The Friendly Beginners' R Course

written by Toby Marthews at the BCI Research Centre, Panama (http://www.stri.org)

This course is only 14 pages long (inc. pictures) and you work through it in your own time so it's probably the least painful introduction to R currently around. Make sure you have the example files that accompany this text ("first.r", "mystery.r", "quadrats.r" and "quadratdata") otherwise many things won't make sense. Start reading below the line of stars and all should be self-explanatory including how to install R in the first place (if necessary).

Toby, August 2005 (last updated May 2014)

****

SO - you've decided you want to learn to use the R language and environment? Well, hmmmmm ... would it perhaps be more accurate to say that either a) your boss/ supervisor/ advisor has told you that you have to and you have a very bad feeling about the whole idea, b) you have some analysis to do and a friend has promised - against all your common sense - that R is easy to use and can help you, c) you have tried reading statistics or modelling books, have given up and are desperately hoping that R is a way around them or d) you've just decided to increase your egg-head rating and impress people?

Whatever your reasons, I think learning to use R *is* a good idea - if only to be aware of what a software suite like this can do. R does a lot of very clever things and *can* make your life easier if you have to analyse data a lot. The egg-head bit is also a good point: since I put it on my CV everyone believes I'm much cleverer than I really am.

Some comments for those who think R is 'just another' statistics package: Well, R is *both* a programming language *and* a means to do statistical analysis and this is partly why I think it's a step ahead of anything else around at the moment: by learning R you will acquire programming skills (these skills are 70-80% of what people learn - or should learn - in modelling courses) *and* the ability to do statistics on a computer. So, by learning both together, you can gain two sets of skills for the price of one (I wrote this paragraph in 2005 but I noticed in a 2009 article "Shock and Awe by Statistical Software - Why R?" by Owen Petchey, Andrew Beckerman and Dylan Childs in *Bulletin of the British Ecological Society* 40 (http://www.academia.edu/2673156/Shock_and_Awe_by_Statistical_Software-Why_R) they made similar comments and suggested that R could replace all of Sigmaplot, MS Excel, SAS, Genstat *and* Mathematica !). Also in 2009 the New York Times ran an article on R: http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html?_r=1&em=&pagewanted=all.

DON'T BUY AN R TEXTBOOK (at least not before you finish these pages): firstly because there is a 2300 page R manual downloadable for free from the R website http://www.R-project.org and secondly because R is not 'new statistics' but a way of doing standard statistics more quickly, so you can just use a STANDARD textbook, if you like, adding notes to it as required. R is similar to (and in some ways is a freeware alternative to) MATLAB (http://www.mathworks.com/products/matlab; for a comparison of the two you can look at http://germain.umemat.maine.edu/faculty/hiebeler/comp/matlabR.pdf). For users of SAS, SPSS, Stata or Systat, "Quick-R" (http://www.statmethods.net/index.html) explains why R can be useful to you too and http://www.burns-stat.com/pages/Tutor/R_relative_statpack.pdf talks about the relative merits of R *vs.* SAS, Stata and SPSS.

After reading the following few pages you should be able to write your own R scripts, use some R functions, draw some nice graphs and use some of R's capabilities. This guide is written for someone who's USED A COMPUTER BEFORE but has NO PROGRAMMING EXPERIENCE (if you do have some experience, you'll know which sections to skip below).

I can't say how long this text will take to work through (everybody's different), but there are only 6 challenges so hopefully not too long. Set yourself up with a computer, a printout of this text, a strong coffee (or alternative stimulant) and go through the sections one-by-one starting with ....

Installing R & Running an R Program

You need a bit of general knowledge of computers and how they work. If you already know about computer languages and workspace directories and have R installed on your computer then go on to the next section.

Computer programs are always written in some kind of computer language. Computer languages are either script ones (e.g. BASIC, JavaScript, R) or compiled ones (e.g. FORTRAN, PASCAL, C, C++, Java) and whichever one a programmer is using, it all has to be translated into machine code (which is a stream of 1s and 0s) before the computer can actually 'execute' or 'run' it (= do it). Here's where the difference lies: with script languages the computer goes through the program line-by-line and translates and executes each before going on to the next line; with compiled languages the computer translates the whole program in one go, saves the machine code as an 'executable' on disk (in Windows usually with a ".exe" extension) and then runs the executable directly.

Generally speaking, script languages are *slow* but *more user-friendly (esp. error-reporting)* and compiled languages are *much faster* but *are much less straight-forward to use*. So, if you write a program in R then it'll run a lot slower than an equivalent program written in C or FORTRAN - and you should be aware of this - *but* a) the difference will only be noticeable to you if you're doing really lots of calculations, b) if you've never used a computer language before then you'll be pulling your hair out if you start with something like FORTRAN, c) in the case of R there are all these extra features like graph-plotting and statistical functions that can make your life a *lot* easier (and FORTRAN, for example, can't do those without special add-ons like IDL) and d) if you learn how to program using a language like R then you'll find it very easy to pick up any other computer language afterwards because all languages have similar structures (repeat loops, for loops, if statements, etc.).

That's all just to set the scene: now let's actually do something. Here's how to install R on your computer. I've done instructions here only for 

(apologies to non-Windows users!) although R is also available for MacOS and Linux too.

INSTALLATION FOR Windows :

1. Go to the R website http://www.R-project.org, click on Download/CRAN on the left and choose a mirror site geographically near to you (to reduce download time). Choose "Download R for Windows", click on "base", download the Setup Executable (click on "Download R x.x.x for Windows", where the "x"s are numbers) and save it on the Desktop (an ".exe" file). Double-click on this to run the installation (accept the default startup options).

2. Say YES to a desktop icon but NO to a Quick Launch icon. R is now on your computer (and you can delete the "R-x.x.x-win...exe" file on the Desktop).

