Getting Started with R

Dr Nick Fieller

Department of Probability & Statistics

University of Sheffield

2011/12

Contents

1 Introduction 2
1.0 Preliminaries and reference manuals 2
1.1 Objectives 3
2 An Outline Guide to R 3
2.1 What is R? 3
2.2 Installing R 4
2.2.1 Updating R 5
2.3 R is an interactive program 6
2.3.1 Commands from the keyboard in the R console 6
2.3.2 Use of the toolbar icons 7
2.3.3 Use of the menus 7
2.3.4 Use of script files 8
2.4 Customising the appearance of R 8
2.5 R is a function language 8
2.6 R is an object orientated language 9
2.7 Saving workspaces 10
2.7.1 Mistakenly saved workspace 11
2.8 R is a *case-sensitive* language 11
2.9 Packages or Libraries 12
2.10 Obtaining help in R 13
3 Inputting and outputting data to and from R 15
3.1 Reading data from the keyboard 15
3.2 Reading data from files 16
3.3 Writing data to files 17
4. Working Directories 18
4.1 What is a working directory? 18
4.2 Checking working directory 19
4.3 Changing working directory 19

1 Introduction

1.0 Preliminaries and reference manuals

R is an open source system and is available free. It is 'not unlike' the expensive commercial package S-PLUS; the prime difference is that **R** is command-line driven without the standard menus and dialog boxes for statistical operations in S-PLUS. Otherwise, most code written for the two systems is interchangeable. There are however a few differences, for example in the way external files are referenced (S-PLUS uses a single / and **R** uses a double // in full file pathnames). There may also be differences in the available optional arguments to statistical commands and functions. These are quickly verified by use of the Help system.

The sites from which **R** and associated software (extensions and libraries) and manuals can be found are listed at

http://www.ci.tuwien.ac.at/R/mirrors.html

The nearest one to Sheffield is at

http://cran.uk.r-project.org (in Bristol, UK)

If you are located elsewhere then use your local [geographical] knowledge. Free versions of full manuals for **R** (mostly in PDF format) can be found at any of these mirror sites. There is also a wealth of contributed documentation.

1.1 Objectives

The aim of these notes is to provide a guide to installing and running **R**. In many places (especially in referring to filenames and pathnames etc) the notes are specific to a Windows operating system. Experienced Mac users should be able 'translate' where necessary to the Mac equivalents. In particular the notes indicate how to obtain help of various kinds, simple ways of reading in data, finding out what is in your workspace and troubleshooting problems of unexpected objects in your workspace. For absolute beginners it is intended that the notes should be used in conjunction with an Echo recording of a session working through the examples given here.

2 An Outline Guide to R

2.1 What is R?

R is a powerful interactive computer package that is orientated towards statistical applications. It will run on the most commonly used platforms (or operating systems) Windows, Linux and Mac. The notes here are orientated towards use on a Windows platform. The are several differences on a Mac platform, notably in the menus and icons. It consists of a *base system* that can be downloaded without charge together with many contributed packages for specialist analyses. It offers:–

- an extensive and coherent set tools for statistics and data analysis
- a language for expressing statistical models and tools for using linear and non-linear statistical models
- comprehensive facilities for performing matrix calculations and manipulations, enabling concise efficient analyses of many applications in multivariate data analysis and linear models

- graphical facilities for interactive data analysis and display
- an object-orientated programming language that can easily be extended
- an expanding set of publicly available libraries or packages of routines for special analyses
- libraries or packages available from the official Contributed Packages webpages are thoroughly tested by the **R** Core Development Team
- packages have manuals, help systems and usually include illustrative datasets and examples.

