



## 1.2 Introduction to RStudio and packages

### Objectives:

- Understand the RStudio environment
- Understand how to interact with R
- Understand how to install packages

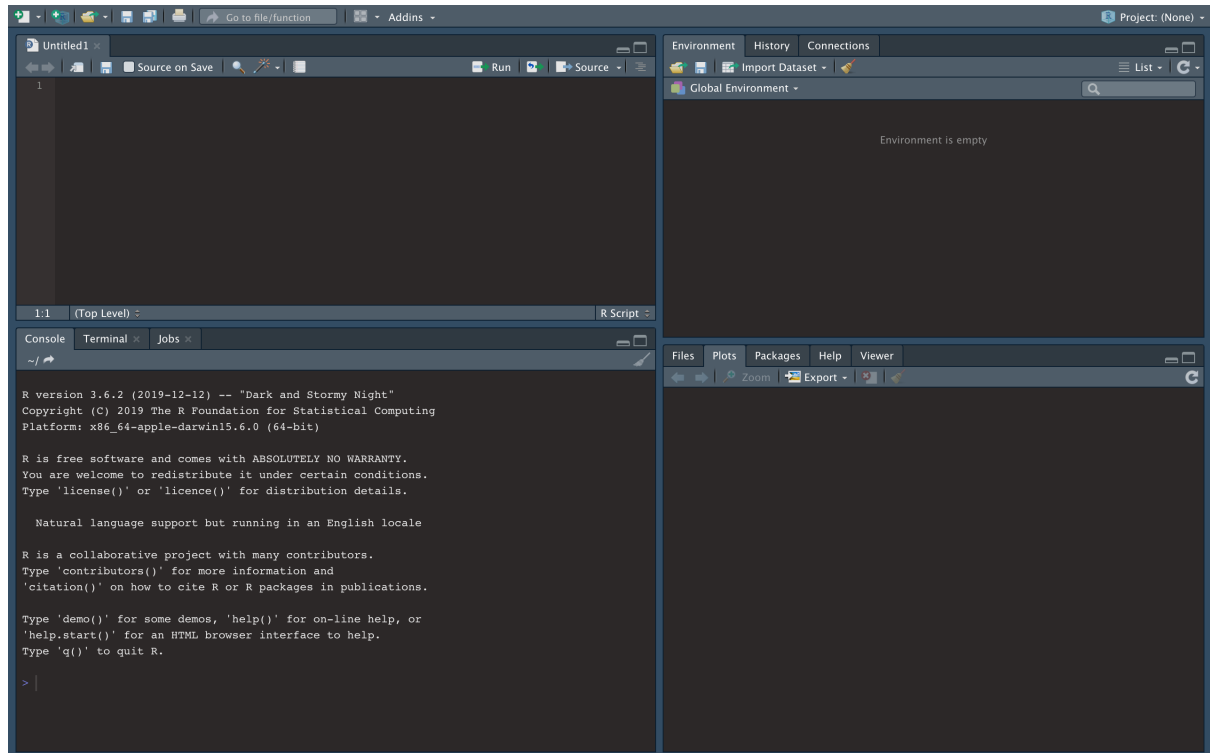
### Contents:

- The RStudio environment
  - Interacting with R
  - Packages
  - Scripts
  - Summary
  - Additional exercises
- 
-

# The RStudio environment

When you first open RStudio, you will see a screen like this:

There are 4 main sections to recognise here:



## 1. Source pane (top left)

The source pane allows us to create and edit R Scripts. Anything written here behaves like a text file (just saved as a .R executable file). Upon opening RStudio, it will automatically open a blank script.

Typing any commands in the source pane will not be evaluated when you press *"Enter"*.

To run **all** of the code in your script, you can press the *"Run"* button.

Alternatively, you can highlight and then press run if you just want to run a single line or section.

## 2. Console (bottom left)

The console in RStudio is where the code evaluates. In this section, pressing *"Enter"* will evaluate the code immediately.

Unless you are testing a single line, writing scripts in the source pane is always better. This is because the console doesn't save your output.

### 3. Environment (top right)

In the environment tab, you will see the names of all the data objects (i.e. functions and dataframes) you have defined. In the case of data, you will see some very basic information, such as the dimensions of the data.

### 4. Files/Plots/Packages/Help/Viewer

This window is very flexible, and you can use the mouse to select between the options.

**Files:** In the files pane, you can navigate the files on your computer. To set your working directory (where you want to read and write files from), you can go to a folder you want to read and save files to, click “*More*” and then “*Set As Working Directory*.”

**Plots:** This is where you will see the plots produced by your code. When you execute the code for a plot, you will see it appear here.

**Packages:** In this pane you will see a list of all the R packages installed on your computer. The checkbox indicates whether the package is loaded in the current session. We will discuss packages in more detail in the next section.

**Help:** Help menu for R functions. You can either type the name of a function in the search window, or use the code to search for a function with the name.

---

---

## Interacting with R

We are going to demonstrate some very basic but essential functions in R. The first command we will learn about is the `print` function. We can use `print` to print out variables (i.e. dataframes, tables) or strings.



### Exercise 1:

Let's write our first R programme, the famous 'hello world'.

```
print('hello world')  
  
[1] "hello world"
```

The second command we will introduce in this section is the `help` function. Using the help function in RStudio will bring up the documentation for the function. You can use `help` on any function to get information on its description, usage and arguments. For example, if we wanted the documentation for the `rnorm` function:

```
help(rnorm)
```

## The Normal Distribution

### Description

Often, we will need to generate numbers following a normal distribution. We can do this with the `rnorm` function, which will generate `n` numbers with a mean equal to `mean` and standard deviation equal to `sd`.