DO THESE STEPS ***EVEN IF YOU HAVE R ALREADY INSTALLED on your machine*** 

:

3. Create a workspace directory on the Desktop (or elsewhere if you prefer) for using R (right-click on the Desktop background, choose New → Folder and give it a name something like "Rwork") and copy "first.r" (accompanying this text) into it. This directory is used by R for storing variables and function definitions (in a file called ".RData") so you have to have one (oh, and "A → B" is my way of saying "go to menu A and select B from it"). WATCH OUT: in a particularly annoying way, some windows systems automatically rename email attachments called "xxx.r" as "xxx.r.txt" or "XXX.R.TXT" when you save them and you need to keep renaming them back to "xxx.r".

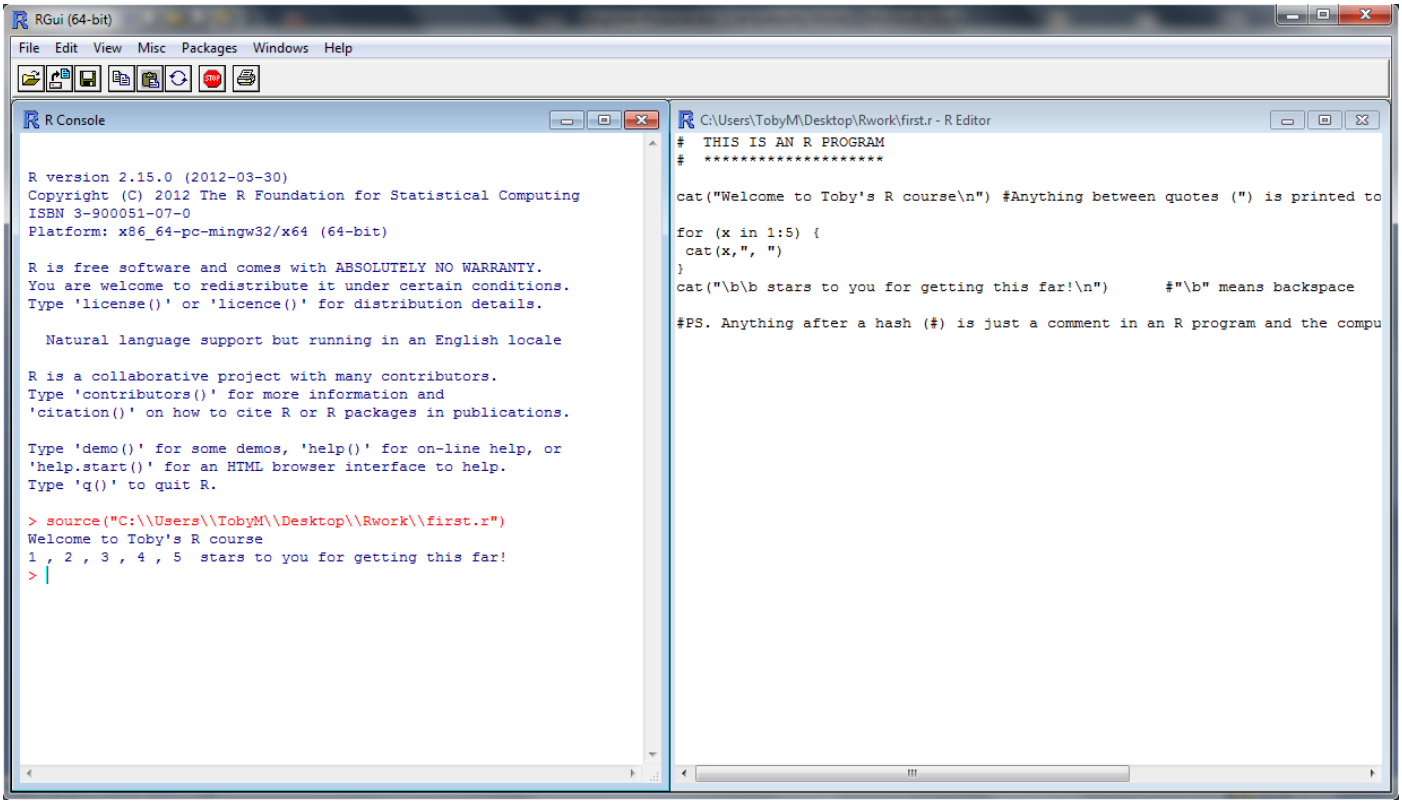
4. Right-click on the desktop shortcut that should have appeared during installation (called something like "R x64 x.xx.x", which I usually rename "Start R"), and choose "Properties". Leave the "Target" as it is, but modify the "Start in" box so that it has the location of the workspace directory you created in the last step and click "Apply". Next, open the RGui by double-clicking on the "R x64 x.xx.x" shortcut ("Gui" = "Graphical User Interface"). By looking at File → Source R code..., check that R opens in the right folder (the window that appears should be the directory from the last step: if it is, just cancel without sourcing any files). If you want a Quick launch icon on the task bar as well, use the mouse to drag the desktop shortcut on to the task bar (normally just to the right of where the "start" of the Start Menu is).

5. Now start up R (i.e. double-click on "R x64 x.xx.x" or "Start R"). Test whether R can run a simple program: use File → Source R code... in the File menu, find first.r in the workspace directory and open it. R will run the program and you should get a welcome message (the file first.r is just a text file, by the way, as you can see if you open it in any text editor).

6. Not quite finished yet: go to File → Open script... and choose first.r. An R Editor window should open up to allow you to change the program (I need to check you can do this too). Find the "5" on line 6 and change it to a "10". Save it by going File → Save as... and save it under the name "first2.r" (then close the editor window).

7. Now run first2.r in the same way as in Step 4. If you got 10 stars then you're doing well and you deserve them!

8. You can exit R by clicking on the red "X" or by typing "q()". For now, you don't need to save the workspace image (in fact, throughout this Beginners' course, you can always say NO to saving the workspace).



