SESSION 1

In this session you will learn

a) how to use R to assign values to objects you created and to use these objects to make calculations
b) how to determine which objects exist in your R session and how to delete them,
c) why we use 'scripts' to document and keep track of our work,
d) how to find help when using R, and how the arrow keys are helpful

1. How do I access/ start R?

We will be working in RStudio, which is a user-friendly interface for interacting with R.

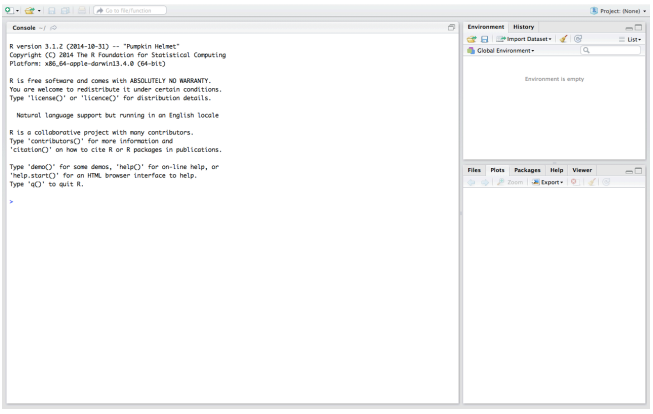
If you installed RStudio on your own computer, click the RStudio icon.

All University of Edinburgh computers have access to RStudio via the web; you can also access it from anywhere with an internet connection. In order to access RStudio via the web, go to the following link, and enter own UUN and your EASE password:

https://rstudio.mvm.ed.ac.uk/

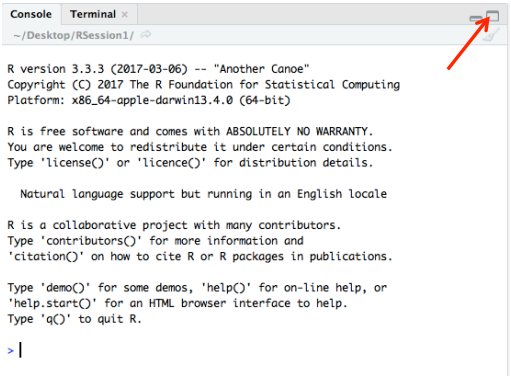
RStudio has a few panels. For now, you will primarily be interacting with the "Console" and a "Source" panels, on the left of the window (we'll make the Source panel visible in a moment). Take a moment to read what is presented at the top of the Console window. You will see information regarding warranty, the version of the R installed, and information on demos and how to find help (see more below regarding help).

You will see this:

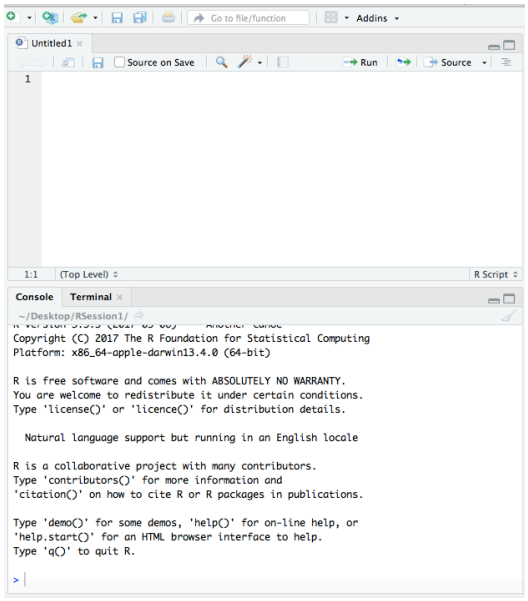


1a. <u>Setting up your work environment</u> – on your own computer, or using RStudio installed on computers in the Greenfield Computing Suite or the Main Library.

