# Rientation

This lab guide will give a first look at using R statistical software. It includes:

- hints on setting up a new project in R
- the use of scripts in R
- simple arithmetic
- using vectors, including data types, named vectors, and indexing
- functions built in to R, including sum(), mean() and sd()
- attaching add-on packages
- tidying up and saving your work, and closing R at the end of session.

You should have already downloaded R and installed it on your computer before starting this session; if not:

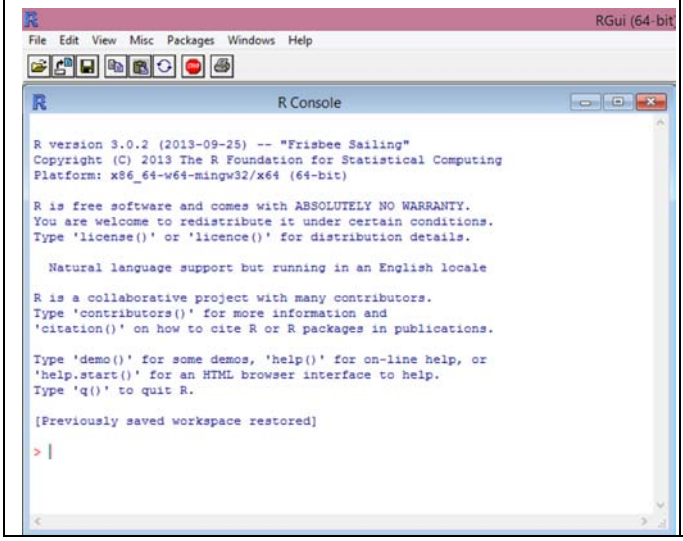
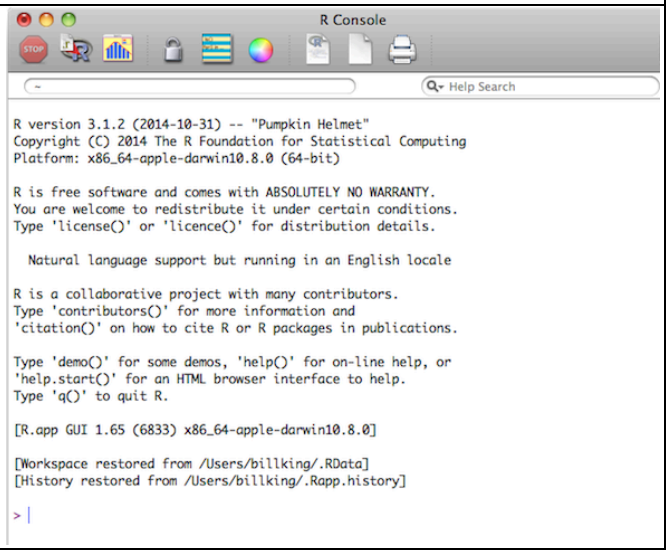
| Windows | Mac |
|---|---|
| You must have Windows 7 or later; many things will not work on older versions.<br><br>Go to https://cloud.r-project.org/bin/windows/base/ to download the installer. | You must have El Capitan or later; current versions of R will not work on earlier versions.<br><br>Go to https://cloud.r-project.org/bin/macosx/ to download the installer. |

## *Managing separate projects*

You will probably want to use R with a range of projects. A good way of keeping track of the different projects is to create a separate folder for each. We will use the folder for Day 1 of the workshop, which will have a name such as "Day01_Prelim&sampling&R".

| Windows | Mac |
|---|---|
| Go to the folder for Day 1 and you will see a big blue R icon with the name ".Rdata"[1]. Double-click on this to start R. The folder containing ".Rdata" is now the default. Put a copy of the ".Rdata" file in each of your project folders. Alternatively, you can start R from the "Start" menu or from a desktop icon and then set the default folder from the File > Change dir… menu entry. | Add R to the Dock. Drag the Day 1 folder from Finder onto the R icon in the Dock. This folder becomes the default folder for this R session, and the results will be saved there when you exit R.<br>Alternatively, you can start R then set the default folder from the Misc > Change Working Directory… menu entry |

[1] If the name isn't visible: in Windows 8 or 10, go to the 'View' tab in My Computer and make sure File name extensions and Hidden items are both checked; in Windows 7, go to Organize > Folders and search options > View, then select the Show hidden files… button and uncheck the Hide extensions to known file types box; click OK.

| Windows | Mac |
|---|---|
|  |  |



## *The R console*

When R opens, you will see the R Console (see screen shots above), with information about the version you are using and terms of distribution, and some hints about getting started.

The wedge (**>**) is the prompt, and the blinking vertical line (|) is the cursor.

