

STAT40730 Data Programming with R (online)

Isabella Gollini

Lecture 1 - Introduction to R

Introduction to R

- Why R?
- How to install R and Rstudio
- Layout of Rstudio
- A first R session
- How to get help
- Introduction to functions

Slides



When R code is shown in the lecture slides it will be shown in a blue box. Eg.

```
2 + 2
```

```
## [1] 4
```

- The text in the blue background indicates what has been typed into R. The results are shown below the code lines.
- A computer monitor in the top right-hand corner indicates that there is an accompanying screencast.
- Screencasts can be found in the screencast folder on Blackboard.

Why R?

- R is a language and environment for statistical computing and graphics <https://www.r-project.org/>
- R was created by Ross Ihaka and Robert Gentleman in the early 90s at the University of Auckland New Zealand.
- R was developed from another statistical language S that was developed at Bell Laboratories by John Chambers and colleagues.
- R is free and open-source so many people are contributing to its development.
- R is superior in many ways to existing commercial products such as SAS, SPSS or Stata.
- It is available for Windows, Mac and Linux
- R can be used online with RStudio Cloud <https://rstudio.cloud/>

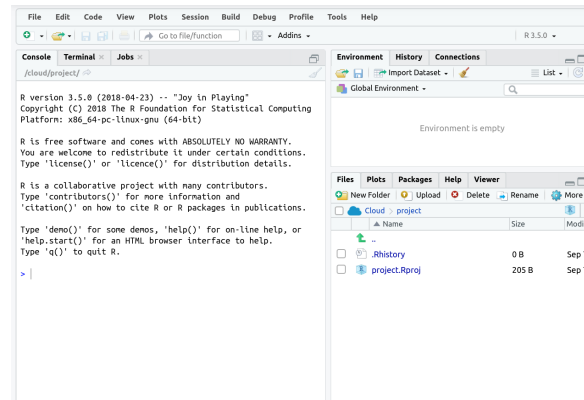
IDEs for R

- It is easiest to use R via an Integrated Development Environment (IDE).
- An IDE provides a ‘Front End’ to R which can make it a little bit easier to use.
- We will use Rstudio as an IDE, though there are many others available.

Installing R and Rstudio

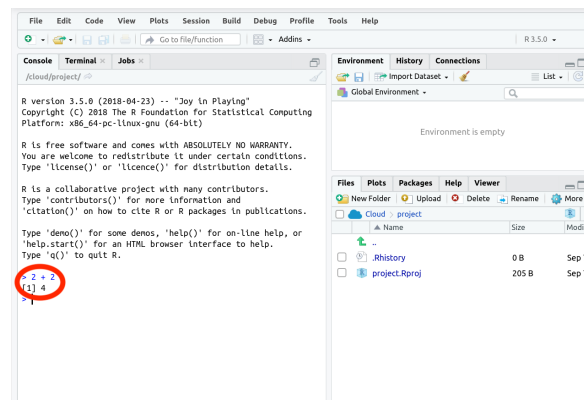
- We need to install R first and then Rstudio.
- R can be installed from: <http://cran.r-project.org>
- Rstudio can be installed from: <http://www.rstudio.com>
- See the guide “How to install R” available on Blackboard for installation instructions.

Rstudio layout

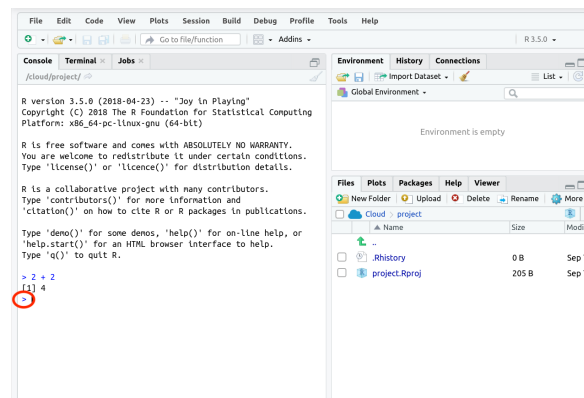


Using R

- The simplest way to use R is to type commands into the console window and press *Enter*:



- When the symbol `>` appears at the start of a line is called *prompt*. It means that R is ready to receive a command:



The simplest way to use R is to type commands into the console window:

```
2 + 2
```

```
## [1] 4
```

The [1] here indicates that this line contains the first element of the output.

Suppose we run a command which gives more output:

```
runif(7)
```

```
## [1] 0.8872186 0.5250860 0.3787199 0.5896092 0.1312525
```

```
## [6] 0.7955006 0.5930804
```

Here the second line starts at the 6th value.

A first R session

Let's try a simple command:

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

When you type this in to the console nothing happens, as the answer is stored in the object `x`.

The `c` stands for concatenate so `x` stores the values 1, 2 and 4.

Sure enough:

```
x
```

```
## [1] 1 2 4
```

Equal or not equal?