2.2 Installing R

Full instructions for installing **R** are given on the **R** Project home page at http://www.r-project.org/. The easy way to go to this webpage is to google **R.** The first step is to choose a site geographically close to you from which to download the package. Under the section Getting Started in the middle of the first screen click on download R or on CRAN Mirror (they link to the same site) and make your choice. Next choose the appropriate operating system (Windows, Mac or Linux), select the option base and download the system. Accepting the option to run the download file will install the package after downloading it. Accepting the default options for locations etc is simplest but these can be customized. By default an icon (ℝ [with version number]) is placed on the desktop. Clicking on the icon will open an **R** session (i.e. start the **R** program). The **R** Graphical user interface (RGui) opens and you are presented with a screen like this:

```
R version 2.14.2 (2012-01-17)
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

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.

>
```

2.2.1 Updating R

New versions of **R** are released quite frequently (every few weeks). Usually these involve just minor changes and this is indicated in the version number. Relatively minor changes are indicated by increasing the third part of the number; more substantial by increasing the second. It is usually not worth updating the version on your machine on every occasion; every year or two is generally sufficient. To check whether there are updates available then check the CRAN web pages (or use the menu on a Mac platform only).

New versions of **R** can be installed without uninstalling the older version, both remain on the machine. If an old version is removed the specialist libraries (see below) you have installed will remain there. They can be updated *in situ* via the menus.

2.3 R is an interactive program

R is *interactive* in the sense that the program will operate as soon as it has an instruction to do so. Instructions can be given to **R** in several ways:-

- by typing direct from the keyboard into the main window (the ***R*** *console* window)
- by clicking one of the icons in the toolbar at the top of the screen
- by clicking one of the menu (& submenu) items.
- using a *script window* to perform a sequence of several commands in order.

The second and third methods provide short cuts for common operational tasks that can also be performed with commands typed directly. None of the icons and menus is concerned with statistical operations; only tasks such as installing and loading packages, changing the appearance of the screen, opening and saving files etc. The final method is an alternative to typing several commands directly into the **R** console window. The advantage is that they can be typed in advance and saved and used on other occasions (perhaps with some editing or alterations, thus avoiding repeatedly typing the same things into the command line). All of the tasks covered by the shortcuts in the icons and menus can also be performed with commands typed into the **R** console window.

2.3.1 Commands from the keyboard in the R console

The symbol **>** is the *command line prompt symbol;* typing a command or instruction will cause it to be executed or performed immediately. For example:

```
> x<-c(2.1,3.4,-1.1,-5.2,4.7)
> x
[1]  2.1  3.4 -1.1 -5.2  4.7
> max(x)
[1] 4.7
> min(x)
[1] -5.2
```

will read the string of numbers (`2.1,3.4,-1.1,-5.2,4.7`)′ into a vector `x`, (note the 'assign' symbol `<-`). Entering just `x` will print the vector itself; `max(x)` and `min(x)` will return the maximum and minimum values in `x.`

All statistical procedures (such as `t.test(.)` or `lm(.)` for performing t-tests and fitting linear models respectively) within **R** must be typed direct into the **R** console or entered through a script file or the `source(`*`filename`*`)` command (which runs all the commands in a separate file, similar to issuing commands from a script file within an **R** session). Further, all the functions of the icons and menus can be handled by commands and in some cases it might be preferable to do so since there may be more options (and therefore flexibility) in the commands.

2.3.2 Use of the toolbar icons

Holding the cursor over an icon will cause a description of its function to appear. Most of these are self-explanatory and are standard. The set of available icons will change according to what type of window is currently active. In the main **R** console window (on a Windows platform) the first two allow you to open *script files* and *saved workspaces* (see later sections). Others allow saving of workspaces, copy and paste. The final icon (a printer) will print the active window. In other windows (e.g. a script window or a graphical window) different icons and menus will appear.

2.3.3 Use of the menus

Along the top of the window is a limited set of menus. These provide short-cuts to tasks such as opening files, changing working directories, changing the appearance of **R**, installing and loading packages etc. These are discussed individually in later sections below. The menus on a Mac platform are rather different. None of the menus (on Windows or Mac platforms) allows statistical operations.

2.3.4 Use of script files

A convenient way of running several commands in sequence is to open a *script window* using the `File>New script` menu which opens a simple text editing window for typing the commands. Highlighting a selection and then clicking on an icon in the tool bar shewing an arrow pointing from one file to another will run the commands in the selection. Lines in the script window can be edited and run again. A script can be saved in a file (with default extension .R) and opened in a later session via the menus or the open file icon.

