Really simple intro to R

Dan Lawson 31/10/08

Getting Started

We follow http://cran.r-project.org/doc/manuals/R-intro.pdf which is a complete introductory guide to R. The labels below refer to the sections from there where more details are available. We just consider here the bits of use for us today. TYPE into R the text after the ">" sign on a line to execute it – don't copy and paste as you won't learn it and it might not work!

1.7 help files. Help is available on all functions and is mostly very useful. To find help on anova:

```
> help("anova")
or
> ?anova
```

Searching help files is done by

```
> help.search("anova")
or
> ??anova
```

Note: the # symbol means a comment in R and everything on the same line after it is ignored.

2.1 Vectors and assignment. Everything in R is stored as a vector or more complex data structure. It operates on each element seperately, except where special functions exist.

```
> myvec <- c(1,5,20)              # Create a vector. The "c"oncatonate operator is very
important!
> myvec                           # Look at it
> myvec2 <- myvec*myvec           # square it
> myvec2
> 2 * myvec + - pi^2              # do arithmatic on it
```

Vectors may contain anything: logical values, strings and complex data objects. e.g.

```
> c("who","likes","cheese?")
```

2.3 Sequences. Functions exist to create arbitrarily complex sequences. Simple sequences are produced by : e.g. 1:5 is c(1,2,3,4,5). More complex sequences can be created by seq(), rep() and others:

```
> seq(-5,5,by=0.2)
> seq(-5,5,length.out=21)
> rep(c(0,1),each=5)
> rep(c(0,1),times=5)
```

2.4 Logical vectors. If a condition ("=", "<",">","<=",etc) is applied to a vector, a logical vector for whether each element obeys the condition is returned. These can be further manipulated with the normal logical operators "&" (and) ,"|" (or) ,"!" (not) ,etc.

```
> x<-1:10
> x > 5
> (x > 5 ) & (x<=8)
> (x > 5 ) | (x<=2)
```

2.7 Selecting vector elements. Elements of a vector are selected by the "[]" operator. Logical vectors can be used to select elements of a vector:

```
> x[(x > 5 ) | (x<=2)]
```
As can index numbers:
```
> x[1:4]
```
Negative index numbers to remove certain elements:
```
x[-c(,4,5)]
```
And also the names of the vectors, if you assign them:
```
> numcars <- c(1,0,3)
> names(numcars) <- c("bob","dan","rich")
> numcars["dan"]
```

2.8 Other data types. Arbitrarily data types can be defined, but the most common are matrices, lists and data frames.

Section 5 deals with matrices, which are two dimensional vectors. They can also contain any type of data, and are created by calling "matrix":
```
>M1<- matrix(0,4,3)
> M2<-matrix(1:12,4,3)
> M3<-matrix(1:12,4,3,byrow=TRUE)
> M1
> M2
> M3
> M2 * M3
```
Note that the multiplication is element wise. "t()" transposes a matrix and "crossprod()" or "%*%" do matrix mupltipication:
```
> M2 %*% t(M3)
```

9: Programming elements. Loops and conditions are essential to do any complex calculations. This chapter covers them in detail – here we just get an overview. "if" is essential and works like this:
```
if(<condition>) {<do something>
}else if(<condition2>) {<do something else>
}else <do something else again>
```
The "{" and "}" brackets are to control what gets executed where, and aren't needed unless its ambiguous. The else if and else are obviously not needed either. ";" can be used to separate statements.
```
> a<-4; b<-5
> if( a>b) print("a is the biggest!") else print("no, b is!")
```

Loops are also extremely useful, although discouraged in R with preference for function based approaches "apply", "sapply" and others – briefly discussed below. Still, you can do things more easily with loops - "for" and "while" are the most common:
```
> for(i in 1:10) print(i)
> n<-0
> while(n<10) {print(n);n<-n+2}
```

10. Functions. Writing your own functions is a very powerful feature of R. They are defined as

```
function_name <- function(<arguments>) { <function code>)
```
for example:
```
> add_two <-function(x){return(x+2)}
> add_two(1:10)
```
The value returned is given in the "return" function, or it is the last result obtained in the function:
```
> my_poly <- function(x){x^2 + x}
> my_poly(0:10)
```

10.3. Named arguments and defaults is worth a read at some point.

Session Management

Sessions can be saved by quitting, asking R to save a session, and loading again:
```
> chips                    # this gives an error since we have no chips :(
> chips <- TRUE
> q()                      # Select "y" when asked to save the state
```
Load R again, it will autoload the state
```
> chips                    # returns TRUE
> ls()                     # lists all the objects from before.
```

Specific objects can be saved using "save" and loaded with "load". An example:
```
> save(chips,file="myRsave.RData")        # saves the object "chips"
> rm(list = ls())          # removes ALL objects!
> chips                    # Error again
> load(file="myRsave.RData")
> chips                    # true again now.
```

Plotting

Plotting is really easy to get started with, though difficult to understand fully because the details can be complex. To plot "scatterplot" or line style, "plot(x,y)" is enough. Most objects that you'd want to plot have their own special version of plot. Using the in built vector dataset "islands", a histogram is plotted by:
```
> plot(hist(islands))
```
(See what this plotted by:
```
> islands
```
)

There are many graphical options to make pretty graphs. see "?par" and "?plot" for full details. Some examples are:
```
> x <- seq(0,10,by=0.1)
> plot(x,exp(x),xlab="X label", ylab="Y label", main="Main title",type="l") # Line graph
> plot(x,exp(x),xlab="X label", ylab="Y label",log="y") # logarithmic y axis
```

Other useful things

Random numbers and distributions
"
There is way too much here for now, but everything you might want is available. See 8. Probability Distributions in the reference. The problem sheets take you through "sample" and in Sheet 4 you use distributions. As a start, the uniform distribution is *unif, the normal is *norm, and the binomial is *binom, where * is r for random numbers, p for the probability distribution, etc: see for example
```
> ?runif.
> runif(20,0,5)          # Uniform distribution between 0 and 5
> rbinom(20,10,0.1)  # binomial distribution, size parameter 10 and probability 1
> x<-1:10
> plot(x,dbinom(x,10,0.1))
```

Other sections particularly worth looking over are 1.10 Commands in a script, 7. Reading data.