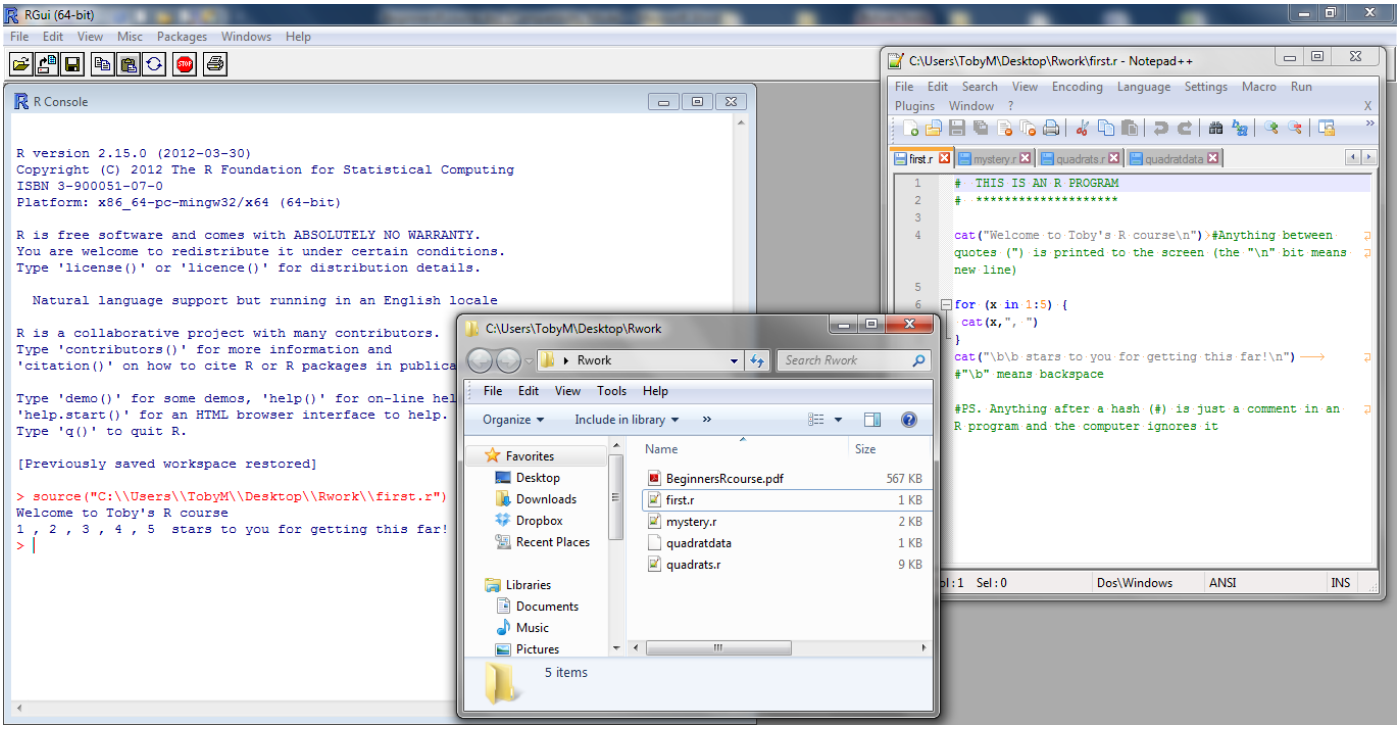
Two Windows: Console & Editor

With the heady feeling of success from having run your first R script, I'm sure you'll be wanting more, more, *more!* Well, just to get you used to what we've done up to now, please could you open up the original first.r into your editor again. See if you can manage to do the following two:

Q1. Can you make the FOR loop count *down* from 5 to 1 instead of up from 1 to 5?

Q2. Can you make it count up 1 to 5 and then down 4 to 1 (which is easiest to do using two FOR loops one after the other)?

If you try those two questions (I know they're tedious: you've got to learn to walk before you can run) then you'll have to get used to the way R programmers keep two windows open at once: you edit the program in an "editor" window, then save it, flip to the "console" window (aka. "terminal") and run the program from there (Windows version only: note the different "File" menus depending on which window is active). This is the way programming is done in a lot of languages, by the way, and many people resize and move the two windows so they are as large as possible without overlapping.



The R in-built text editor (the "editor" window) is very basic and *I don't recommend you use it* unless you really have to (although note that it is sufficient for getting through all of this course). There are many much better editors that are free to install, e.g. above is a screenshot with the first.r open in Notepad++ instead of the built-in editor (some people end up using MS NotePad and MS WordPad, but these are really not much better than the built-in editor for text editing[1]). I use and recommend Notepad++ (a free download from http://notepad-plus.sourceforge.net/uk/site.htm[2]), which is just great[3]. Please believe me that to do programming without a proper text editor is making life *unnecessarily* hard for yourself!

Please don't skip Q1 and Q2: they're there to force you to check that the editing-saving-running process works OK on your version of R and you need this to be working for what follows. If it doesn't work then please re-check what you've done so far and/or panic and call for help (try the FAQs about installation on http://www.R-project.org). A "syntax error", by the way, means there's something wrong in the code you're editing: check for typos, unclosed brackets and other things like that.

By the way, I think I ought to mention at this point that when you installed R, it also installed a set of Beginners' documentation and Frequently Asked Questions (FAQs) on your machine. You can have a look at these at any time by typing

```
help.start()
```

[1] If you do end up having to use WordPad, be careful to turn off the "smart quotes" facility: copying cat("Hello\n") into the Console window will give an error: you need to copy in cat("Hello\n"). Also, be aware that when you save in .txt format these programs use *Windows*-format textfiles rather than normal textfiles (see http://en.wikipedia.org/wiki/Text_files#Standard_Windows_.txt_files), which may cause you problems if you're doing something complicated (e.g. UNIX scripting), but for now you should be OK.

[2] To get the defaults I use on Notepad++, go to Settings→Preferences, make sure "Display line number margin" in the "Editing" tab is ON, click OFF "Auto-indent" in the MISC tab and also "Don't check at launch time", go to the "New Document/Default Directory" tab and make sure the format is "Unix" (rather than "Windows") and in the "General" tab check "Multi-Line" and "Show close button on each tab" too. Then go to the Encoding menu and check the encoding there is "UTF-8 without BOM". Then go to the View menu and click ON "Word wrap" and "Show Symbol"→"Show White Space and TAB" and "Show wrap symbol". Additionally, I strongly recommend installing NppToR (http://npptor.sourceforge.net/) along with Notepad++, which will give you syntax highlighting for R (however, only works in the default theme).