2.4 Customising the appearance of R

It is possible to alter the appearance of **R** on the screen. From the menu select Edit > GUI preferences ... and you can select the colours used for the background, text in various contexts and also the font used. Perhaps the most useful is the size of characters which by default is 10pt. Increasing this to 12 or 14 may be useful, especially if displaying on a data projector.

2.5 R is a function language

All commands in **R** are regarded as *functions*, they operate on *arguments*, e.g. `plot(x, y)` plots the vector `x` against the vector `y` — that is it produces a scatter plot of `x` *vs.* `y`. Even Help is regarded as a function:— to obtain help on the function `matrix` use `help(matrix)`. To end a session in **R** use `quit()`, or `q()`, i.e. the function `quit` or `q` with a null argument. In fact the function `quit` can take optional arguments, type `help(quit)` to find out what the possibilities are. Another function which takes a null argument is `search()`but this one does not have any optional arguments.

2.6 R is an object orientated language

All entities (or 'things') in **R** are ***objects***. This includes vectors, matrices, data arrays, graphs, functions, and **the results of an analysis.** For example, the set of results from performing a two-sample t-test is regarded as a complete single object. The object can be displayed by typing its name or it can be summarized by the function `summary()`. Even the results of `help` are objects, e.g. of `help(matrix)`. If you want to store the object created by this for later examination (though the need for this may be rare), giving it say the name `matrixhelp` then do `matrixhelp<-help(matrix).` Typing `matrixhelp` will print the help information on the screen (or it can be exported to a file).

The most common use of saving an object is to retrieve specific quantities from a statistical analysis. For example typing `t.test(x,y)` will present the results of a two-sample t-test of equality of means of `x` and `y` on the screen:

```
> t.test(x,y)

        Welch Two Sample t-test

data:  x and y
t = -0.9864, df = 38.475, p-value = 0.3301
alternative hypothesis: true difference in means is not equal to
0
95 percent confidence interval:
 -1.026029  0.353536
sample estimates:
 mean of x  mean of y
-0.1992073  0.1370391
```

Storing the results of the t-test in an object, `xytest` say, allows items such as the p-value or confidence interval to be retrieved on demand:

```
> xytest<-t.test(x,y)
> xytest$p.value
[1] 0.330087
> xytest$conf.int
[1] -1.026029  0.353536
attr(,"conf.level")
[1] 0.95
```

To find out what items (or *values*) are stored in an object created by the command use the help system. Then they can be retrieved by entering `objectname$value.`

Especially useful in multivariate analysis is storing the scores and rotation matrix from a principal component analysis.

2.7 Saving workspaces

Objects such as a matrices and vectors (see below) created during an **R** session can be saved in an **R** workspace file through the `File>Save Workspace...` menu or via the save icon (like an old 3½″ disk). They can be loaded into the **R** session by the menu or (if the default Rdata file extension is accepted) the file can be located in Windows Explorer and clicking initiates an **R** session with this workspace open. Issuing the command `objects()` (or equivalently `ls()` or use the menu under **`Misc`**) will list the objects created (or retrieved from a workspace) during the current session.

When you close down **R** you are prompted to say whether or not you want to save the workspace image (the default is 'yes' which is often not advisable). If you have loaded a workspace during the session then this will be overwritten by the current one if you say yes. It is better to save a workspace with a memorable name in an Rdata file using the `save(.)` command. Then, when you next run **R** you will start with an empty worksheet and can load any previously saved one.

2.7.1 Mistakenly saved workspace

BEWARE:– If you have created all the objects in the workspace during the current session (i.e. you have not loaded or opened a previously saved workspace) when you accept the invitation "Save workspace image?" at the end of a session it will be saved somewhere on your drive. ***When you next start R this workspace will be automatically restored***. This can have unexpected consequences and can cause mysterious problems because you will have objects in the workspace that you might not have intended to be there. To cure it you can remove all objects by `rm(list = ls(all = T))` or use the menu under **`Misc`** and choose `Remove all objects`. This should give a response `character(0)` to the command `ls()`.

