Using R in the Mathematics computing lab: Statistics 1

Introduction

This document gives general information about accessing the statistical package **R** in the Mathematics computing laboratory. It also includes some basic commands for accessing, saving and displaying data. Futher commands will be introduced as required in the handouts for Statistics 1.

A fuller introduction to the **R** language is given in the attached Self-learn Tutorial. Alternatively, you may find it helpful to consult the book **Introductory Statistics with R** by **Peter Dalgaard** (published by Springer), especially if you intend to take Statistics options in your second, third or fourth year.

1 Accessing R on the Mathematics PC's

To access **R**, first log on to a Windows PC in the computer lab. You should have already been given a user ID and password.

To start **R** double-click the **R 2.2.1** icon on the desktop. This will bring up a large window containing a smaller window: the **R** *console*. You type all the commands into this window; other windows may appear during the course of your session. The **R** prompt is usually `>`.

Most questions that you have about **R** can be answered by pulling down the `help` menu, or sometimes just by typing `help(name)` or `?name`, where `name` is the topic name.

To exit **R** either type `q()` in the commands window or select `File | Exit` (that is, `Exit` on the `File` menu). You will be asked if you wish to save the 'workspace image': a 'no' answer is appropriate since you will not be allowed to save as a default. If you have created some **R** objects that you wish to use again, select 'cancel', and see below to save your image. Do not forget to log-off (by typing ctrl-alt-delete, and then the `logoff` button).

2 Correcting mistakes

If you have entered a long command and find there was a small mistake in it, you can easily correct the command without retyping the whole line. Press the up-arrow on the keyboard to take you back to the previously entered command. Use the right or left arrow keys to take the cursor to where you want to insert or correct text. Use the Delete or Backspace keys to delete incorrect text, or just type the new text you want to insert. Finally, press Enter to enter the revised command.

Note that you can repeat and edit any previous commands by using the up- and down-arrow keys. E.g. pressing the up-arrow key k times will take you back to the kth previously entered commands.

3 Accessing, Entering and Saving Data

Accessing the Statistics 1 data sets

Most of the data sets used in the Statistics 1 unit are contained the data file `stats1.RData`. To access these data sets type in the **R** console window (you do not type the prompt `>`)

```
> attach("//arthur/R/stats1.RData")
```

Note that this command links you to the Statistics 1 data sets but does not copy them to your personal filespace. Ignore the onscreen comments about some files being 'masked' – this just

indicates that there are similarly named files in some of the other **R** packages. When you quit **R** the link disappears and you have to re-attach the data sets with the above `attach(...)` command each time you want to use them.

The command attaches the Statistics 1 data file in position 2 in a heirarchy of accessible files, folders etc. Once you have attached it, you can list the individual data sets by typing

```
> ls(pos=2)
```

and you can inspect or operate on individual files by typing their name, e.g.

```
> quakes
> hist(newcomb)
```

Accessing other data sets

The **R** package also includes a number of built-in data sets. You can list these by typing

```
> data()
```

and you can access one of the individual data sets, for example `faithful`, by typing

```
> data(faithful)
> attach(faithful)
```

Entering Data

Two basic functions for creating data vectors are the `c` function and the `scan` function. Say you wanted to ceate a data vector `mydata` with the five enties `0.1 2.3 4.5 7.2 6.6`. Using the `c` function you would type (here `<-` is the assignment operator in **R**, corresponding to `=` in Fortran or C).

```
> mydata <- c(0.1, 2.3, 4.5, 7.2, 6.6)
```

Alternatively you could enter the data by typing `mydata <- scan()` followed by ENTER, then typing each data value followed by ENTER (so each value is on a new line) and finally typing ENTER twice after the last data value to let the package know you have finished data entry. Thus you would type

```
mydata <- scan()
1: 0.1
2: 2.3
...
5: 6.6
6:
```

Saving Data

The workspace image contains all the datasets created in the current session. If you have created datasets that you wish to keep for a subsequent session, you should explicitly save the workspace image before exiting. The default location for doing this, which is offered to you when you type `q()` (see above) may not be appropriate; instead you should (1) select `File | Save Workspace` then (2) select `My Documents` in the left hand pane, give the file an appropriate name(e.g. `ProblemSheet1` in the `File name` box and finally (3) click `Save`. It will automatically get a `.RData` item extension. Note that this does not save attached files - only files created or copied into **R** during the current session.