[3] I'm aware of other R users who use **ConTEXT** (http://www.contexteditor.org), **TextPad** (http://www.textpad.com; **NOT FREEWARE**), **Tinn-R** (http://www.sciviews.org/Tinn-R) and **Crimson Editor** (http://www.crimsoneditor.com), but even though Tinn-R and Crimson Editor have syntax highlighting for R, and TextPad offers it as an add-on (http://www.textpad.com/add-ons/files/syntax/r.zip), I still prefer the combination of **Notepad++ and NppToR**. Other favourites are **Eclipse+StatET** (http://www.walware.de/goto/statet), **Emacs+ESS** (http://en.wikipedia.org/wiki/Emacs_Speaks_Statistics), **Vim** (http://www.vim.org/) and I've recently been trying **Sublime Text** (http://www.sublimetext.com/). More options are on http://www.sciviews.org/_rgui/projects/Editors.html.

into the Console window. There's hundreds of pages of information there, but *you don't need any of it just now* because you are already reading *this* Beginners' course *which will tell you everything you need to know (!)*. I feel I ought to mention it because it's there and if you really can't get through my short Beginners' course then that's the place to look, but since you're already a fair way into *this* course, why not stick it out to the end and find out what all those stars are for?

The R Manual

While you're concentrating on first.r to answer those questions, please make sure you can understand what *every* line does. I haven't explained everything in my comments there (the # lines) because you need to get into the habit of using R's very comprehensive manual system. There are no annoying paperclips, funny dogs or wizards. Here's how to use it:

Imagine you are sent an R script and you open it in your text editor to try to figure out how it works. Say the first line is:

```
a=seq(-2,4,length.out=5)
```

but you don't know what this does yet. The command here is "seq" (the bracket afterwards contains the arguments 'passed' to this command) so the first thing you would do is open up the R manual page for seq by typing "?seq" into the Console window. The manual page will then appear (in Windows it appears in a new window, in Linux in the same window: you press "q" to go back to normal). These manual pages are generally written in a pretty technical way, *but* you don't usually have to read much of it: ignore the text and scroll down to the bottom to see the examples (the first one on the seq page is "seq(0, 1, length.out=11)". The examples are the best bit of the manual page to start with because you can copy them into the Console window to see what they do (in Windows mark the example you want with the mouse, do CTRL+c to copy, click on the Console and do CTRL+v to paste; in Linux mark it and do Edit → Copy, then q, then Edit → Paste). Do this with the first of the seq examples:

```
seq(0, 1, length.out=11)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

It doesn't take a genius to work out from this that the "seq" command makes a sequence of numbers, so that mysterious command in the program you were sent probably creates a sequence of 5 numbers from -2 to 4 and stores it in a variable called "a". You can confirm this by typing:

```
> a
[1] -2.0 -0.5  1.0  2.5  4.0
```

Now try finding out about a different command:

Q3 Copy and try out the "Discrete Distribution Plot" example at the end of the "plot" manual page and the "setting row and column names" example from the "matrix" manual page.

If there doesn't appear to be a manual page for a particular command (e.g. typing "?normality test" doesn't work), there is a search facility you can use: type "help.search("normality test")" and this should lead you quite quickly to the command "shapiro.test" and the man page ?shapiro.test. which will tell you how to use it (note the two examples at the bottom of that page).

Perhaps a more user-friendly way of searching for help is to download the "R Reference Index" from the "Manuals" part of the R website (http://www.R-project.org): this is in PDF format and you can search for words in it using CTRL+f. You can also search for more general information: try typing "RSiteSearch("normality test")" into the Console window to search the R website and also put the same

search into the search boxes on Rseek (http://www.rseek.org/), R Bloggers (http://www.r-bloggers.com/) and MarkMail (http://r-project.markmail.org/search/).

Another example: a standard kind of statistics plot is a box plot, but at the moment you don't know how to do this in R and if you type "?box" you don't get the right manual page. Typing "help.search("box")" or "??box" into the Console window, however, or searching for "box" in the reference index will lead you to the keyword "boxplot" which is the right one to use (and both sources give you examples to try too). Sometimes it's not so clear how the examples work, but generally they are very helpful (e.g. the examples at the end of the ?boxplot page use data sets called "InsectSprays", "OrchardSprays" and "ToothGrowth" that are pre-loaded whenever R starts up: this doesn't mean the examples won't work if you copy them into the Console window, but it's not so clear where the numbers come from until you type the name of the data set into the Console window to see what the data set contains).

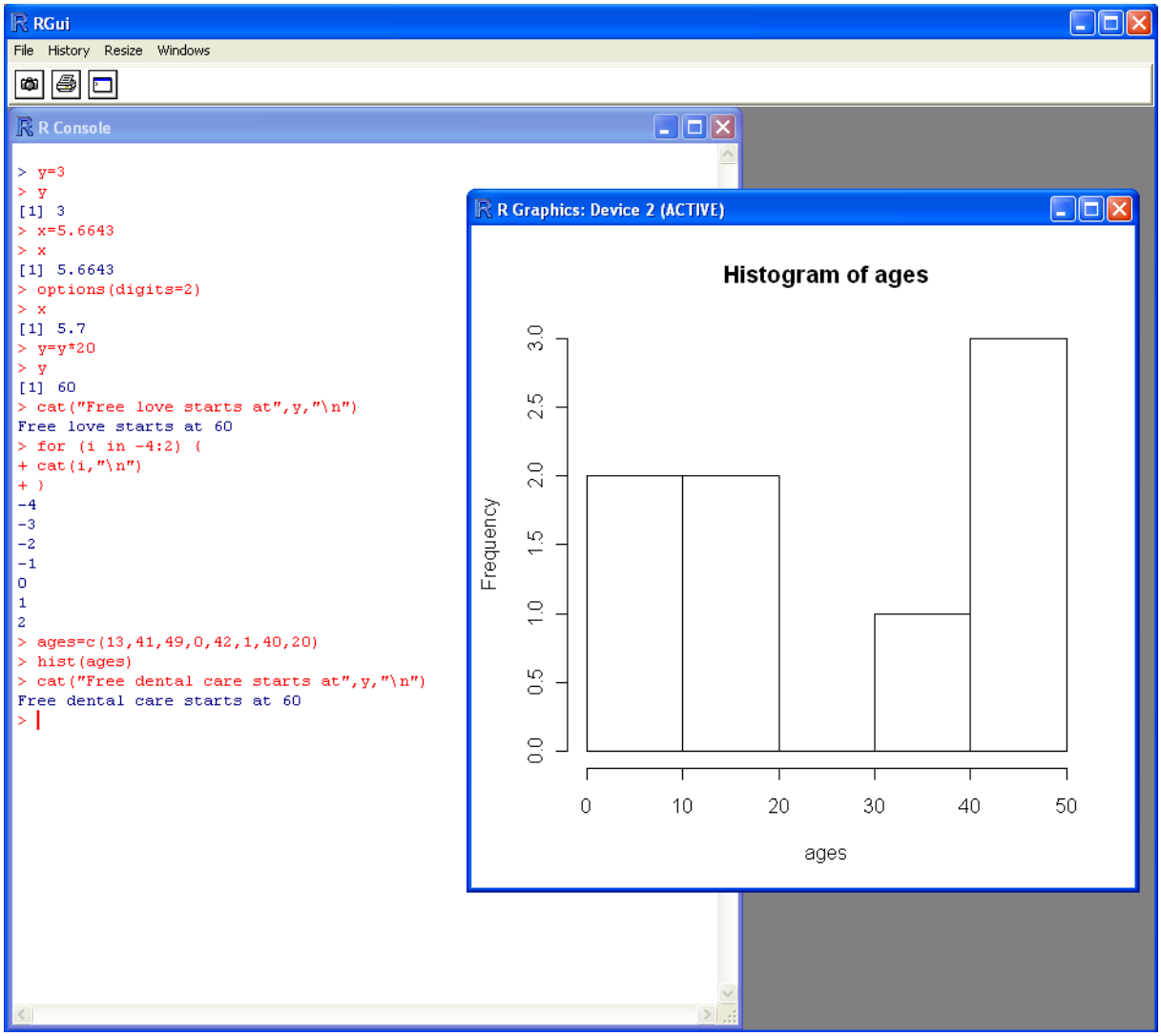
By the way, all R commands work the same way: they have some options (the "Arguments" list on the corresponding man page) and you change the options to get exactly the result you want. You might notice that some commands have very many options, and this is why R is not menu-driven: it would simply be impossible to make menus with that many options on them! Everyone would agree that command-line-driven software like R isn't as user-friendly as menu-driven software, but the alternative is to have a much-restricted set of options and that means you simply can't do what you need to do.

Putting Commands Straight into the Console Window

I hope you like these short, easily-digestible sections by the way: I'm trying only to tell you what you *need* to know to use R. Just to make sure everyone is following, I'd better give the answers to Q1, Q2 & Q3: for Q1 just change the "1:5" to "5:1", for Q2 you have two loops with the first going up (1:5) and the second going down (4:1), for Q3 you should get a pretty graph ("rpois(100,lambda=5)" means 100 draws from a Poi(lambda=5) distribution - we'll get to this sort of thing later) and a 2x3 matrix with 1,2,3 on the top row and 11,12,13 on the bottom row. All those who got these answers get 10 stars (are you keeping track of your stars?).

Next, please click on the console window and type in "y=3" and ENTER. Now type "y" and ENTER. Now type "x=5.6643" and "x". Now "options(digits=2)" and "x". Now type "y=y*20" and "y" again. Do you see what's going on? You can put commands in straight like this. Now type "cat("Free love starts at",y,"\n")" and "for (i in -4:2) {". The prompt has changed from ">" to "+", which means R has found an incomplete command and you need to type more in, so type "cat(i,"\n")" followed by "}". You get the idea, I think. Type "ages=c(13,41,49,0,42,1,40,20)" followed by "hist(ages)" to get a quick glimpse of R's statistical side. Also, there is a "history" function whereby you can press the up and down arrows to find, modify and re-use a previous command: click back on the Console window and press the up arrow a few times to get the first "cat" command in this section, change "love" to "dental care" and press ENTER to get "Free dental care starts at 60" which is, of course, what I meant to say really. Use CTRL+L to clear the console window (clearing up).

This facility of being able to try out any command directly is really useful and one of the main reasons for using a script language (you can't do it so easily with compiled languages). If you're given a program containing lots of incomprehensible command lines (as may very well happen in the next section ...), you can try out the lines one-by-one by copying them into the Console window and seeing what they do.



A Mystery Program

Time for another challenge. Have a look at the program mystery.r (accompanying this text): your task is to work out what it does.

When trying to figure out what a program does, the first thing to do is to run it (save it in the workspace, start R, use source("...") in the File menu, etc.). The second thing to do is to open it in a text editor and see if you can figure out from the code and code-comments what it's trying to do. You'll get another 15 stars if you can tell me:

Q4 What does the program mystery.r calculate and put in the NND column of its printout at the end?

At this point many people might be saying "whoa – what?" because we've suddenly jumped to something with sin, cos, abs, sqrt and pi in it (maths ....), a couple of control loops (the for (…) { … } bits) and arrays (the variables with something in [ ] after them). Don't panic (yet) – just find any commands you're not familiar with, check the manual pages to see what the keywords are supposed to do (type "?sqrt" and copy the example at the end into the Console window to see what it does, etc.) and, if that doesn't help, copy the whole line you're not sure about into the Console window to see if it works there too (most will), alright?

Q5 By inserting "tiff(file="plot1.tif")" just before the plot command and inserting "dev.off()" just after the abline commands, the plot should appear in your workspace directory as a .tif graphics file instead of being plotted on the screen (TIF or TIFF is the format you need to submit something to a journal). Type "?tiff" to find out how to export as a .bmp or .png file too. This is very useful because it allows you to save plots for use in other programs. Another 5 stars if you manage this too. If you're super sharp and have

noticed that you can right-click on the plot window and cut & paste it into MS Paint and Save As various image types without fiddling with that tiff command then I'm afraid *no stars for you!*: go back and do it with the command like I said because it's good to know how to do it that way too. Jeepers - people can be a bit too clever sometimes ... (:-9).

So, how are you feeling about this gentle art of programming? Far too hard or insultingly easy? Keep count of your stars and pat yourself on the back if you got this far because you're doing well. Take off a star for every bit of help you got from someone else, by the way: I'm watching you. The answer to Q4 is at the end of this text, written backwards, but DON'T LOOK NOW: work it out first!

Vectors

OK, I think you need a breather!

This is an easy section just to tell you more about "vector" data types, which are the c(...) lists of numbers you've already met. Click on the Console window and type "b=10;c=11;d=12" (the semi-colon is just a way of putting more than one command on one line). Now type "vec=c(3,b,b,8)" and "vec" and you can see that vec now holds a list (aka. vector) of numbers. You can access the numbers directly too: type "vec[2]" and "vec[4]" and "for (i in 1:4) {cat(vec[i],"\n")}" (nb. 3 different types of brackets and they have to be in the right places).

Now type "vec2=c(1,2,3,4)" and "vec3=vec+vec2" and "vec3" and "vec4=(vec*vec2)+6" and "vec4" and "vec5=c(rep(3,times=20))" and "vec5" and if you can follow what's going on with all that then you know pretty much all you need to know about vectors.

Matrices

After vectors come matrices. This is not the enslaving-the-human-race, world-simulating, karate-kicking kind of matrix, but a more mundane & traditional mathematical matrix, i.e. just a grid or array or numbers. Type "mat=matrix(c(7,8,2,3,4,5,6,3,2,1,-2,-9),nrow=3,ncol=4)" and "mat", which should give you a matrix. Now type "mat2=matrix(c(7,8,2,3,4,5,6,3,2,1,-2,-9),nrow=3,ncol=4,byrow=TRUE)" and "mat2" and "mat3=matrix(7,nrow=3,ncol=4)" and "mat3" and "mat4=matrix(3:14,nrow=3,ncol=4)" and "mat4" and "mat2+mat3" and "mat4*2" and phew, but I think we'll stop there!

All these commands about vectors and matrices can be used in a proper program in just the same way as they work in the Console window. If you feel that hell would freeze over before you voluntarily did anything with matrices then fair enough: we'll have no more of them here.

Miscellaneous

Here are some useful commands to try out. Here's a way to control the number of decimal places on what you print out:

```
cat("x=",formatC(10.5,width=8,format="f",digits=3),"\n")
```

Here's some string manipulation:

```
substring("abracadabra",3,8)        # Extract characters 3 to 8
nchar("abracadabra")                # Count the number of characters
strsplit("abracadabra","r")         # Split wherever "r" appears
strsplit("abcd",split="")           # Split into separate characters
paste("ab","c","d",sep="")          # Join together
paste(c("a","b","c"),collapse="")   # Join vector elements
```

and try these commands to learn a bit about date formats:

```
dd=as.Date(c("2003-08-24","2003-11-23","2004-02-22","2004-05-03"))
diff(dd)
as.Date("1/1/1960",format="%d/%m/%Y")
as.Date("1:12:1960",format="%d:%m:%Y")
as.Date("1960-12-1")-as.Date("1960-1-1")
as.Date("31/12/1960","%d/%m/%Y")
as.integer(as.Date("1/1/1970","%d/%m/%Y"))
as.integer(as.Date("1/1/2000","%d/%m/%Y"))
dec1=as.Date("2004-12-1")
format(dec1,format="%b %d %Y")
format(dec1,format="%a %b %d %Y")
strptime("2005-02-07 14:00","%Y-%m-%d %H:%M")   #use strptime for Date+time

times=strptime("07/02/2005 14:00","%d/%m/%Y %H:%M",tz="America/Lima")+1800*(0:16)
plot(x=times,y=runif(length(times),min=8,max=12),bty="l")

#...and finally the following should give you 864:
library(hydroTSM)
dip(from="2007-05-01",to="2009-09-10",out.type="nmbr")
```

(the tz option here allows you to specify different time zones, using the standard Linux/UNIX naming conventions: http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/ReferenceGuide/timezone.html)

Plotting Data on a Graph

R has a lot of clever plotting functions and these are one of the main reasons for learning how to use the thing SO here are a couple of examples. Try typing the following into the Console window to get a basic plot:

```
years=c(2004,2005,2006,2007,2008)
rainfall=c(1500,1300,1800,1350,1950)
plot(x=years,y=rainfall)
```

Now that's fine, but the plot comes out using the R defaults which (oddly) don't correspond with the standard guidelines on figures that most scientific journals insist on (e.g. see "Figures" on the Journal of Ecology page http://www.blackwellpublishing.com/submit.asp?ref=0022-0477 which starts **"**Figures should not be boxed (superfluous bounding axes)..."; have you ever heard of "chartjunk" http://en.wikipedia.org/wiki/Chartjunk ?). If you look at the manual pages ?plot and ?par you'll find out how to change the formatting options (you'll almost always need at least bty="l" and las=1) and here is an example of a different way of displaying the same data:

```
thisdata=data.frame(years,rainfall) #Try fix(thisdata) for a nice way to see the
contents of a data frame
bestfit=lm(rainfall~years,data=thisdata)
dev.new();plot(thisdata,main="Annual Rainfall",xlab="recorded
years",ylab="mm",bty="l",las=1,ylim=c(0,2000),pch=4,sub=paste("Best Fit line is y =
(",bestfit$coefficients[2],") x + (",bestfit$coefficients[1],")"))
lines(thisdata)
abline(bestfit,lty=2)
```

and here's a plot with data labels:

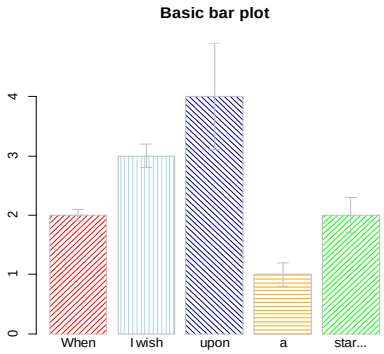
```
sp=c("Alchornea costaricensis","Alseis blackiana","Annona spraguei","Apeiba
aspera","Cordia alliadora","Sapium caudatum")
seedmass.mg=c(38.5,0.12,40.4,14.2,2.9,64)
leaflifetime.mo=c(5.3,10.2,3.9,5.9,8.5,5.3)
posns=c(4,3,2,1,1,1)     #1=below, 2=left, 3=above, 4=right
dev.new();plot(x=leaflifetime.mo,y=seedmass.mg,xlim=c(0,12),xlab="Leaf lifetime in
months",ylab="Seed air-dry mass in mg",bty="l",las=1,log="y",type="n")
points(x=leaflifetime.mo,y=seedmass.mg,pch=4)
text(x=leaflifetime.mo,y=seedmass.mg,labels=sp,pos=posns)    #adding ",offset=-0.305"
and removing the points command would mean the data label is actually on the correct
point
```

Plot looks messy? Try the following to zoom in on a part of the plot (although this does require installation of the TeachingDemos package - see below):