## *Simple arithmetic*

R is a rather fancy calculator, and it will do all the simple calculator-type things too. You can start immediately typing into the R console:

Start with some simple arithmetic: type "2 + 3" and press enter:

```
> 2 + 3
[1] 5

> 
```

Try some other calculations: type these into the script and use Ctrl-r to run them in R:

2 * 3

2 / 3

3^2

The symbol ^ means "to the power", so 3^2 means "3 to the power 2" or "3 squared"

2 + 3 * 5

(2 + 3) * 5

R recognizes parentheses

See what happens if you don't complete the expression:

Type this line in R – it has no ")" at the end – and press Ctrl-r:

```
  2 * (3 + 5
```

In R you will see this:

```
> 2 * (3 + 5
+
```

Instead of the >, a + appears. R is waiting for you to finish the instructions. You can finish the command by typing ")" and pressing Ctrl-r, or press Esc to cancel.

It's okay to spread your commands over more than one line, but if you see the red + when you aren't expecting it, don't just carry on entering more commands!

That works fine, but you will have no record of what you did, and if you make a mistake you will have to start over. So most R users ("useRs") work with scripts.

## *Using scripts*

A 'script' in R is simply a text file containing the same commands that you would type into the R Console. The Day 1 folder contains a script we've prepared for this session.

| Windows | Mac |
|---|---|
| On the main menu in R, use File > Open script… and select the file "Day_1_Intro_R.R". | On the main menu in R, use File > Open document… and select the file "Day_1_Intro_R.R". |

The first few lines begin with "#". These are comments for human use and are ignored by R. Comments can also go after the code on a line; R ignores everything on the line after the # symbol.

Next come lines with R code: the first is **`2 + 3`**, We can pass this line to R and get it to do the calculation:

| Windows | Mac |
|---|---|
| Place the cursor anywhere in the line and press Ctrl-r; the line appears in red in the R Console and the command is run: the answer (5) appears in blue. (Instead of Ctrl-r you can use Edit > Run line or selection, or right-click and choose Run line or selection.) | Place the cursor anywhere in the line and press Command-return; the line appears in blue in the R Console and the command is run: the answer (5) appears in black. (Instead of Command-return you can use Edit > Execute) |

I use scripts a lot! I rarely type anything directly into the R Console; instead I have a script file open, type the commands into the script, then use Ctrl-r / Command-Return to run them. I save the script, so that I have an exact record of the calculations I did, and I can go back to it, review it, and change it if necessary. This is essential if you are doing an analysis of real data for a report or thesis.

Save your script regularly when you are modifying it: make sure the script window is active and press Ctrl-s (Windows) or Command-s (Mac) or use File > Save from the main menu.

## *Creating objects*

A good calculator has a "memory" button to store numbers, some even have 2 or 3.

To store "3" in "M1", type "M1 **<-** 3". "**<-**" (a less-than sign followed by a hyphen or minus sign) is called the *assignment* operator, but it's easier to read it as 'gets': "M1 gets 3". On the next line type "M1", select both lines and press Ctrl-r:

```
> M1 <- 3
> M1
[1] 3
```

R has an unlimited number of memory buttons! It stores numbers in named *objects* and you can have as many as you like, and give them sensible names. Object names must start with a letter and can include letters, numbers and dots. R is case sensitive: "m1" and "M1" are different objects.

R treats objects containing numbers just like numbers themselves: try typing "M1 + 5", "M1 * 5", etc

You can also assign the result of a calculation to an object, like this:

```
total <- 3 + 7 + 9
total
[1] 19
```

To see a list of the objects you have created, select "Misc > List objects" from the pull-down menus.

The list which appears will include the objects you just created – M1, total.

To remove an object, use 'rm':

```
rm(total)
total
Error: object "total" not found
```

'rm' is an example of a function in R; we'll look at functions in detail later.

## *Creating vectors*

A neat feature of R is that an object can store more than one number. The ":" operator produces a sequence of numbers:

```
1:10
-3:9
```

We can store the sequence in an object, perhaps to use later as points along the x axis:

```
xx <- 0:100
```

You can also use the *concatenation* function, c(), to assign a sequence to an object:

Try this:

```
prm <- c(2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, 41)
prm
[1] 2 3 5 7 11 13 17 23 29 31 37 41
```

An object which contains a row of numbers like this is called a *vector*. You can now do arithmetic with all the numbers in a vector at the same time:

```
prm/3
[1] 0.6666667 1.0000000 1.6666667 2.3333333 3.6666667 4.3333333
[7] 5.6666667 7.6666667 9.6666667 10.3333333 12.3333333 13.6666667
```

Notice that the second line of output begins with the 7th number in the vector, as indicated by the [7] at the beginning of the line.