We recommend that, for any project you're working on, you save your work for that project in its own project directory. For example, if you were using RStudio to analyse data from a practical, you might save all of your RStudio work for that practical in its own directory (or folder); this helps to keep your work organized. To set up a project directory, we'll start by changing the appearance of the console slightly: click on the 'folders' icon at the top right of left-hand panel, as shown here:



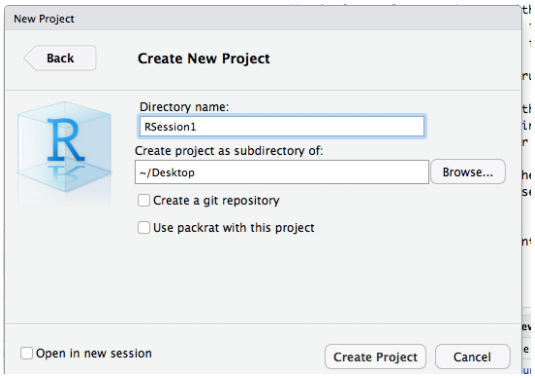
Clicking on this icon will cause a new panel to appear, like this:



Alternatively, RStudio might have already been configured on your computer to display these two panels.

1b. <u>Saving a "Project"</u>: on your own computer, or using RStudio installed on computers in the Greenfield Computing Suite or the Main Library.

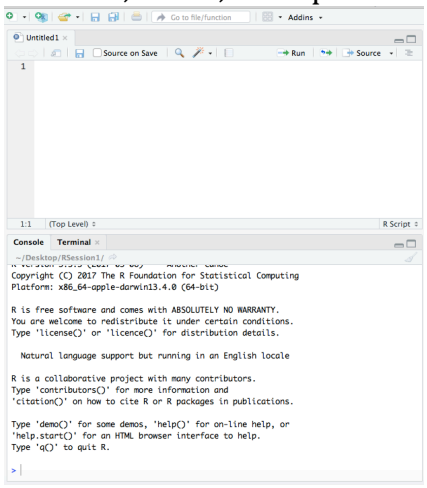
We will now simultaneously create and save a new directory that will hold your work for your current 'project'. Select "File", and then select "New Project…". You will be given a series of choices; I typically choose to create a "New Directory" and then a "New Project". You may then browse your computer (use the "Browse button") to select a place to save your new project directory. Provide your project directory with an informative name; in the example pictured below, I chose to save my project with the name 'RSession1', saved to my computer desktop. Note that if you're using RStudio installed on a University computer, you'll likely need to save your project on your M: drive.



If you are using the **online version of RStudio**, you may save a project Directory in a similar manner as described, above. However, the directory will be saved in an online space (i.e., not saved on your own computer, and also not on your M; drive, I believe).

1c. <u>Saving your script</u>.

As noted, above, we expect that the left-hand panels will look like this:

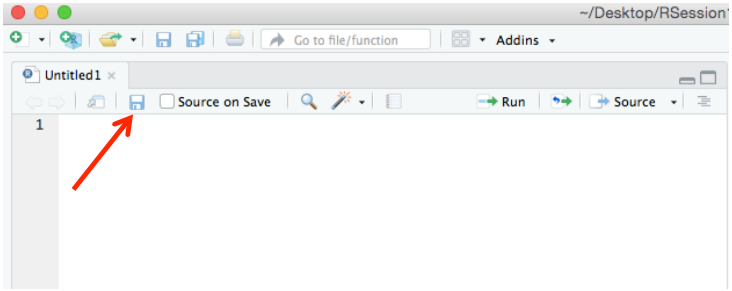


…if these panels appear somewhere else on your screen, that's perfectly fine, too. The important point here is that these two panels provide two alternative ways to submit commands to R. In the picture, above, the bottom panel displays the command prompt:

>

…this is one of the places where can enter your commands.

You can also write commands in the **"source"** panel, which is the top panel in the picture, above. We recommend that you write your commands in this "source" panel, because doing so allows you to save your work in a file called a '**script**'. We highly recommend that you save your work in scripts. Doing so allows you to easily and exactly reproduce analyses you completed in a previous work session. It also increases the 'reproducibility' of your work, and allows you to share scripts with your friends and colleagues. **To save you script**, click on the "floppy drive" icon, as shown below, and choose a sensible name for your script (e.g. RSessions1Script):