2.8 R is a *case-sensitive* language

Note that **R** treats lowercase and uppercase letters as different, for example inverting a matrix is performed using the function `solve()` but **R** does not recognize `Solve()`, nor `SOLVE()`, nor…… The objects `x` and `X` are distinct (and easy to confuse). The function `matrix` and the library `Matrix` are distinct.

2.9 Packages or Libraries

A package or library is a set of related functions or commands for performing various types of analyses. Most of these are for specialist types of statistical analysis which most people would use only occasionally, e.g. analysis of angular or circular data, analysis of failure time or survival data etc. A few packages contain a wide range of commonly used commands which most people will use on every occasion they use **R**. The commonly used ones are automatically downloaded and installed when **R** is installed. A few are loaded (i.e. all the commands in the library are made available) in each **R** session. These include `stats` and `graphics` (but not `survival` and `MASS` even though they are installed with the standard system). Other more specialist ones need to be downloaded from a CRAN site and installed on your machine before you first want to use them.

The `Packages` menu allows you to install specific packages `Packages>Install Package(s)...` (which needs to be done only once) and then to load them into the session. You will be asked to 'set the CRAN mirror' (i.e. choose the geographical location from which to download the package) if you have not done so already before you select the package(s) to download. This facility is itself available from the `Packages` menu. The CRAN mirror needs to be set only once during each **R** session and then only if you wish to install additional packages or updates. Once the package is installed on your computer you do not need to download it again; it will be available in all further **R** sessions. It is advisable to update your installed packages from time to time (especially if you install a new version of **R**) and this can be done from the `Packages` menu with `Packages>Update Packages...`.

Each time a new session is started you need to load the packages which you will need during that session (other than `stats`, `base` and `graphics` which are automatically loaded in each session). This can be done from the `Packages>Load Package...` menu or by the command `library(`*`packagename`*`).` Some of the commands needed for matrix manipulations are within the `MASS` library which is automatically installed (together with a few others such as `stats`, `survival`, `graphics`, `...`) when **R** is first installed, i.e. it does not need to be installed from the `Packages` menu but it does need to be loaded if needed during each **R** session. [MASS is *Modern Applied Statistics with S*, by W N Venables & B D Ripley, (2002) Springer]. Similarly, many of the functions needed for Survival Analysis are in the package `survival` which is installed with the base system of **R** but needs to be loaded at the start of each session if it is to be used. If the survival analysis planned includes fitting accelerated failure time models or analysis of competing risks then the libraries (or packages) `eha` and `cmprsk` need to be downloaded and installed.

To discover which packages are loaded during a session issue the command `search().`

2.10 Obtaining help in R

Obtaining help on specific functions during an **R** session can by done by using either `help(functionname)` or `?functionname`. This will give you the list of possible arguments and the list of possible values produced. There may also be examples of their use, including script files which can be cut&pasted into a script window for you to run. Typing `example(functionname)` may run one or more examples. This of course requires you to know the name of the function.

Typing `library(help=libraryname)` will give summary description of the library together with an index of all functions and datasets supplied with the library. Having found the name of a function or a dataset then use `help(.)`, `?` and `example(.)` to find out more about it. For example `library(help=stats)`lists all the functions in library `stats`; these are mostly statistical functions such as `t.test` and then typing `help(t.test)` shows exactly how to perform a Student's t-test and you will see that he same function with different options will handle both one-sample and two-sample tests, both one-sided and two-sided.