To recover this material in a subsequent session, after signing on to **R**, (1) select `File | Load Workspace` then (2) select `My Documents`, and finally (3) double-click on the `.RData` item, (e.g. `ProbSheet1`).

You can also save or recover a workspace image with the `Save Image` and `Load Image` buttons on the main **R** toolbar.

4 Printing

Many of your interactions with the computer will be exploratory, and you will not want to save or print the results. But other output will be wanted. There are several ways to get printed output.

If you simply want to print the contents of a graphics plot, select the graphics window containing the plot and note that it brings up a different toolbar from the main **R** console, then click on the printer tool-button, or on `File | Print`.

Warning! You must be sure to give your plot a title which includes your identifier, e.g. `title("Plot of data (zy4321)")` or `main="Plot of data (zy4321)"`

Sometimes you may want to print part of the contents of the **R** console. Do not select `File | Print` with the **R** console on top as this prints the entire contents of the session, not just what is currently visible on the screen, so can produce voluminous output.

For more selective printing, the easiest approach is to 'copy-and-paste' text from the **R** console into an editing program, for example Notepad. (You can start Notepad by typing Notepad into the box after clicking `Start | Run`). Highlight the required text by dragging the mouse over it with the left button depressed, select `Edit | Copy`, then focus on the Notepad window and select `Edit | Paste`. You can then tidy up the result as you wish in the Notepad window, and add your name and computer ID before sending it to the printer by selecting `File | Print`.

5 Plotting

Suppose you want to make a plot of the square roots of the integers $5, 6, ..., 15$. First, create a vector containing the square roots by typing

```
x <- sqrt(5:15)
```

Then plot `x` with

```
plot(x)
```

You should see a new window containing a plot of `x[i]` against `i`. You can customise the plot by adding labels and specifying that you want a solid line to join the points, rather than asterisks at each data point as follows, where the `type` is `l` for `line` rather than the number 1.

```
plot(x,type="l",xlab="i",ylab="x[i]")
```

To see what other options are available, you might want to type `?plot` or `help(plot)` or `?par` or `help(par)` in the console window.

To send your plot to the printer, make sure the graph window is on top, then select `File | Print` as above.

Again, you must be sure to give your plot a title which includes your identifier, e.g. `title("Plot of data (zy4321)")`.

6 A practice session with R

The **R** system is a sophisticated interactive package for statistics and graphics, with its own programming language built in. There should be no need to learn this language formally, as it works quite intuitively. The following practice session is intended to introduce you to the main ideas. Try typing these instructions (in order, down the columns) and observing the resulting output. If you do not see what is going on, experiment by varying the input a little to see the effect. In very broad terms, **R** is like a powerful calculator. If you type an expression, it is evaluated; the result is then printed, unless it is assigned to a variable. Remember, assignment is achieved using `<-`.

```
2*3                     sort(a)
z <- 2*3                sum(a)/length(a)
z                       sample(17)
a <- c(1,7,4,5)         sample(10,5)
a[2]                    rnorm(20)
a[2:4]                  z <- rnorm(20)
a[c(3,3,2)]             mean(z)
a[-2]                   var(z)
b <- 1:4                median(z)
a+b                     stem(rnorm(20))
a*b                     x <- 6*pi*(0:200)/200
a%*%b                   plot(x,exp(-0.09*x)*sin(x))
m <- matrix(1:12,3,4)   plot(cumsum(runif(200))/(1:200))
m                       x <- 1:10
m[2,c(1,3)]             y <- 3.4+1.2*x+rnorm(10)
nrow(m)                 plot(x,y)
m^2                     abline(3.4,1.2,lty=2)
log(m)                  abline(lsfit(x,y))
m%*%a                   hist(rgamma(500,2))
```

If you cannot understand what any of the functions above does, try the **R** help system.

7 Getting help with R

A lot of **online** help is available. In particular, for instructions on how to use a particular function, select `Help | R language` and for a long discussion of general issues, follow `Help | Manuals | An Introduction to R`.