

R In One Page

Barry Rowlingson

R is an interpreted language for statistics, programming and graphics. The home page, with lots of useful links to more tutorials and books, is <http://www.r-project.org/>.

Start R by opening R from the Windows desktop or typing R at a Unix command prompt.

Objects

R stores information in named 'objects'. Assignment is with the '=' sign or the '<-' arrow. The simplest object just stores one value:

```
> x = 1.234
```

Functions

R operates on objects with functions. Functions take their arguments in parentheses. You can type function calls at the R prompt and R will print the answer:

```
> sqrt(x)
[1] 1.110856
```

Vectors

The [1] above shows us this is the first value in the object. R objects can be vectors with more than one element. One way of making a vector is with the c() function:

```
> c(1,3,6)
[1] 1 3 6
```

R also has the ":" operator, which generates sequences:

```
> 1:50
[1] 1 2 3 4 5 6 7 8 9 10 11 12
[13] 13 14 15 16 17 18 19 20 21 22 23 24
[25] 25 26 27 28 29 30 31 32 33 34 35 36
[37] 37 38 39 40 41 42 43 44 45 46 47 48
[49] 49 50
```

Now the numbers in brackets are labelling the first element on each line.

Vector Functions

R functions mostly operate on whole vectors. So now we can compute the first 100 numbers and their square-roots, and plot them on a graph. This example also shows how named options can be passed to functions – try this example without the "type='l'" option:

```
> x = 1:100
> y = sqrt(x)
> plot(x,y, type='l')
```

Help

You can get help on R functions by typing, for example, "help(sqrt)", or just "?sqrt". There is also "help.search" for topic-based searches (e.g. "help.search('t-test)'). That will give you a list of functions.

Don't Panic!

If you give R an incomplete line, such as missing a closing bracket, then it will give a "+" prompt and you can continue the input line. If you can't see where you've gone wrong, hit the Stop button (Windows) or Ctrl-C (Unix) to get a fresh R prompt back. You can also use this to interrupt a running R function. It's better than killing the process, since you get your prompt back and have a chance to save your work.

Further Examples

Overleaf is a set of R command lines you can run. Feel free to experiment.

```

x = 1:10                # simple sequence
x = seq(1,100,2)       # more flexible sequence generator
x[3:5]                 # slices from a vector
y = sqrt(x)           # function call
plot(x,y)              # plot points
plot(x,y,type='l')    # plot lines

m = matrix(1:12,ncol=3,nrow=4) # a 2d matrix
m[1,]                  # a single row
m[,2]                  # a single column
m[,1:2]                # two columns
image(m)                # matrix display

r1 = rnorm(100, mean=1) # random numbers – normal distribution
hist(r1)                # histogram
r2 = rnorm(100, mean=1.2) # more random numbers, different mean
hist(r2)                # show
mean(r1) ; mean(r2)    # separate functions with semi-colon
t.test(r1, r2)         # does confidence interval include zero?
t.test(r1, r2+2)      # shift one dataset and try again...

x = 1:10                # explanatory variable/independent variable
y = 1.2 + 3*x + rnorm(10, mean=0, sd=2) # construct a measurement/response variable
plot(x,y)                # show
lm(y~x)                  # model y depends on x – check intercept/slope
fit = lm(y~x)            # save it
summary(fit)             # R-squareds, F-statistics and all that
plot(fit)                # some diagnostic plots
names(fit)                # list components of the fit object
fit$fitted.values        # access them with dollar
plot(residuals(fit), fitted(fit)) # access components with functions and plot

quad = function(x,a,b,c){return(a*x*x+b*x+c)} # define a function ax^2+bx+c
x = seq(-3,3,len=100)    # x-axis
plot(x,quad(x,1,-2,-1),type='l') # plot it
abline(h=0)              # add horizontal zero line

squad = function(a,b,c){ d = sqrt(b^2 - 4*a*c);
  roots = (-b + c*(-1,1)*d)/(2*a) ; return(roots) } # solve quadratics
squad(1,-2,-1)           # solve for y=0
abline(v=squad(1,-2,-1),lty=2) # add dotted vertical lines

```