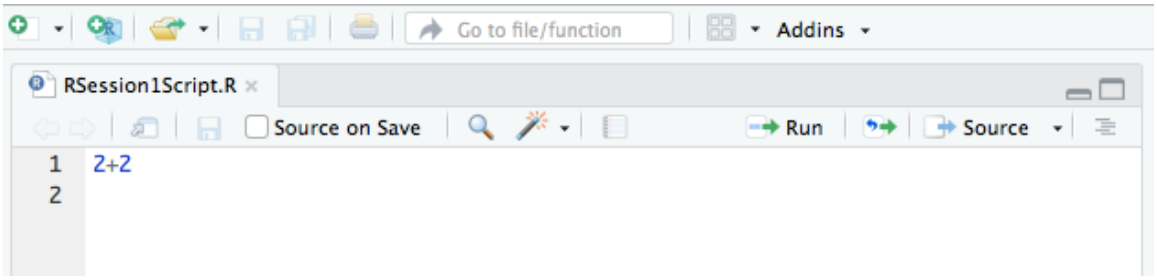


1d. <u>How to submit commands.</u>

As noted above, you can submit commands either from you're the source panel (i.e., from your script) or using the command prompt. Let's try two both approaches now, by adding 2 and 2.

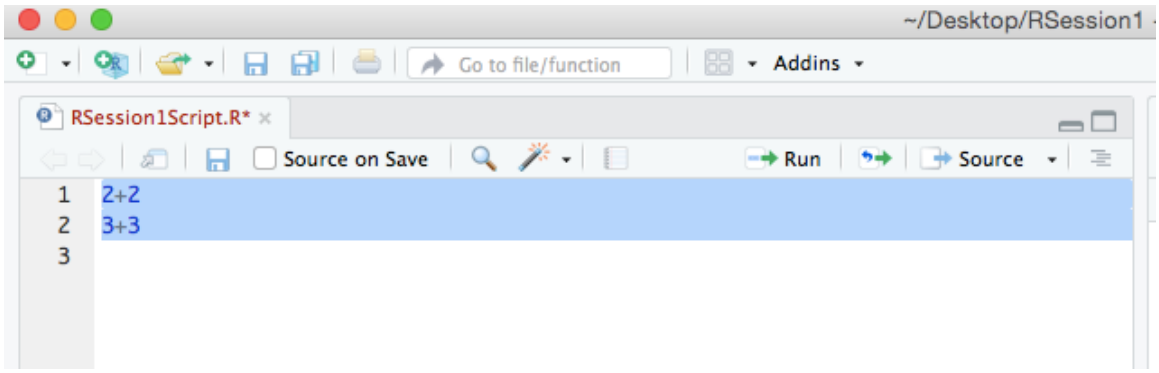
At the command prompt, type 2+2, and then 'enter'. You should see the correct output.

Now, type 2+2 in your source code; leaving the cursor (i.e., the flashing line that shows you where your typing will appear) on the command you've typed, click on the "Run" button above your script, as shown below:



You should now see your command appear in the command prompt, with the answer also provided.

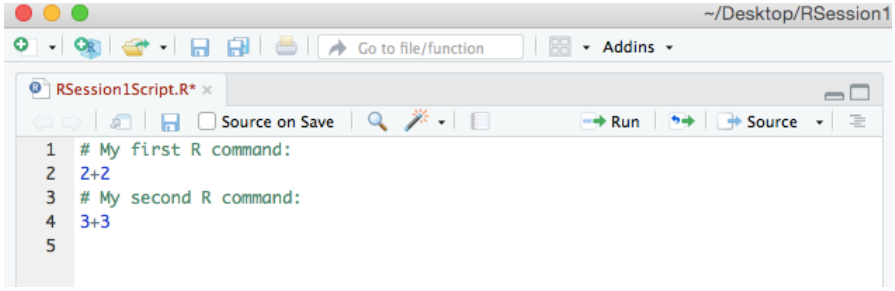
If you wish to submit more than one command simultaneously, you can highlight the commands that you wish to submit (as shown below) and then press the Run button:



As noted above, we encourage you to write your commands in a script form, and submit the commands with the 'run' button.

