MATH11400 Statistics 1 2009-10

Homepage http://www.stats.bris.ac.uk/∼mapjg/Teach/Stats1/

Using R in the Mathematics computing lab – Introduction

This document gives some basic commands for accessing, saving, analysing and displaying data using the statistical package **R** in the Mathematics computing laboratory. Further commands will be introduced as required in the handouts for Statistics 1. A fuller introduction to **R** is given in the **Self-learn Tutorial** or in the book **Introductory Statistics with R** by **Peter Dalgaard**.

1 Accessing R on the Mathematics iMACs

You should have already been given a user ID and password for the iMACs in the computer lab. To start **R**, double-click the **R** icon in the left-hand toolbar. This will bring up a window containing the **R** *console*. You type all the commands into this window but other windows (e.g. help, plots etc.) may appear during the course of your session. You can resize the window as necessary. 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 click the red button in the top left hand corner of the **R** console or type `q()` in the console window. You will be asked if you wish to save the 'workspace image'. Usually it is best not to save, as saving it may overwrite your current data files. If you have created some **R** objects that you wish to use again, select 'cancel', and see below to save your image.

2 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 you could (a) type the following command in the **R** console window

```
load(url("http://www.maths.bris.ac.uk/~mapjg/Teach/Stats1/stats1.RData"))
```

or (b) use a web browser to go to the Statistics 1 homepage

```
http://www.maths.bris.ac.uk/~mapjg/Teach/Stats1/
```

and select the **stats1.RData** link in the Statistical Computing section.

Notes: For (a) if you cut and paste the ~ character may need to be corrected and retyped - see **Correcting mistakes** below; for (b), this works with IE and Firefox browsers, but there may be problems with Safari.

Once you have loaded the workspace, you can list the individual data sets by typing

```
> ls()
```

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

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

Saving Data
The workspace image contains all the datasets created in the current session. It may be best to keep an unmodified copy of the stats1.RData file in your own workspace, and load it afresh (double-click the file) to do the work for each new problem sheet.

If you wish to keep a modified dataset to come back to, you should explicitly save the resulting workspace image with an appropriate new name before exiting. On the menubar select `Workspace | Save Workspace File` and give the file an appropriate name with an .RData extension (e.g. `Sheet1.RData`) in the `Save As` box. To reload this material in a subsequent session, again just double click the relevant .RData file.

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)
> faithful
```

Entering Data
Two basic ways of creating data vectors are the `c` function and the `scan` function. To create a data vector `mydata` with the five entries `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 show you have finished data entry, i.e. type:

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

3 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. 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 return you to the kth previously entered command.

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 top menubar from the main **R** console, then select `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)"`

If you may want to print part of the contents of the **R** console, do not select `File | Print` with the **R** console as the selected window. This can produce a massive output – it prints the entire contents of the session, not just what is currently visible on the screen! Instead, the easiest approach is to 'copy-and-paste' text from the **R** console or plots from a plot window into an editing program, for example TextEdit. (You can open TextEdit from the left-hand toolbar). You can then tidy up the result as you wish in the TextEdit window, and add your name and computer ID before sending it to the printer by selecting `File | Print` from the active TextEdit window.

5 Plotting

Suppose you want to make a plot of the square roots of the integers $5, 6, ..., 15$. The following commands first create a vector `x` containing the square roots and then produce a plot of `x`:

```
x <- sqrt(5:15)
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 plot window is active, then select `File | Print` as above. Remember to give your plot a title which includes your identifier, e.g. `title("Plot of data (zy4321)")`.

6 Downloading R for your own computer

R is a freeware system; you can download it onto your own computer, and you may find it more convenient to do your coursework that way. The following guidelines are for Windows users. For Mac users, the procedure is similar, but with different detailed instructions as appropriate. Start by going to the website

`http://www.stats.bris.ac.uk/R`

In the `Download and Install R` section select `Windows` precompiled binary distributions, then `base` distribution, then `R-2.8.1-win32.exe` (ignore the Security Warning), then choose your language, then follow the Setup Wizard instructions.

You can accept most of the default options - in particular, the location
`c:\Program Files\R\R-2.8.1`
is a good place to install the files. As well as the default options in the 'Select components' menu, it may be useful select `PDF help pages` in the `Online PDF Manuals` section.

When the installation is complete, you should have a desktop or Quick Launch icon displaying the **R** symbol. It is a good idea to edit the Properties of this icon (right-click on the icon to get the menu where you can choose this), and edit the entry in the 'Start in' box under the 'Shortcut' tab to specify a directory of your choice for your work.

7 Getting help with R

I suggest you print off and read the **Self-learn Tutorial** in the **Statistical Computing using R** section the Statsistics 1 homepage at http://www.maths.bris.ac.uk/∼mapjg/Teach/Stats1/

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

8 A practice session with R

The **R** system is a sophisticated interactive package for statistics and graphics, with its own programming language built in. In very broad terms, it is like a powerful calculator – there should be no need to learn this language formally, as it works quite intuitively. To introduce yourself to some of the main ideas, try typing the following 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 or try the **R** help system.

```
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))
```