## Introduction to R

My name is Kevin O'Brien (email: Kevin.obrien@ul.ie)

Dublin R User Group (www.meetup.com/DublinR)

Contents

[Introduction to R 1](#_Toc334214803)

[What is *R?* 1](#_Toc334214804)

[Working with R 4](#_Toc334214805)

[Packages 8](#_Toc334214806)

[Using R's help commands 10](#_Toc334214807)

### What is *R?*

*R* is a statistical Environment for statistical computing and graphics, and is one of the major tools used in statistical research and in applications of statistics research.

*R* is a suite of software facilities for:

* reading and manipulating data (our focus)
* computation
* conducting statistical analyses
* displaying the results graphically (useful packages like ***ggplot2***)

*R* is an open-source (GPL) statistical environment modelled after *S* and *S-Plus*, maintained and distributed by the R core-development team: an international volunteer team of statisticians and computers scientists.

(We can find out more about them by using the command **contributors()** a little later on)

Statisticians and data scientists have implemented hundreds of specialised statistical procedures, known as ***Packages***, for a wide variety of applications as contributed packages, which are also freely-available and which integrate directly into R.

More about Packages later (much more!!)

The S language was developed in the late 1980s at AT&T labs. The *R* project was started by Robert Gentleman and Ross Ihaka of the Statistics Department of the University of Auckland in 1995. *R* has quickly gained a widespread audience across the world.

*R* is the product of an active movement among statisticians for a powerful, programmable, portable, and open computing environment, applicable to the most complex and sophisticated problems, as well as more “routine”analysis.

**Using *R***

*R* offers the additional advantage of being a free and open-source system under the GNU general public licence, available for windows, Unix and Mac OS platforms.

The R project web page ( http://www.r-project.org ) is the main site for information on R. At this site are directions for downloading the software, accompanying packages and other sources of documentation.

R is a flexible language that is *object oriented* and thus allows the manipulation of a complex data structures in a condensed and efficient manner.

In common with functional languages, assignments in R can be avoided, but they are useful for clarity and convenience.

In addition *R* runs faster when loops are avoided, which can often be achieved using matrix calculation instead however, thus results in obscure looking code.

Additionally there are alternatives to “loops” that improve the readability of code, from the point of view of the user.

R's graphical abilities are also remarkable (with packages like “***lattice***” and “***ggplot2***”), with possible interfacing with text processors such as Latex with the package “***SWeave****”*.

An online guide "*An Introduction to R*" is automatically installed and can be easily accessed at any time (by typing **help.start()**at the command prompt**)**.

We will go into detail on using the help systems for *R* later.

**Key points:**

* R is easily downloadable from [www.r-project.org](http://www.r-project.org)
* R packages are also accessible from the R website.
* A user manual can be accessed by typing **help.start()**
* Matrix calculations will be key programming approach.

### Working with R

*R* is most easily used in an interactive manner, typing code into the command line and *R* gives you an response.

Questions are asked and answered on the command line.

*R* can be started in the usual way by double-clicking on the *R* icon on the desktop.

The following text will appear at the top of the screen. Let’s try out some demonstrations – particularly the graphics ones.

|  |
| --- |
| **Type 'demo()' for some demos, 'help()' for on-line help, or**  **'help.start()' for an HTML browser interface to help.**  **Type 'q()' to quit R.** |

**The prompt**

The **>** is called the prompt, used to indicate where you are to type.

If a command is not complete at the end of a line, *R* will give the "**+**" prompt on second and subsequent lines and continue to read input until the command is syntactically complete.

Commands are separated either by a semi-colon (**;**), or by a newline.

Elementary commands can be grouped together into one compound expression by braces (**{** and **}**).

Comments can be put almost anywhere, starting with a hashmark (**#**); everything after "**#**" is a comment.

The R console opens with information and then a prompt mark  (“ **>** “), ready to accept commands.

**Commenting and Colour Schemes**

Comments can be put almost anywhere, starting with a hashmark (**#**); everything after "**#**" is a comment.

Conventionally inputted material, such as codes and comments are coloured red. Computer output is coloured blue.

**The Assignment operator**

The assignment operator is a "**=**". This is valid as of R version 1.4.0. Previously it was (and still can be) a "**<-**".

Both will be used, although, you should learn one and stick with it.

Many long term R users prefer the second approach.

You can also use "**->**" as an assignment operator, reversing the usual assignment positions.

**Defining Variables**

A convention is to use define a variable name with a capital letter (*R* is case sensitive).

This reduces the chance of overwriting in-build R functions, which are usually written in lowercase letters.

Commonly used variable names such as x,y and z (in lower case letters) are not "reserved".

|  |
| --- |
| **x = 2           # create variable x and assign the value 2**  **y <- 4          # create variable y and assign the value 4**  **5 -> z          # create variable z and assign the value 5**  **x  #print x to screen**  **y  #print y to screen**  **z  #print z to screen** |

Remark:

The value of each variable is prefaced with a " [1] ". This is because the value that is presented is a vector. The value in square brackets indicates the index of the nearest data value. A vector may contain only one data value. More on that later.

Data may be numeric values, characters or logical values.