```
library(TeachingDemos)
zoomplot(locator(2)) #Now click on 2 points in the plot to zoom in
```

Keep going!...

```
plot(0:3,0:3,type="n")
polygon(c(1,1.5,2),c(1,2,1),col="red",border="black")
curve(dnorm,from=-3.25,to=3.25,yaxs="i",ylim=c(0,1.025*dnorm(0)),bty="n")
#Fill each 2.5% tail in grey:
x=seq(from=1.96,to=3.25,length.out=20)
polygon(c(x[1],x,3.25),c(0,dnorm(x),0),col="grey")
polygon(-c(x[1],x,3.25),c(0,dnorm(x),0),col="grey")
```



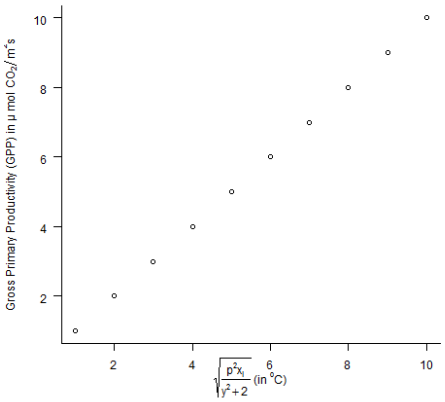
And some error bars (there are many other ways to do this, e.g. see http://myowelt.blogspot.com/2008/03/beautiful-error-bars-in-r.html or the `plotCI` command in package plotrix):

```
mr=c(2,3,4,1,2) #Example measurement values
sr=c(0.1,0.2,0.9,0.2,0.3) #Half height of the error bars
(=1SD, =1SE or =1CI)
barxpos=barplot(mr,main="Basic bar
plot",ylim=c(0,max(mr+sr)),col=c("red","lightblue","darkblue
","orange","green"),border="grey",angle=(1:5)*45,density=20)
barwidth=0.1
segments(barxpos,mr-sr,barxpos,mr+sr,col="grey")
segments(barxpos-barwidth,mr+sr,barxpos+barwidth,mr+sr,col="grey")
segments(barxpos-barwidth,mr-sr,barxpos+barwidth,mr-sr,col="grey")
mtext(side=1,at=barxpos,text=c("When","I wish","upon","a","star..."))
```

Some bright spark at the R Office decided to write a nice graphics demonstration which you can look at too:

```
demo(graphics)
```

(keep pressing ENTER or RETURN to go through it - the commands also appear so you can work out how those plots were done too). See http://zoonek2.free.fr/UNIX/48_R/04.html and http://cran.r-project.org/doc/Rnews/Rnews_2003-2.pdf for a lot of basic examples.



For those worried about how to put maths on the axis labels, see `demo(plotmath)` or my example here:

