of R.

- Coding Club Tutorials & Useful Links <https://ourcodingclub.github.io/>
- Stack Overflow <https://stackoverflow.com/>: Try searching with the tag [R]
- RStudio Cheatsheets <https://www.rstudio.com/resources/cheatsheets/>

## 3 Loading and Preparing Data.

Now that we are familiar with the RStudio environment, it's time to start working with real data.

In the folder data, you have been provided with a single dataset on Peruvian Soil in two common formats - .txt (tab-delimited) and .csv (comma-delimited).

The easiest way to read the data into R is to use the Import dataset button in the Environment tab, selecting From Text (base) . . . and choosing either datafile (.txt or .csv). A box will appear (Figure 2). Check each of the options that applies to the data (i.e. change nothing) and click "Import". You will notice that the object Peru\_Soil\_Data is now in the R environment, but you may also have noticed a command appearing in the Console:

```
> Peru_Soil_Data <- read.delim("C:/Users/sjohns10/Google Drive/  
+ Teaching and Seminars/201711 NERC E3 DTP/Intro_to_R/data/  
+ Peru_Soil_Data.txt")
```

This command can be copied and pasted into your script, which would save you from clicking the next time. However, providing such a long path is problematic - if you were to rename one of the directories, or move the folder somewhere else, then the code would no longer work.

RProjects provide the solution to this. Try typing the following into your script, and guiding the command to the data file using the Tab key:

```
Peru_Soil_Data <- read.delim("
```

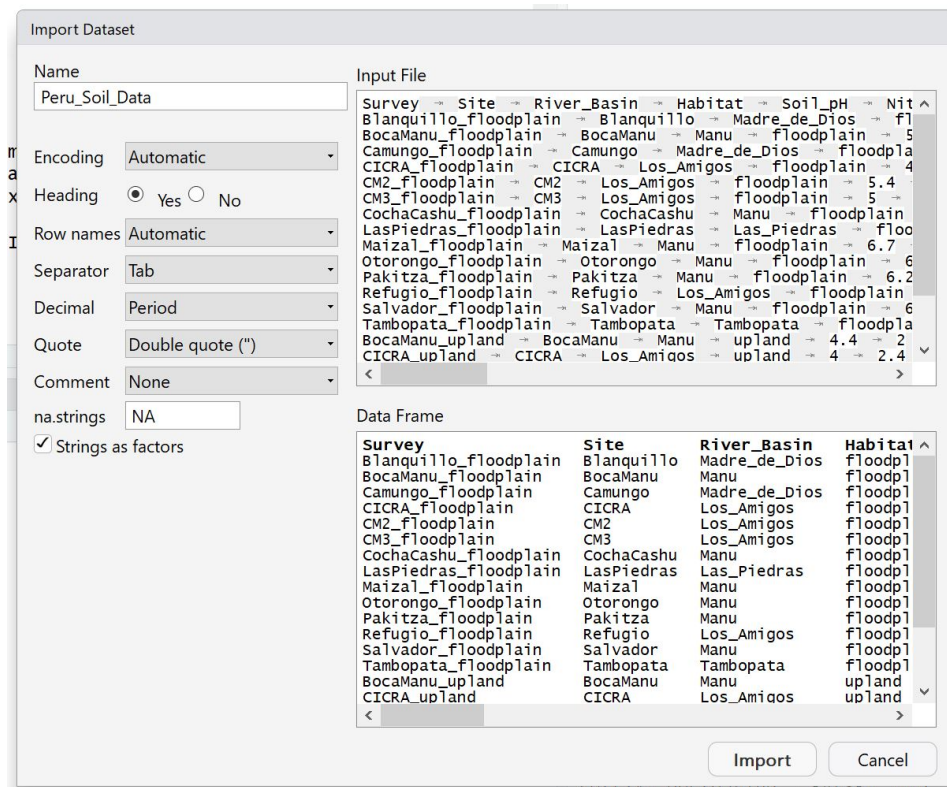


Figure 2: Importing data manually into R

You should now have the following code in your script:

```
> Peru_Soil_Data <- read.delim("data/Peru_Soil_Data.txt")
```