To find out what various packages can do look at the CRAN website and click on packages. This has a basic search facility with CTRL+F (for the Windows Find dialogue box) which will search for an exact match for the character string entered in the dialogue box. For example to find packages which have functions for imputation of missing values then go to the Packages page on the CRAN project page and scroll down to the list headed **`Available Bundles and Packages,`** press CTRL+F and enter `impute` in the dialogue box. This will list in turn `arrayImpute`, `impute`, `imputeMDR`, and `yaImpute`. This technique will only find strings which appear on the very brief summary descriptions of the packages. A more thorough search can be performed from within an **R** session with `help.search` or `??`

For example `help.search(”characterstring”)` or equivalently, `??characterstring`, will search all installed packages for an approximate match in the summary description of each function in the package to the `characterstring`. The default is to use fuzzy matching which may have unexpected consequences. For example using `help.search(”impute”)` or equivalently `??impute` will also find all such occurrences of `compute`. To find out how to avoid this and instead use exact matching try `help(help.search)`.

To find out more about **R** the **R** website has links (under `Documentation`) to various manuals and other contributed documentation. Following the link from the CRAN page under `Other` and **R-related projects** gives to **The R Wiki** **at**

http://wiki.r-project.org/rwiki/doku.php

3 Inputting and outputting data to and from R

3.1 Reading data from the keyboard

Small amounts of data can be typed directly from the keyboard. For example to create a vector x of length 4 containing the four numbers 1.37, 1.63, 1.73, 1.36 do

```
x<-c(1.37, 1.63, 1.73, 1.36)
```

Entering numbers directly into a matrix is rather more complicated but see the following examples:

```
> A<-matrix(c(1,2,3,4,5,6),nrow=2,ncol=3,byrow=F)
> B<-matrix(c(1,2,3,4,5,6),nrow=2,ncol=3,byrow=T)
> A                                  > B
     [,1] [,2] [,3]                       [,1] [,2] [,3]
[1,]    1    3    5                  [1,]    1    2    3
[2,]    2    4    6                  [2,]    4    5    6
> A[1,2]                             > B[2,2]
[1] 3                                [1] 5
> A[1,]                              > B[2,]
[1] 1 3 5                            [1] 4 5 6
> A[,2]                              > B[,3]
[1] 3 4                              [1] 3 6

> C<-matrix(c(1,2,3,4,5,6),2,3)
> D<-matrix(c(1,2,3,4,5,6),2,3,byrow=T)
> C                                  > D
     [,1] [,2] [,3]                       [,1] [,2] [,3]
[1,]    1    3    5                  [1,]    1    2    3
[2,]    2    4    6                  [2,]    4    5    6
```

Items to note are that if you want **R** to take the numbers across the rows then you need to specify this, the default is different. You can refer to individual elements of a matrix, e.g. **`A[1,2]`** and individual rows or columns, e.g. **`A[1,]`** and **`A[,2]`** and to a range of columns (or rows) using a colon e.g. **`X[1:5,]`** will print the first 5 rows of **`X`**, however many columns **X** may have. This is particularly useful to check the column names of **X** or variable names of a dataset to make sure they are what you expected.

The function `scan()` can be used to enter data and will stop when a complete blank line is read. For example:–

```
> x<-scan()
1: 1.37
2: 1.63 1.73
4: 1.36
5:
Read 4 items
> x
[1] 1.37 1.63 1.73 1.36
>
```

`scan()` is a very flexible function with facilities for entering tables of numbers etc, to find out more type `help(scan)`.

3.2 Reading data from files

The three main functions for reading tabular data from files are, `read.table()`, `read.csv()`and `read.delim()`. The first is used primarily for plan text files (i.e. with extension .txt or .dat), the second for comma separated values (e.g. as produced by Excel) and the third for tab separated values. The default format of the data in `read.table()`is that the first row should contain the column names (i.e. variable names) and the first item of each row is the row name, so the first row contains one fewer item than the other rows. If the data are not in such a standard form then look at the help system to find out how to use the additional arguments `header`, `row.names` and `col.names` to read the data correctly.

If a data file has been saved during an **R** session (using the `save()` function (see `help(save)`) then the data can just be retrieved by `load(“`*`filename`*`”)`.

The `source(“`*`filename`*`”)` function will execute all the **R** commands in the specified file and this can be a convenient method of delivering a data file.

