Notes on R

1 Resources

- The home page for **R** is at http://www.r-project.org/. This contains links to obtain the software itself, gives access to the wealth of online documentation, and gives details of further resources, e.g. books.
- There are some recommended online tutorials at http://www.macs.hw.ac.uk/actuarial/R/ including how to download and install **R** on your computer.

2 Starting and quitting R

Start **R** by double-clicking on the **R** shortcut. This will start the **R GUI** (graphical user interface). Type commands in the **R console** (the command window within the GUI). Exit via the File menu, or the `q()` command. **R** prompts you to save an **image** of your current **workspace** (currently defined **objects**, etc). This is automatically reloaded when **R** is next started.

3 Running R

Within the command **console**, **R** issues its characteristic **prompt**: `>`

The user types **expressions** which are then **evaluated**. The **results** are either displayed immediately, or may be **assigned** (using *either* of the **assignment operators** `=` or `<-`) to **R objects**.

In the examples below **input** typed by the user is preceded by the above prompt—all else is the computer's response:

```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> x = 1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> y = log(x)
> y
 [1] 0.00000 0.69315 1.09861 1.38629 1.60944 1.79176 1.94591 2.07944 2.19722
[10] 2.30259
> x*y
 [1]  0.0000  1.3863  3.2958  5.5452  8.0472 10.7506 13.6214 16.6355 19.7750
[10] 23.0259
> z = x*y
> z
 [1]  0.0000  1.3863  3.2958  5.5452  8.0472 10.7506 13.6214 16.6355 19.7750
[10] 23.0259
> mean(z)
[1] 10.208
> objects()
[1] "x"           "y"           "z"
> rm(x,y)
> objects()
[1] "z"
```

Typing

```
> objects()
```

(a built-in **R function** with, on this occasion, no supplied arguments) produces a list of all user-defined **objects** (variables, datasets, functions) in the current **workspace**.

Objects may be selectively removed with the function `rm`. All the user-defined objects in the current workspace may be removed at once via the Misc menu.

There are other objects stored elsewhere on the system. Some are available for immediate use; these are the in-built functions (e.g. `log`, `mean`, `var`, `plot`) and constants (e.g. `pi`). Others are available via add-on **R packages**—see Section 8.

Previous command lines may be recalled for **editing** via the up-arrow and down-arrow keys.

Long calculations may be **interrupted** via (on Windows) the `<Esc>` key.

4 Managing data in R

Small datasets may be typed directly into the **R** workspace, e.g.

```
> marks = c(53, 67, 81, 25, 72, 40)
> marks
[1] 53 67 81 25 72 40
> mean(marks)
[1] 56.3333
```

Alternatively the function `scan` may be used:

```
> marks = scan()
1: 53 67 81 25 72 40
7:
Read 6 items
> marks
[1] 53 67 81 25 72 40
```

Data already stored elsewhere is either available for immediate use, or may usually be loaded into the workspace via the function `data`. For example

```
> iris
```

displays Anderson's famous `iris` data. Type

```
> data()
```

to see a both a list of immediately available datasets and also instructions on how to find all datasets already on the system.

The functions `data` and `load` will also load data from external files which use **R**'s own storage format. The function `save` can be used to save objects in this format. The function `save.image` can be used to save the entire current **workspace** (see below).

The function `read.table` can be used to read a rectangular array of data stored in an external ASCII (text) file (for files with tricky formatting it may be necessary to use the function `scan`). Similarly the function `write.table` can be used to write a rectangular array of data to an external ASCII (text) file

For help on any of these functions see Section 10.

5 Managing your work in R

The **R workspace** is the current working environment, primarily the user-defined **objects** (variables, datasets, functions).

On exiting **R**, you are prompted to save an **image** of your current **workspace**. This is automatically reloaded when **R** is next started.

In a lengthy **R** session you may wish to save the workspace periodically to protect against system crashes. Use the File menu, or the function `save.image`.

It is sensible (i.e. *essential*) to keep a separate **R** workspace for each of your projects. The easiest way to do this is to install a separate desktop **shortcut** to **R** for each project. Give this a **name** which corresponds to the project.

On Windows systems copy an existing shortcut, change its name, and use the shortcut Properties, Start in entry to correspondingly change the name of the **R working directory**.

There are other ways to keep workspaces separate, but the above is the simplest.