### Usage

```
rnorm(n, mean = 0, sd = 1)
```

### Details

If `mean` or `sd` are not specified, R assumes the default values of `0` and `1`, respectively.

### | Exercise

Call the `help()` function on the `print()` function

---

---

## Packages

As outlined earlier, the functionality of R lies in the thousands of packages available on the CRAN network. To use these packages, we need to install them. This process downloads the code onto your personal computer. We can use the following method to install any package on the CRAN network:

```
install.packages('ggplot2')
```

Once you have run this command, you will see lots of output, you can ignore all of this. After we have done this once, that's it, there is no need to run this command again.

If you want to actually use the package, you need to load it in your R session, and this step needs to be repeated each time you begin an R session.

```
library('ggplot2')
```

---

---

## Scripts

As you progress on your R journey, the programmes you write will become more complex. This means that you will want to understand how scripting works. In this section, we describe some important parts of writing good code. After completing an R script you can simply save by using

*file -> save as*

### Comments

Lines that begin with "#" are comments. If a line starts with a "#" R will simply act like it's invisible. This can be useful to remind you (or others) what you are doing or why you are doing it, for example:

```
# I am testing the print function
print('hello world')

[1] "hello world"
```

## Numeric output

Sometimes it can be hard to identify the position of an object in the output. The console output starts with a number(s) in brackets such as [1], [21], [41] etc., and this identifies the position of the specific number in a sequence. We can see this below:

```
> print(1:100)

[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
[21] 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
[41] 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
[61] 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
[81] 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
```

## Bugs

Almost every single time you use R, you will run into bugs and errors. This can be caused by many factors, including spelling of functions, data types, missing data etc. Here we outline some potential solutions.

### 1. Data types

We will touch on data types later in this course. For now, understand that R evaluates objects based on what it thinks they are. The number 1 and the string 'one' can not be added to produce 2. Below, we can see that  $1 + 1$  evaluates to 2, but when we treat 'one' as a character, we get an error.

```
> 1+1

[1] 2

> 1+'one'

Error in 1 + "one" : non-numeric argument to binary operator
```

Sometimes the error codes in R are cryptic. The best thing to do in this situation is to google the error code you get, you can guarantee that someone on [stackoverflow](https://stackoverflow.com) has had the same issue before. Here, is an example:

Stack Overflow

Products Customers Use cases Search...

Log in Sign up

2 Answers

active oldest votes

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

TEAMS What's this?

Free 30 Day Trial

Because your question is phrased regarding your error message and not whatever your function is trying to accomplish, I will address the error.

54

– is the 'binary operator' your error is referencing, and either `CurrentDay` or `MA` (or both) are non-numeric.

A binary operation is a calculation that takes two values (operands) and produces another value (see [wikipedia for more](#)). `+` is one such operator: `"1 + 1"` takes two operands (1 and 1) and produces another value (2). Note that the produced value isn't necessarily different from the operands (e.g., `1 + 0 = 1`).

R only knows how to apply `+` (and other binary operators, such as `-`) to numeric arguments:

```
> 1 + 1
[1] 2
> 1 + 'one'
Error in 1 + "one" : non-numeric argument to binary operator
```

When you see that error message, it means that you are (or the function you're calling is) trying to perform a binary operation with something that isn't a number.

Serverless crawling engineer - Salary well above market rate  
 Visa sponsor  
 aws-lambda amazon-dynamodb  
 View all 3 job openings!

Linked

2474 How to make a great R reproducible example

1 colvar in scatter3D return error "dim[2] - dim[1] : non-numeric argument to binary operator"

1 Error in numInClass[i]%%k : non-numeric argument to binary operator

-1 What syntax error is causing this specific error message?

1 Error Multiply all rows by rows

## 2. Spelling

When something is spelt wrong, you will get these errors: `Error: could not find function` or `Error: object 'blah blah blah' not found`.

## 3. Punctuation

Often extra spaces, brackets in the wrong place, missing a comma, or using a full stop rather than a comma will lead to errors. In the code below, I will try to create a vector using full stops instead of commas.



```
# this will produce an error
> x = c(1.2.3.4.5)
Error: unexpected numeric constant in "x = c(1.2.3"

# this will not :)
> x = c(1,2,3,4,5) # commas, not full stops!
```

## 4. R is busy (>)

When R does not respond to you pressing `'run'`. It's likely that R is 'busy'. For example, if you wanted the standard deviation of your sequence of numbers `(x)` you might run `sd(x)` but you get no output (not even an error). Most often because R isn't *ready* for new code.

R is either **Ready** (`>`) or it is **Waiting** (`+`) for you to finish the old code. To work out if R is ready or waiting, you need to look to your console. The `>` symbol means that R

is Ready and a  symbol means that R is waiting. If you see the  symbol, R won't evaluate it until you finish the last command.

In the example below, you can see that the command is not complete.

```
> print(  
+
```

To fix this issue, you just need to press the **escape** key.

---

---

## Summary

You learnt:

- What Rstudio is and how to use it
  - How to create a script and some common errors
  - How to download a package
  - How to ask for help in R
- 
- 

## Additional exercises

1. Use a # to comment a description of your script
2. Install and load the 'dplyr' package
3. Print the statement 'i have installed dplyr'
4. Get help on the dplyr functions 'mutate', 'filter' and 'arrange'
5. Save the script

You can find the solutions to these exercises in the github repository

---

---

Materials used in this course can be cloned directly by clicking [here](#). Alternatively, you can view the repository online:

| <https://github.com/RJODRISCOLL/Introduction-to-R>



For any help and advice, We can be contacted at:

| [R.ODriscoll@leeds.ac.uk](mailto:R.ODriscoll@leeds.ac.uk) [pspjo@leeds.ac.uk](mailto:pspjo@leeds.ac.uk)