The library `foreign` can be used for reading data files created by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ..., (but not S-PLUS). There are commercially available packages for reading S-PLUS and other files, most with a substantial discount for academic use.

3.3 Writing data to files

The main mechanisms for saving data and **R** objects are the functions `save()`, `write()` and `write.table()`. All of these take optional additional arguments (see the `help()` system). The general form of these is

```
> write(x, file=”datafile”)
```

where `x` is the set of data to be output, and `datafile` is the full filename (including pathway from current working directory) where the set is to be saved.

Similarly

```
> write.table(dataframe, file=”datafile”)
```

and

```
> save(Z,file=”filename”)
```

where `Z` is any **R** object (including **R** data sets and **R** script files). The difference between `write()` and `write.table()` is that the latter is used specifically for saving dataframes.

For example, to read a comma separated file `bulls.csv` in the current *working directory* (see below) into **R** and then save it as an **R** data set for later use under the name `bulls.Rdata` we have

```
> bulls<- read.csv(file=”bulls.csv”)
> save(bulls, file=”bulls.Rdata”)
```

and then on a later occasion the data set can be loaded into **R** by

```
> load(“bulls.Rdata”)
```

or by double clicking on the file `bulls.Rdata` in Windows explorer. The latter method has some advantages since it then also switches the current *working directory* to that containing the file `bulls.Rdata` (see next section).

4. Working Directories

4.1 What is a working directory?

On a windows platform if **R** is opened by double clicking on the ℝ icon on the desktop (if this has been selected during installation) or from the Start menu via All Programs then the current *working directory* is My Documents. This means that if a filename is given with no pathway then it is assumed that the file is located in the working directory. If the file is somewhere else then the full pathname has to be given. On a Mac platform, the initial working directory is the top level directory of the current user (i.e. that containing the Documents folder).

For example, the **R** datafile bulls.Rdata on this Windows machine is located in a subdirectory on the machine used to write these notes called

```
C:\Documents and Settings\Nick Fieller\My Documents\My Teaching\MVA05\Rfiles
```

so to load in the datafile to an **R** session we need

```
> load("C:\\Documents and Settings\\Nick Fieller\\My
Documents\\My Teaching\\MVA05\\Rfiles\\bulls.Rdata")
```

which gives the full pathname of the file or, since the current working directory is My Documents which is a direct ancestor of the file needed, just the pathname from this directory is sufficient,

```
> load("My Teaching\\MVA05\\Rfiles\\bulls.Rdata")
```

Note that the \ in pathnames as shewn by Windows Explorer must be replaced by \\ in **R**. This is a peculiarity of **R** and one of the few distinctions from S-PLUS.

4.2 Checking working directory

To find out what the current working directory is then use the function `getwd()`.

It is often sensible to use a dedicated subdirectory for a piece of analysis with all data files, script files, output, saved objects etc together in one place.

4.3 Changing working directory

To change the working directory to a different one then use `setwd()`, e.g.

```
> setwd(My Teaching\\Lisbon\\Medexercises”)
```

will change the working direct to `Medexercises` in the folder `Lisbon`.

Instead of using `setwd(.)`it is perhaps easier to use the File menu in **R** and then `change dir...` which allows you to navigate (up or down) to the desired folder starting from the current working directory, possibly My Documents.

The method I personally prefer on a Windows platform is to place an **R** datafile (`coughs.Rdata` say) into the folder `Medexercises` (using Windows Explorer) and then double clicking on the file `coughs.Rdata`

This will open **R** with the datafile already loaded and also change the working directory to that containing the datafile `coughs.Rdata` (i.e. `Medexercises`). Thus by default (i.e. unless full pathnames are quoted) **R** will send any output to that window and look for input there.

This facility is only available on a Windows platform. On a Mac **R** can be started by double clicking on a file with extension .Rdata or .R but it does not load in the datafile or workspace nor change the working directory which remains as the default.

4.4 Checking contents of working directory

To find out what files are in the current working directory use `dir()` or `list.files()`. These can also be used to find out the contents of any folder (see the `help()` system).