6 R object types

Here is a list of a few of the more important **object types**.

Vector: a one-dimensional array of arbitrary length. Subsets of the vector may be referenced. All elements of the vector must be of the same type—numerical, character, etc.

Matrix: a two-dimensional array with an arbitrary number of rows and columns. Subsets of the matrix may be referenced, and individual rows and columns of the matrix may be handled as vectors. Again all elements of the matrix must be of the same type.

Array: as a matrix, but of arbitrary dimension.

Data frame: a set of **data** organised similarly to a matrix. However each column of the data frame may contain its own type of data. Columns typically correspond to variables in a statistical study, while rows correspond to observations of these variables. A data frame may be handled similarly to a matrix, and individual columns of the data frame may be handled as vectors.

Function: see Section 7.

List: an arbitrary collection of other **R** objects (which may include other lists).

7 R functions

A **function** is one type of **R** object. **R** has a vast number of ‘built-in’ functions. Examples are `mean`, `var` and `plot`, in addition to those functions already mentioned above.

You may write your own functions. These enable you to reduce complex procedures—which you might wish to run repeatedly with differing inputs—to single commands. Examples of such procedures are experimental data analyses and simulations.

To create and edit functions, use the (built-in) **R** function `edit`. (As usual, type `?edit` at the command prompt for details—see Section 10.)

Alternatively the function `source`—also available via the File menu—can be used to input lengthy function definitions from an externally edited file. (The function `dump` will write an existing function definition to a file.)

For details of how to get help on any function, see Section 10.

8 R packages

R is built from **packages** of datasets and functions. A number of packages are loaded by default and contain everything necessary for basic statistical analysis. Other packages may be loaded on demand, either via the Packages menu, or via the **R** function `library`.

Once a package is loaded, the functions and datasets within it are automatically available. To make available a dataset from within a package, use the function `data`

Of particular interest to *advanced* statistical users is the package MASS, which contains the functions and datasets from the book *Modern Applied Statistics with S* by W N Venables and B D Ripley. This package can be loaded with

```
> library(MASS)
```

Documentation on any package is available via the **R help system**—see Section 10.

A useful package containing various actuarial functions is the package **actuar**. This, and other packages that are not on your computer, can be downloaded and installed in **R** using the menu option
`Packages` → `Install package(s)...` and then `Load package...`

9 R graphics

Graphs are created automatically by the appropriate **R** functions, e.g. `plot`.

A graph may be **resized**. It may also be **printed** via the menu obtained by clicking on it with the right mouse button.

On Windows a graph may be **copied** and **pasted** (or **dragged** and **dropped**) directly into a **Microsoft Word** document, **Excel** spreadsheet, or other **Windows** application. To **copy**, use the menu obtained by clicking on the right mouse button in the graph (it is probably best to use the **Windows** `metafile` format). To **paste**, e.g. in **Word**, again use the right mouse button menu.

Note the very different final effects between **resizing** a graph *before* and *after* copying and pasting it.

Alternatively, and perhaps more *safely*, a graph may be **saved** to an external **file** in any one of various formats. Again, the **Windows** `metafile` format is probably the most useful if the file is subsequently to be inserted into another **Windows** application. Insert a graphics file in **Word** via the Insert, Picture, From File menu.

10 R help and documentation

A very comprehensive **help system** is available via the Help menu. In particular:

- Html help starts an **HTML** version of the entire help system (equivalent to typing `help.start()` at the **R** prompt);
- R functions (text) gives immediate help on a particular function or dataset (equivalent to typing, for example, `?plot` at the **R** prompt);

The standard manual for **R** is *An Introduction to R* by W. N. Venables, D. M. Smith and the R Development Core Team. It is available in the following formats: as HTML via the **R** Help, R language (html) menu; as PDF (best for printing), via **R** Help, Manuals menu, or from the **R** homepage, where you can always get the latest version.

Further resources on **R** are available from the **R** homepage at http://www.r-project.org/.

11 Useful reading

- Verzani, J. (2005) Using R for Introductory Statistics. Chapman & Hall/CRC.
- Schumacker, R.E. (2014) Learning Statistics Using R. Sage.
- Faraway, J. (2006) Extending the Linear Model with R :Generalized Linear, Mixed Effects and Nonparametric Regression Models. Chapman & Hall/CRC.