

MA40198: Some R basics

Manipulating data

- `<-` The assignment operator. Most important thing to know in R apart from `q()`!
- `x <- a:b` provided that `a` and `b` are integers, this produces an ordered sequence of all the integers from `a` to `b`.
- `c(a,b,c,...)` makes an array out of `a`, `b` `c` etc (which may themselves be arrays).
- `x[i]` gives element `i` of array `x`.
- `x[3:6]` gives an array containing elements 3,4,5 and 6 of `x`.
- `x[c(1,3,5,3)]` gives an array containing elements 1,3,5 and 3 again of `x`.
- `x[x<a]` returns an array of all elements of `x` less than `a`.
- `array(x,n)` produces an array of length `n`, containing whatever is in `x` repeated enough times to fill the array.
- `attach(d)` allows access to the arrays of data frame `d` without the need to prefix the array names with `d$`.
- `data.frames` are named lists of arrays. If a data frame `d` contains arrays `x` and `y` then these can be referred to using the `$` symbol as `d$x` and `d$y`.
- `for (i in 1:n) {}` repeats whatever is in `{}` for each integer value of `i` from 1 to `n`.
- `length(x)` returns the length of array `x`.
- `names(a)` returns the names of elements of object `a`.
- `objects()` lists all objects that you have created.
- `rm(a,b,c,...)` deletes (removes) objects `a`, `b`, `c` etc.
- `rm(list=objects())` removes everything created so far this session.
- `sample(n,x)` sample `n` elements at random from array `x`: returns a length `n` array containing the random sample.
- R arithmetic is vector oriented! Typically operators and function work element-wise. Try `a <- 1:3`; `b <- 5:6`; `d <- a*b^a`, for example.

Some basic statistics functions

- `acf(x,lag.max=20)` find the ACF of the time series data in array `x` up to the lag given in the optional parameter `lag.max` (20 in this example).
- `cor(x,y)` finds the correlation (r^2) between data in arrays `x` and `y`.
- `cov(x,y)` finds the covariance between data in arrays `x` and `y`.
- `max(x)` find the maximum value in array `x`
- `mean(x)` finds the mean of the data in `x`.
- `min(x)` find the maximum element in `x`
- `rnorm(n,m,sd)` returns `n` observations of a normal random variable with mean `m` and standard deviation (square root of variance) `sd`.
- `runif(n,a,b)` returns `n` observations of a $U(a, b)$ r.v.
- `summary(x)` provide summary information on object `x` (not just arrays).
- `var(x)` finds the variance of the data in `x`.

Libraries

- `data(name)` load the data frame called `name` supplied with R or a package. If you leave out `name` you get a list of data frames that are available.
- `library(mgcv)` load package `mgcv` which adds some extra modelling functions to R (see `?gam`)

Plotting

- `boxplot(x,y,z,...)` produces boxplots for the data in arrays `x`, `y`, `z` etc. you can have any number of arrays from 1 upwards.
- `hist(x)` plots a histogram for the data in array `x`. Labelling and colour options as for `plot()`
- `lines(x,y)` plot array `y` against array `x` on the existing plot, joining the points with lines.
- `par(mfrow=c(3,2))` split graphics window into 3 rows and 2 columns for plotting purposes (can substitute any sensible numbers for 3 and 2).
- `plot(x,y)` plots array `y` against array `x`. Additional useful options are: `type="l"` to join the plotted points with lines; `col="red"` to set the plotting colour to a colour (red in this case!); `main="some title for plot"` to give the plot a title; `xlab="some x-axis label"` to label the x axis; `ylab="whatever"` to label the y axis.
- `points(x,y)` plot array `y` against array `x` on the existing plot.

Functions

- Functions are R objects which take a list of arguments, and do something with them in order to produce a result. `sin`, `plot`, `rnorm` etc. are all examples of built in R functions.
- Typing the name of a function without the usual brackets `()` causes its defining code to be printed.
- You can write your own functions. e.g. The following produces a function called `nsim` which simulates `n` random normal deviates, plots a histogram and returns a list with the sample mean and variance.

```
nsim <- function(n,mu=0,sd=1) { ## notice some arguments can have default values
## function body
  x <- rnorm(n,mean=mu,sd=sd) ## simulate data
  hist(x) ## plot histogram
  list(mean = mean(x),stdev=var(x)^.5) ## last object is returned object
}
```

Linear models

- `lm()` function for fitting linear models to data by least squares. Usage best seen by example. Let `y` contain data y_i , `x` contain data x_i and `z` contain data z_i , and let e_i be zero mean normal random variables:
 1. `mod<-lm(y~x)` fits model $y_i = a + bx_i + e_i$ by least squares and returns fitted model object called `mod`.
 2. `mod1<-lm(y~x+z)` fits model $y_i = a + bx_i + cz_i + e_i$ by least squares and returns fitted model object called `mod1`.
 3. `mod2<-lm(y~x+z-1)` fits model $y_i = bx_i + cz_i + e_i$ by least squares and returns fitted model object called `mod2`.
- `coef(mod)` gives an array containing the model parameter estimates for the model fit stored in `mod` (which will have been returned by `lm()`)
- `summary(mod)$sigma` extracts the variance of the estimated e_i terms (see `lm()`) from a fitted model object `mod`, which will have been produced by a call to `lm()`.
- Other standard statistical models behave similarly.