CSMMA16

Introduction to Matrices and Vectors in R

Aim

The aim of this session is to introduce (or remind you about) R, and to show how R can be used to perform various operations on vectors and matrices. Refer to the lecture handout on matrices and vectors. We will also briefly introduce constructing graphs.

Introduction

R is a free and open source statistical package that has gained substantially in popularity in recent years. It is also useful for scientific and engineering calculations. R is available in most PC labs in the University. Several graphical interfaces are freely available. We will use the R console for windows.

This session shows how functions can be used and constructed in R. It reinforces some concepts given in the lectures. R provides various functions for processing matrices. Click on the **R** icon to start the program. When R is invoked the prompt shown is:

```
>
```

The user can then enter any command, which R will obey if it is valid. Most R functions have online help documentation. For example, to get help with syntax for standard trigonometric functions, type

```
> help(Trig)   # or ?Trig
```

In the practicals you will investigate some of the many aspects of R. Note that, rather than typing in a command you have already used, use the “up arrow” on the keypad to recall previous commands. If you prefer (recommended), you can type commands using the R editor, accessed via the “pull-down” menu (File → New script). Commands in the script file are run by first “high-lighting” and “right click” on the mouse or via the Edit pull-down menu.

Initial Investigations

The first stage is to investigate some basic manipulation of numbers, using simple operators and functions. In the following, anything following a “>” should be typed.

```
> 5+4*3
```

You should see

```
[1] 17
```

Next investigate some functions, the absolute and sin functions:

```
> abs(-67)
```

```
> sin(45)
```

Are the answers you expect? Possibly not, as R expects the angle passed to sin to be in radians, not degrees. Type the following to find the sine of 45^{O}, using the built in constant pi:

```
> sin(45*pi/180)
```

Type the following, which declares a variable which is used in the subsequent command

```
> x = sin(45*pi/180)
```

```
> asin(x)
```

R has few restrictions for variable names. Valid names consists of letters, numbers and the dot (.) or underline (_) characters and starts with a letter or the dot not followed by a number. So for example, `s.1` and `.s1` are valid names but `.1` and `_s1` are not.

R is case sensitive.

Note also that R has three different assignment operators (=, `<-` and `<<-`). The operator `<-` can be used anywhere but there are restrictions on use of `=`. The `<<-` operator is used for assignment in the global environment. For further details, type

```
> ?assignOps
```

Vectors

In R, a vector is a set of elements of the same type. Vectors can be constructed using the concatenate `c`, `seq` and `rep` functions. Try the following:

```
v1<-c(1,5,0,2); v1

v2<-seq(1:4); v2

v3<-rep(v1,2); v3

v4<-c(TRUE, FALSE, FALSE, TRUE); v4  # vector of logicals

v5<-c("aa","bb","cc"); v5      # character strings
```

The number of elements in a vector is found using the `length` function.

```
> length(v4)

[1] 4
```

You can check whether a variable is a vector using `is.vector` function. A variable can be coerced to a vector using `as.vector`.

An element in a vector is specified by its position. For example

```
> v1[2]   # 2nd element in v1
```

Can you guess the output for the following?

```
> v1[v4];  v3[-1];    v3[-2:-4]
```

Matrices

A matrix is a two dimensional rectangular array of elements. Matrices may be constructed using the `matrix` command. Vectors can be coerced to matrices using `cbind` and `rbind` functions.

Type the following commands to understand what each does.

```
m1<-matrix(v1); m1
```

```
m2<-matrix(v2,nrow=2,byrow=T); m2
```

```
m3<-cbind(v1,v2); m3
```

```
m4<-rbind(v2,v4); m4
```

You can check whether a variable is a matrix using `is.matrix` and it can be coerced to a matrix using `as.matrix`.

An element in a matrix is specified by its row and column: A(r,c) is element in row r and column c of matrix A. The command in R to find the element in row 2 and column 1 of matrix m2 is

```
> m1[2,1]
```

You can get all elements in row 2 by

```
> m2[2,]
```

You can also access elements in a matrix (or vector) that meets a criterion. For example, m3[m3<=2] returns a vector of elements in m3 with value less than or equal to 2. Guess what the following does and then try it.

```
> m3[m3<=2]<-0
```

Construct a vector `x` of length 200, with first element equal to 0 and last element 360. Use this vector to generate the vector `y` with elements the sine angles between 0 and 360 degrees and save both vectors as columns of a matrix.