Suppose you had trapped and weighed four squirrels in the forest. You can put the data into an object called (for example) 'sqrl':

```
sqrl <- c(26, 35, 29, 30)
sqrl
[1]  26 35 29  30
```

Our beautiful old spring balance is calibrated in ounces; we want to convert the weights from ounces into grams (1 ounce = 28.6g):

```
sqrl * 28.6
[1] 743.6 1001.0  829.4  858.0
```

If you want to keep the new values in grams, you need to assign them to an object:

```
sqrl_gm <- sqrl * 28.6
```

R has many *functions* which use vectors: you can add up the numbers in sqrl with sum(sqrl). Also try max(sqrl) and min(sqrl). length(sqrl) tells you how many numbers (or *elements*)

the vector contains. And we can get the sample mean and the sample standard deviation (SD) for our squirrel data:

```
mean(sqrl_gm)
[1] 971.5
sd(sqrl_gm)
[1] 111.2550
```

You can get help on a particular function by typing ? and the function name at the prompt (this is one case where I do type directly into the Console instead of the script):

```
> ?max
```

If you don't know the name of the function you want, use two question marks:

```
> ??average
```

or go to 'Help > Search help…' from the pull-down menus.

You can get at individual elements of a vector by using an *index* placed in square brackets [] after the object name. So the third element in prm is prm[3]:

```
prm[3]
[1] 5
```

Try typing prm[-3] : this gives all the elements except the 3rd. Try prm[1:4] : it gives the first 4 elements.

## *Data types*

Data can be in the form of numbers, character strings or true/false observations.

```
vn <- c(3.4, 5.6, 7.8)
vn
[1] 3.4 5.6 7.8
vc <- c("Melvin", "Joshua", 'Nurul')
vc
[1] "Melvin"  "Joshua" "Nurul"
vl <- c(TRUE, FALSE, TRUE)
vl
[1]  TRUE FALSE  TRUE
```

Note that character strings need quotes (single or double).

Logical values are often the result of a Boolean calculation, using the operators **>**, **<**, **==**, or **!=**. Note that the operator to test for exact equality in R is a double equal sign.

```
prm == 3
 [1] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
prm <= 3
 [1]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

You can't mix data types in the same vector; R will change the type of data to something that makes sense, so for the following input

```
c(1, 2, "three", TRUE)
[1] "1"     "2"     "three" "TRUE"
```

only the character type makes sense.

You can give names to the elements of a vector. For example:

```
names(vn) <- c("Jan", "Feb", "Mar")
vn
Jan Feb Mar
3.4 5.6 7.8
```

This can be a useful way of formatting data: for example, the data set for the areas of large islands looks like this:

```
> islands
          Africa       Antarctica             Asia        Australia
           11506             5500            16988             2968
etc etc
```

Use **?islands** to see details of this data set.

## Packages

One of the main strengths of R is the ability to create add-on packages with functions for special purposes. There are over 15,000 packages available on the main repository, CRAN. To use a package you need to install it; that only needs to be done once. Then, in each R session where you need it, you will attach the package with `library`. Two packages we will need:

### 1. wiqid

```
library(wiqid)
Loading required package: HDInterval
This is wiqid 0.2.3. For overview type ?wiqid.
```

If you get an error: `there is no package called 'wiqid'`, you need to install it.

### 2. rjags

```
library(rjags)
Loading required package: coda
Linked to JAGS 4.2.0
Loaded modules: basemod,bugs.
```

If you get an error: `there is no package called 'rjags'`, you need to install it, but you can only do that if you have JAGS installed and working properly

## Getting help

As we've seen, you can get help on a specific function or data set in R with the question mark:

```
> ?mean
> ?islands
```

If you don't know the name of the function or data set, you can use double question marks:

```
> ??average
> ??oats
```

which opens a window with a list of occurrences in the R help files.

And of course you can consult the manuals and FAQ files via the Help menu. You might find Help > Manuals (in PDF) > An Introduction to R useful.

## Getting finished

I rarely save the objects in the "workspace" when I exit R; instead I save the script file with the commands I used to create the objects, so that I can quickly recreate them next time.

If you do want to save the workspace, it's a good idea to tidy up first.

Use 'Misc > List objects' from the pull-down menu bar to see a list of the objects in the workspace.

Use the remove function rm() to get rid of objects you don't want to save:

```
rm(M1, total, prm, conseq)
```

| Windows | Mac |
|---|---|
| The easy way to exit is to click on the ☒ in the top right corner of the RGui window (or you can select 'File > Exit' from the pull-down menus, or press Alt-F4). R will ask if you want to "Save the workspace image?" If you click on Yes, it will be saved in the .RData file and reloaded next time you start R by clicking on that file icon. | Exit by clicking on the red button top left or the 'switch' icon on the to right corner of the Console window (or you can select 'R > Quit R' from the pull-down menus, or press Command-Q). R will ask if you want to "Save workspace image?" If you click on Yes, it will be saved in the .RData file in the default folder and reloaded when you drag that folder to the R icon in the Dock. |

Most software packages allow you to choose the name for the file when you save it. **R doesn't**! It always saves the data in ".Rdata", and if a file called ".Rdata" already exists, it will be overwritten; **R will not ask** before overwriting. So you can't keep data from different projects in R files with different names: that's why I recommend that you use separate folders.