```
plot(x=1:10,y=1:10,xlab=expression(paste(sqrt(over(p^2*x[i],y**2+2)),"                (in
",{}^o*C,")")),ylab=expression("Gross   Primary   Productivity   (GPP)   in   µ"*"   mol
"*CO[2]/m^2*s),las=1,bty="l")
```

R can do LOTS of more advanced plots including trellis/lattice plots (package "lattice": see next section for what packages are, http://www.stat.auckland.ac.nz/~paul/RGraphics/chapter4.pdf), 3D graphics (package "rgl", http://rgl.neoscientists.org/gallery.shtml), dynamic visualisation (package "rggobi" to link with GGobi, http://www.ggobi.org/), etc. (http://www.math.yorku.ca/SCS/Gallery/ is worth a look) but these are far too much for now.

Packages

For a lot of statisical analyses you have to get to know R's system of 'packages'. Type "library()" into a Console window to find out what packages were installed on your computer when you put in R. Now type "search()" or "sessionInfo()" to get a list of the packages from that list that are already attached/loaded in (installed packages are not necessarily loaded in when you open R).

A package like "methods" is already installed and loaded in by default, but "survival" is only installed so if you want to use it you have to load it in by typing "library(survival)" or "require(survival)". Try the command "mean(rats$time)" both before and after loading in survival (which should give you 89.4) to see what I mean.

If you want to use a package that isn't already installed, then, hmmmm .... To be honest, if you're reading a beginners' course on R (which you are) then you should have no cause whatsoever to be using uninstalled packages .... OK, if you insist, then type "install.packages("nlme")" into the Console window, choose a mirror site geographically nearby and see what happens (you may have to close R and reopen it as an administrator (right-click on the R icon for this option) in order to be able to install packages). For a list of (the many) available packages, go to http://www.R-project.org, click CRAN on the left, choose a mirror site near you and then click Packages on the left (you can find "nlme" in that list to see what you've just installed). Every package has a PDF manual uploaded onto the R website at this location and some also have example files to show what the package does (e.g. check out the 'world map' package at http://cran.r-project.org/web/packages/rworldmap/index.html).

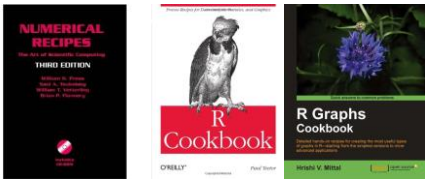
The packages list on the R website is also how you can find out whether there's a package that can accomplish a particular task. For example, let's say you have to deal with netCDF files (.nc). There is an R package that gives you commands for reading and writing to netCDF format, which you can find on that list page, so can you download the manual and install it? (this isn't quite a trivial question because the package isn't called "netcdf" so install.packages("netcdf") won't do anything).

Quadrat-o-phenia

This is only supposed to be a beginners' course and I think it's long enough already so I'm going to wrap it up here by giving you a final example program that demonstrates some R features (vector arithmetic, reading in data from a textfile using the read.table() command, a χ^2 test, a bar plot, a function definition and sampling from a Poisson distribution). Have a look at quadrats.r and try to figure out how it works using the manual resources I've been telling you about above (the program explains what it's working out as it goes along). Apart from running this program and pausing in admiration at the clear and concise way I write code (or perhaps staring in disbelief at how much pain can be compressed into a single page of text ...), I'd like you to do something as a final challenge.

Q6 Dig out a *different* worked example of a χ^2 test (you may have encountered one before in a textbook or you could look at http://www.mste.uiuc.edu/patel/chisquare/intro.html). They should all follow exactly the same calculations and I'd like you to modify the program quadrats.r so that it calculates whatever problem you've dug out and gets *the same answer* as your book or website. A big, fat 30 stars to you if you can do that!

Whilst trawling through quadrats.r you should have found examples of many unfamiliar commands. This is deliberate and you may have noticed that I've been trying to slip into all these programs as many different useful commands as I could. This is because I want you to keep the programs and use them. You see, no-one ever really writes an R script straight off from scratch: most programmers maintain a little 'library of useful programs' somewhere and when they have to write a new one they start with an old program in this library and modify it until it can do the job in hand. There's nothing bad in this (C and FORTRAN pogrammers even publish many standard routines - see "Numerical Recipes", http://www.nr.com and the "R Cookbook" and "R Graphs Cookbook" are similar ideas for R) and the idea of giving you these scripts is that they will form the kernel of a similar library for you.



All Done and Dusted

That's it! The end of the course. You're now no longer a beginner at R (!). Where to go from here? For people past beginner level, I'm afraid there's not really many substitutes to "learning by doing", which is *very* tedious and boring! You can buy a textbook if you like, but check out the R website http://www.R-project.org before you do that: click on Documentation/Other on the left and then "contributed documentation" in the main part of the screen. There are lots of freely-downloadable resources there, e.g. I liked Tom Short's reference card at http://cran.r-project.org/doc/contrib/Short-refcard.pdf. Also, don't ignore the tutorials that are on the R website e.g. you should try the "sample session" on p.78 of http://cran.r-project.org/doc/manuals/R-intro.pdf (only 3 pages) and have a glance at the rest of that introduction too.

There are many R courses online nowadays, e.g. have a look at `help.start()` (mentioned above) or CSIRO's course (http://www.csiro.au/resources/Rcoursenotes.html) or Ruth Ripley's course (http://portal.stats.ox.ac.uk/userdata/ruth/APTS2013/APTS.html), or "Uso da Linguagem R para Análises de dados Ecológicos" (somente disponível em português, http://ecologia.ib.usp.br/bie5782/doku.php) or you could check out http://www.r4all.org/the-courses/ (though that's not free). You can use the web to find more specific help, e.g. if you search "tutorial R functions" on Google (http://www.google.com) then you will find several pages which explain functions fairly well. Also *use the web if you get stuck*: if you are struggling to work out how to do something, then 99% of the time someone has encountered the same problem before and obligingly put a solution on a website. For example, having trouble rotating the axis labels on an R plot? It took me only a few seconds to find https://stat.ethz.ch/pipermail/r-help/2000-December/009507.html on Google. I'd like to put in here a very big THANK YOU to all the people who put solutions on websites like these: you have saved me MANY times.

If you've got this far and you feel that R is not very user-friendly and you're unhappy about that, then there are some people who have put a menu-driven 'front end' on to the interface: have a look at R Studio https://www.rstudio.com/, R Commander http://socserv.mcmaster.ca/jfox/Misc/Rcmdr, Tinn-R http://www.sciviews.org/Tinn-R, JaguaR http://rforge.net/JGR/, Brodgar http://www.brodgar.com and (Linux only) RKWard http://rkward.sourceforge.net. However, I should point out that none of these softwares *extend* R's capabilities in any way: they just offer a more 'point-and-click' interface for using it. If you're happy with the R Console window and manual pages and entering commands in directly (as I am) then you don't need to look at any of them.

For further information, the R website has links to a lot more (e.g. see http://cran.r-project.org/doc/Rnews/Rnews_2001-1.pdf for some history about R) and a mailing list for help (*don't* email this list with general questions like "how do I do XXX?": be *specific* e.g. "I'm trying to do XXX. I've tried AAA and BBB (example code ...) but they don't work ...").

Almost done: just have to explain about the stars. For each star you earned during this course (max. 70), please now go out and buy a seed or a plant and put it in your garden or somewhere similar (native species only, of course). In this way the study of R can contribute to the greening and beautifying of the world around us.

All the best!

Toby Marthews.

PS. Most R users (and most R manuals) say you should type "x <- 3" instead of "x=3" to make x equal 3. There are reasons for this, but I find "x=3" much more straight-forward (and easier to explain to my boss when he looks at my code!) and they are both equivalent.

PPS. Watch out for my 4 most common errors when writing R code:

i) Thinking that if something like mean(5.4,6.7,2.3) gives me 5.4 then that's the mean, when actually it's 4.8 (from mean(c(5.4,6.7,2.3)))

ii) Forgetting that "log(x)" means ln(x) not log10(x),

iii) Forgetting to use "==" in if statements (e.g. you have to type "if (x==3) { ..." instead of "if (x=3) { ...") and

iv) Comparing with a negative number (e.g. "if (x<-3) { ..." doesn't work: you have to do "if (x<(-3)) { ...").

PPPS. Almost forgot: the answer to Q4 is: "ecnatsid ruobhgien tseraen" rof sdnats "DNN".

PPPPS. Yes, of course you have to buy the plants! We have ways of watching you ... OK, at least buy one potted plant and just love it ...?