- R allows for two different versions to assign a value to a name:
 - Both `<-` and `=` are valid and produce the same answer:

```
z <- 2 + 2
```

```
z = 2 + 2
```

Selecting parts of objects

Parts of individual objects can be accessed via square brackets:

```
x[3]
```

```
## [1] 4
```

We can access multiple parts of objects with a colon:

```
x[2:3]
```

```
## [1] 2 4
```

Here we are accessing the second and third elements of the object.

Functions of objects

Suppose now we want the sum of `x`

```
sum(x)
```

```
## [1] 7
```

the `sum` function is applied to the object `x`

We can similarly store the output of a function in another object:

```
y <- sum(x)
y
```

```
## [1] 7
```

now the object `y` holds the sum of `x`

Functions of function

R allows you to run more than one function in a single line of code, for example:

```
z <- log(sum(x))
z
```

```
## [1] 1.94591
```

- This will first find the sum of `x`, and then take the natural logarithm of that value.
- R provides the ability to write our own functions.

Accessing data

R has a large number of built in data sets. You can access these by typing:

```
data()
```

Try out the following functions and see if you can work out what each is doing:

```
Nile
help(Nile)
mean(Nile)
plot(Nile)
hist(Nile)
```

- Notice that in R names are case-sensitive: `Nile`, `nile` and `NILE` do not refer to the same variable.

How to get help

R has lots of functions which do many different things to data.

If you know the name of the function in which you are interested, you can type, e.g.

```
help(sum)
```

This will provide you with the help file for that function.

If you don't know the function name you can type:

```
help.search('standard deviation')
```

This will provide you with a list of functions which fall under this topic.

More on getting help

You can shortcut `help` and `help.search` with `?` and `??` respectively:

```
?mean  
??"standard deviation"
```

Another useful function is `example`, which gives a list of examples for a specified function

```
example(mean)
```

Finally, `demo` can be very useful in giving an example of the range of things R can do:

```
demo(graphics)  
demo()
```

This will provide you with a list of available topics

Even more on getting help



Introduction to functions

Here is an example function which counts the number of odd integers:

```
oddcount <- function(x) {  
  # Set k to be 0  
  k <- 0  
  for(n in x) {  
    # %% finds remainder on division  
    if(n %% 2 == 1) k <- k + 1  
  }  
  return(k)  
}
```

What answers do you get when you run the following commands?

```
oddcount(c(1,3,5))  
oddcount(c(1,2,3,7,9))
```

The oddcount function

Let's take a closer look at the `for` command inside the function (you don't need to run it again)

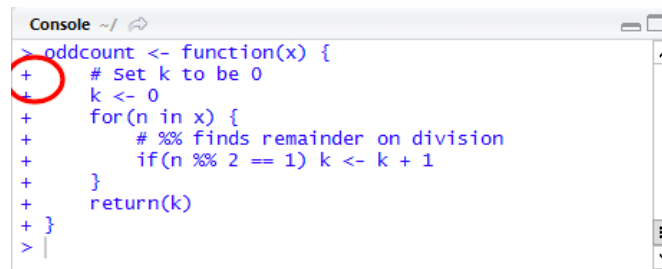
```
for(n in x) {  
  if(n %% 2 == 1) k <- k + 1  
}
```

- This is a `for` loop. It sets `n` to be the first value in `x` and runs the commands inside the `{ }` brackets, then repeats for the next value in `x`.

- The `%%` command finds the remainder on division by a number. Any integer that has remainder 1 after dividing by 2 must be odd.
- Finally `k` is increased every time we find an odd number. We report its value using the `return` function at the end.

Continuation prompt

- When the symbol `+` appears at the start of a line is called *continuation prompt*.
 - It appears when the expression is written in multiple lines.
 - If it appears inadvertently it is possible to stop it either completing the command, or pressing ESC (Windows and Mac) or Ctrl-C (Unix).

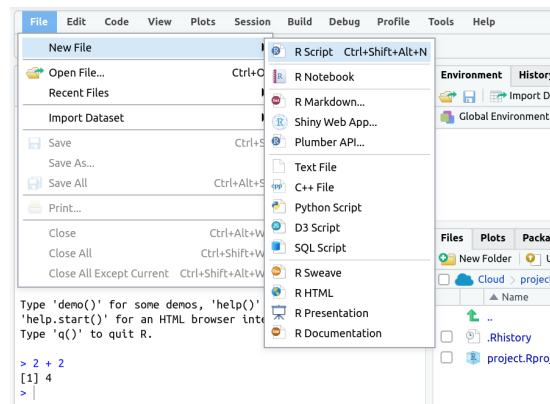


```
> oddcount <- function(x) {
+   # Set k to be 0
+   k <- 0
+   for(n in x) {
+     # %% finds remainder on division
+     if(n %% 2 == 1) k <- k + 1
+   }
+   return(k)
+ }
>
```

The Rstudio source window



- Rather than typing commands individually in the console window, it is often more useful to keep a record of everything you have run.
- You can store commands in a script in the source window and run these en bloc or line by line.



- From now on, always store your commands in the source window and save the script regularly.

Lessons from this week

- We can type commands into R via the console window and store them up in the source window.
- Functions provide a means of manipulating data.
- The `help` and `help.search` functions allow us to find what we need.
- We can write our own functions as required.