1e. Annotating your work

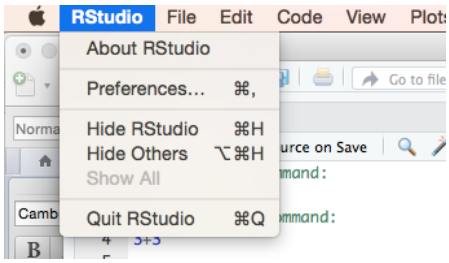
Sometimes you'll wish to use a script that you created a long time ago. When this happens, even your own work might confuse you if you cannot remember why you used the commands that you did. You can add 'annotation' to your scripts to help avoid this confusion. Specifically, you can use the # symbol to insert comments into your scripts, because R/Rstudio will ignore any writing on a line that appears after a #. We **highly recommend** that you add comments to your scripts that explain the steps in your work. Here is a simple example:



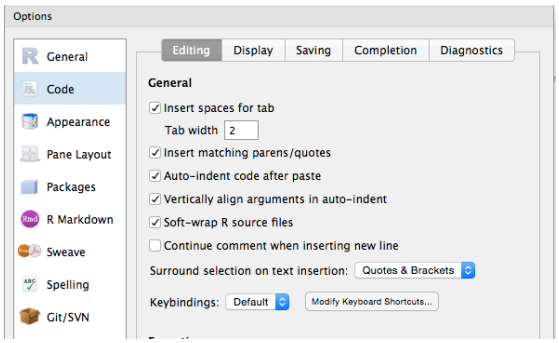
1f. Select "soft wrapping" for your scripts.

The source window (the panel where you create your scripts) has one annoying feature: if you have a long line of code, it will disappear off the right side of the screen and you'll need to scroll right to read it. To avoid this, you can adjust "preferences" or "options" or "global options", depending on your operating system. For example, on a Mac, you may alter your "preferences" by first selecting RStudio in

the tab at the top of the screen, and then selecting Preferences:



With Preferences open, select the "Code" tab on the left, and then select "Soft-Wrap R Source Files", as illustrated below:



NOTE that the exact method to select soft-wrapping will depend on your operating system – be brave and explore the options/preferences that R provides. Note as well that it is not essential to select soft-wrapping; it is merely a convenience. Unfortunately, the soft-wrapping option is **not available** for RStudio installed on the university computers.

2. R is an 'object oriented' language.

2.a What does it mean to be 'object oriented'?

This means that R works with 'objects' that you create. For instance, an object might be a dataset that you imported into R (which we will do on a regular basis), or might be the output of a statistical analysis. You can then do useful things to the objects.

Suppose we wanted to do a simple calculation: 2+3. Enter this calculation at the command prompt, and press Enter; you will see the answer on screen.

You can also store the result of this calculation in an 'object', which we'll call 'x' (you could call it almost anything you want, such as "Jane" or "Bob"). We do this with:

```
x <- 2+3
```

The arrow, '<-' causes R to store the result of 2+3 in the object x; think of the arrow as a means of assigning the output of 2 + 3 to x. (Note that there is no 'arrow' key on the keyboard; to make the arrow, type < followed by a dash (i.e., -)) We will use the

'<-' arrow very often.

Now, you'll notice that when you press Enter, you will not be given the answer to 2+3. To see this answer, you need to enter the name of the object that you wish to see, in this case, x.

Type x and then Enter at the command prompt, and you will see the answer:

[1] 5

So we know at this point that our object x has a current value of 5.

"[1]" in the readout in this case simply means item 1 in the object: in this case, our object contains only 1 item but R lists this anyway.

Useful R tip: *R is 'case sensitive', which means that R treats 'x' and 'X' as completely different objects. Likewise, R will treat the command 'ls()' (see below) differently from 'LS()'. So, be careful with upper-case and lower-case letters, and be consistent.*

This simple exercise demonstrates an important point: if you wish to know the content of an object, simply enter the name of that object.

2.b Managing objects:

You will create many objects while working with R - how do you know what objects you've created? We'll now introduce you to your first R function, **ls()**. ls() is a function that lists all the objects in your current workspace.

Enter ls() (i.e., type ls() and press Enter), and R will list the objects that you've made. In this case, you'll only see the object, "x".

Now create another object, y, and assign another value to it:

> y<- 9*3

And list your objects again:

> ls()

You should see the readout :

[1] "x" "y"

For convenience, RStudio also lists all the objects that you have created in the Environment window (upper right). Compare this to the results of ls().