* **Numeric** values are simply numbers.
* **Characters** are letters or short groups of letters enclosed in quotation marks.
* The **logical** values are TRUE or FALSE respectively (with no quotation marks)

All data in R is stored as “objects”, which have a range of "methods" available. The "class" of an object can be found using the class() function.

The function **str()** can also be used to identify the type of an existing variable.

|  |
| --- |
| **> a=1**  **> str(a)**  **num 1**  **>**  **> b="kevin"**  **> class(b)**  **[1] "character"**  **>**  **> c=TRUE**  **> str(c)**  **logi TRUE** |

**Basic Calculations**

We will briefly look at how **R** accomplished basic calculations.

|  |
| --- |
| **x=10 ; y =5 ; z=2**  **x\*y # multiplication**  **x/z # division**  **x^2 # powers**  **sqrt(x) # square root**  **exp(z) # exponentials**  **log(y) # logarithms**  **pi             # returns the value of pi to 6 decimal places** |

*R* can handle more advanced calculations too, such as complex numbers , trigonometric functions, factorials and binomial coefficients.

When working with complex numbers, the data type needs to specified as complex, rather than numeric.

Binomial coefficients (e.g. 6C2) are computed using the **choose()** command.

|  |
| --- |
| **J = -1**  **sqrt(J)**  **str(J)     # variable is defined as numeric, not complex.**  **K = -1 +0i**  **sqrt(K)**  **str(K)   # variable is** defined as **complex .**  **sin(3.5\*pi)       # correct answer is -1**  **cos(3.5\*pi)       # correct answer is zero**    **factorial(6) # Factorial**  **choose(6,2)       # From 6 how many ways of choosing 2 items.** |

## Installed Data sets

To assist in the demonstration of statistical methodologies, there are several dozen embedded data sets installed automatically with R.

To view the list of such data sets currently available, type in data() to the command line.

One data set we will be using frequently is the Iris data set.

(To view the column names, and the first six cases, we will use the command head(). To view the last six cases, use the command tail() ).

|  |
| --- |
| **data()**  **iris**  **head(iris)** |

While we will be covering very little material relevant to graphics, it is worth nothing how many colours are supported by R. To get a full list of currently supported colours, type in the command colours().

### Packages

*R* contains one or more libraries of packages. Packages contain various functions and data sets for numerous purposes, e.g. the “***ggplot2***” graphics package, the “***actuar***” financial package, the “***nlme****”* statistical methods package, etc.

*R* consists of a base package and many additional packages. Some packages are part of the basic installation.

Others must be downloaded from the ***Comprehensive R Archive Network*** (CRAN), on the *R* website.

(Remark: most, but not all packages, are on CRAN. Another repository is R Forge. There are now several thousand packages there.)

To access all of the functions and data sets in a particular package, it must first be loaded into the workspace.

For example, to load the *nlme* package:

|  |
| --- |
| **> library(nlme)**  **> #you can now use this package** |

You can remove a package you have loaded using the command **detach().**

One important thing to note is that if you terminate your session and start a new session with the saved workspace, you must load the packages again.

**Installing packages**

To install a package use the command **install.packages().** Notice that you use the plural "packages" even if it is just for one package.

|  |
| --- |
| **> install.packages("evir",”outliers”)**  **>**  **> # carry out installation instructions**  **>**  **> library(evir)**  **> library(outliers)** |

To check what packages are currently loaded into the workspace, use the command **search().**

|  |
| --- |
| **> search()**  **[1] ".GlobalEnv" "package:MASS" "package:stats"**  **[4] "package:graphics" "package:grDevices" "package:utils"**  **[7] "package:datasets" "package:methods" "Autoloads"**  **[10] "package:base"** |

**Learning about Packages**

Workbooks and help manuals for ***R*** packages are usually available on the CRAN website. These are an invaluable resource for using packages.

Some packages require a more recent version of ***R*** than the one installed. You will have to install the new version accordingly.

Packages can be updated using the **update.packages()** command.

Also packages are "dependents" on other packages. Installing one package may require the installation of other packages that it is able to run, and these will also be installed.

A list of dependencies usually accompanies a package’s website.

## 

### Using R's help commands

As mentioned earlier, *R* has an inbuilt help facility. To get more information on any specific named function, for example “boxplot”, the command is: “**?boxplot” or “help(boxplot)”**

|  |
| --- |
| **>?boxplot # access help on boxplots**  **>help(Im)        # access help on "Im"** |

A description of the package – as well as some demonstration code – are provided.

Try out the demonstration code.

On most R installations help is available in HTML format by running **help.start()** which will launch a Web browser that allows the help pages to be browsed with hyperlinks.

|  |
| --- |
| **>help.start()** |

There is also an “apropos” function. This command returns a list of the names of all objects matching the argument. This is very useful for finding out what commands and functions are available.

Suppose we want to find out a way of performing a correlation analysis. Use “cor” as the argument (short, but not too short).

|  |
| --- |
| **> apropos("cor")**  **[1] ".\_\_C\_\_recordedplot" "cancor" "cor"**  **[4] "cor.test" "cov2cor" "Harman23.cor"**  **[7] "Harman74.cor" "recordGraphics" "recordPlot"** |