

# An Introduction to R

## To get R:

1. Download R (it's free) from the website <http://cran.r-project.org> There are versions for Linux, Windows and Mac.
2. Tutorials for R at <http://cran.r-project.org/doc/manuals>

## R can be used as a calculator:

Try typing the following expressions at the command line (followed by return): (> is the command prompt).

```
> 5+3
> 10*10
> log(9.4)
> exp(exp(exp(20))) # R is only human! (anything following '#' is a comment)
```

R has most any mathematical function you can think of such as `sqrt()`, `sin()` ... mostly with easily guessable names. Expressions using the logical operators `==`, `!=`, `<`, `>` give Boolean values (T,F)

```
> 4 > 3 # this evaluates to T (true)
> 1 == exp(0) # so does this
> 1 != exp(0) # this evaluates to F (false)
```

It is possible to have variables that hold values in your program. Most strings beginning with an alphabet character will be treated as variables. Try typing the following lines in succession

```
> x = 3 # set x to 3
> y = x*x+x
> y # print the value of y
```

## Vectors

One of the nicest aspects of R is the way it handles vectors. Here are a several ways to create vectors:

```
> x = 1:100 # x is now the vector (1,2,...,100)
> y = seq(-pi,pi,length=100) # y consists of 100 evenly spaced values from -pi to pi
> y = seq(0,10,by=.03) # 0, .03, .06 ....
> z = c(1,4,8,20) # z is the vector (1,4,8,20)
> a = x+y # vectors of same length can be added, multiplied, etc.
> b = 4*x # this is interpreted correctly too
```

## Random Number Generation

R has lots of built-in functions for doing things with random numbers. For instance

```
> x = runif(100) # creates a vector of 100 (uniformly distributed) random numbers between 0 and 1.
> punif(v) # is the probability that a Unif(0,1) rand number is less than v
> qunif(u) # gives the uth quantile of a Unif(0,1). More on this later.
```

There are similar functions for a variety of other distributions including the normal(0,1) (`rnorm`,`pnorm`,`qnorm`) Cauchy (`rcauchy`, `pcauchy`, `qcauchy`), Exponential, Binomial, Poisson, and others.

## Subsets

```
> x = runif(100) # creates a vector of 100 Unif(0,1) random numbers
> x[1] # the first element of x
> x[c(1,3,5)] # a vector containing 1st, 3rd and 5th elements of x
> y = x > .5 # a 100-long vector of Boolean values y[i] is T iff x[i] > .5
> z = x[x>.5] # the 'x's' that are greater than 5
```

**Plotting** Try the following

```
> x = seq(0,1,length=100)
> y = x^2                # y = x squared
> plot(x,y)              # plot with (x[1],y[1]) \ldots, (x[100],y[100])
> plot(y,x)
> plot(x,y,pch=5,col=3)  # use plot character 5 and color 3
> plot(y)                # same as plot(1:length(y),y)
> plot(runif(100),pch=1:100,col=1:100)  # alphabet soup
```

**Source Files** You will want to write simple programs in R and this always requires some trial, error and iteration. I recommend the following procedure: Create a “source” file in any text editor containing your R commands. This could be emacs or the Windows “Notepad” or whatever you are comfortable using. Suppose you create the following file named “myprog.r” in your editor:

```
len = 100
x = runif(len,-.5,.4)
y = cumsum(x)  # y[1] = x[1], y[2] = x[1]+x[2], etc.
plot(exp(y))
title("my stock price")
print("history is: ")
print(y)
```

This technique allows you to write a program in the usual incremental way. If you want to get a hard copy of the printout and the plot (for example, to submit as your homework), do the following

```
> postscript("myplot.ps")  # write plot in the postscript file ‘myplot.ps’
> sink("myout.txt")        # write text output to ‘myout.txt’
> source("myprog.R")       # run the program you created
> dev.off()                # redirect plots to screen. Don’t forget this!
> sink()                   # redirect output to screen. ditto.
```

**Quitting and help**

```
> help("rnorm") # gives information about the function rnorm. Of course this works
>               # for other functions too.
> q()           # quitting the program. Hope you had fun.
```