To delete an object, use the function, **rm()**, which removes an object from a workspace. Alternatively, **remove()** works similarly.

To delete the object x, enter the command: rm(x)

Now, list the objects in your workspace again, using ls(). x should no longer be listed; if you see output reading '**character(0)**', this indicates that there are currently no objects in your workspace.

***Useful R tip**: If you wish to remove more than one object, separate them by commas. For example, rm(x,y) will delete objects previously created with names " x" and "y".*

So, to summarise:

x <- 2+3

…assigns to an object named 'x' a value defined by the operation to the right of "<-"

To reveal the current value of an object, simply type its name (x in this case)

ls() lists the names of all your objects

rm(x) removes your object (and its value) previously named 'x'

3. How do I find help?

We all need help with R from time to time. Fortunately, there are many resources for help with R.

3.a Help within R:

R, itself, has facilities to help.

If you know the name of a function that you wish to learn about (e.g., rm()), you can ask R for help in two ways:

help(rm)
?rm

Both commands will cause R to provide documentation about rm() in a window.

On the other hand, you may not know the name of a function, but you may know a

topic you wish to learn about. In this case, you could use the **help.search()** function to learn about functions that deal with this topic. For example, to learn about a chi square test (which you'll learn about later), type, **help.search("chi square")**. Try it now; the quotation marks are important, and must be included. The help information appears in the lower right panel of the RStudio window.

Useful R tip: *If you type a help.search command and you get an "unexpected symbol" error message this may be because you neglected to include quotation marks between the help.search brackets*.

3.b Help outside of R

rseek.org is a wonderful website that allows you to search for help on R, in a manner similar to using Google.

Try visiting **rseek.org**, and perform a search. For example, you could search for how to perform a chi square test; or how to take the square-root of a number, etc.

And, of course, you always have help available to you within this course. First, you can always approach staff for help – that's why we're here (see the first page of R Sessions for an e-mail address, and more)! But also remember to use your classmates; one of the advantages of coming to class and meeting your classmates is that you can, and should, help each other. That is part of the university experience.

3.c Helpful arrow keys:

Using scripts can involve a lot of typing. Fortunately, R has a short cut.

Let's say you wish to edit a command you entered previously, perhaps because you made a typo (this will happen a lot!). Instead of re-typing the command from scratch, you can use the 'up' arrow to visit previously used commands, which you can then edit. Similarly, the 'down arrow' lists more recently used commands.

Try scrolling through previously used commands using the up and down arrows. These arrows will become your best friends when using R.

RStudio also has a convenient "autocomplete" mode. When you start typing, it suggests variables or other operations for you to choose from. When these options appear, you can use the arrows to make a selection from the suggested options.

Be careful when using the arrows to access previously submitted commands. If you do not look closely at the command, you may accidentally use the wrong one.

4. Turning R off

At the end of your session, you should quit RStudio by typing the command,

```
q()
```

When you do so, you will be asked whether you wish to save your work. Type "n" (without the "").

5. Exercise:

We recommend that you complete two of the following questions; the additional questions provide more opportunity for practice.

5.1. In this session, you used R as a calculator. For example, adding 2 + 3. You also learned to create objects that contained a value (e.g., the object 'x' has the value 3 with: x <- 3). For practice, perform each of the following calculations both with numbers directly (e.g., 2 + 3), and by assigning a number to the word version representing the number (i.e., two <- 2):

a) 20 + 30
b) (5 + 6) / 7
c) (1 – 5) * 2

5.2. After completing question 1, determine which objects you have created (i.e., ls()). Now, remove the objects that represent even numbers.

5.3. Use the advice on getting help to find the function that will take the square-root of a number (eg using help.search, or the rseek.org website). Use this function to calculate the square-root of 16, and then of 7.

5.4. R knows some constants to a high degree of precision. For example, R knows the value of pi (i.e., 3.14…) to many digits. So users can ask R to perform calculations using these more precise values of a constant, like pi.

a) Use the advice on getting help to determine how to tell R to use the value, 'pi'.

b) Use this value to calculate the circumference of a circle, with diameter of 4cm (recall that the diameter of a circle equals pi * diameter).