Plotting Graphs

R has various commands for plotting graphs. An important command used to set parameters of the graph is `par`. For a list of graphical parameters, type

```
> ?par
```

and for list of current values of `par`,

```
> par()
```

To plot a graph of the elements in `x` (in the previous section) against those in `y`:

```
> plot(x,y, type="l")
```

We can change the axis labels using the `axis` command and add mathematical symbols using `expression`.

```
> plot(x,y, type="l", xaxt="n",
xlab=expression(paste("Angle ",theta)),
ylab=expression("sin "*theta))
> axis(1,at=c(0,90,180,270,360),
lab=expression(0,pi/2,pi,3*pi/2,2*pi))
```

Suppose you want to add $\alpha_1^2 = 2$ at location (3π/2, ½).

```
> text(270,1/2,substitute(alpha[1]^2=="2"))
```

Note use of "==" to print a single equal sign. You can write complicated formulae using `paste` to join parts of the equation.

Suppose we want to superimpose cos(x) on this graph. We can do,

```
> par(new=T)
> plot(x,cos(x*pi/180),type="l",axes=F,xlab="",ylab="",col="red")
```

The `par(new=T)` command tells R to **not clean** the frame before drawing.

Can also superimpose plots using `lines`

```
> lines(x,cos(x*pi/180)^2,lty=2,col="blue")
```

or using `curve`

```
> curve(sin(x*pi/180)^2,from=0,to=360,lty=3,add=T,col="purple")
```

Use `par` argument `mfrow=c(nr, nc)` to construct nr rows and nc columns of plots.

Construct plots of sin(x-π/4), and tan(x) for x between –π and π on two separate plots in the same figure. Add appropriate titles and labels to your plots. Use `abline` to show the asymptotes.

To restore default graphical parameters, use

```
> dev.off()
```

Matrix Manipulation

Next you will investigate manipulation of matrices.

Construct the 2 X 2 matrix $\boldsymbol{A} = \begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}$ and the R vectors ***b***=(4, -1) and ***b1***=(4, -1, 0). Work out by hand the product ***A*b***.

Type the following command and try to understand how R constructed this product.

```
>  A*b;
```

To help understand the above product you may want to try

```
>  A*b1;
```

Next try the following and compare the output with your hand calculation.

```
>  A%*%as.matrix(b)
```

Calculate by hand the determinant and inverse of $\boldsymbol{A} = \begin{pmatrix} 5 & 3 & 1 \\ 0 & 2 & 1 \\ 3 & 1 & 2 \end{pmatrix}$ and confirm your results using R.

Find the vector $\boldsymbol{x} = (x_1 \quad x_2 \quad x_3)^T$ satisfying $\boldsymbol{Ax} = \boldsymbol{b}$ where $\boldsymbol{b} = (-3 \quad 2 \quad 1)^T$ using Gaussian elimination.

Check your answer using

```
> solve(A,b)
```

What is the rank of **A**? Confirm using

```
> qr(A)$rank
```

Calculate by hand the eigenvalues and eigenvectors of $\boldsymbol{A} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}$ and check your answer in R. Comment on any differences.

Use R to obtain matrices in the singular value decomposition of $\boldsymbol{A}$ and compare these with the matrices in the eigen decomposition. Comment on the results.

Determine if $\boldsymbol{A}$ is positive definite.

Vector Manipulation

Suppose $\boldsymbol{a} = (3 \quad 2 \quad 1)$, $\boldsymbol{b} = (-2 \quad 3 \quad 2)$ and $\boldsymbol{c} = (0 \quad 1 \quad 2)$.

Calculate by hand the magnitude of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, the inner (dot) product of $\boldsymbol{a}$ and $\boldsymbol{b}$ and the angle between the two vectors. Also calculate the vector cross product $\boldsymbol{a} \times \boldsymbol{b}$ and the scalar triple product $\boldsymbol{a} \cdot (\boldsymbol{b} \times \boldsymbol{c})$**.**

Next write an R function named `ang` to return the angle (in degrees) between two arbitrary vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. Type the following. Your code to generate the angle should be within the curly braces.

```
ang<-function(a,b){

}
```

You can write just one line in which case the braces are not needed, but feel free to write a series of shorter lines.

Run your function to check your answer above.

Also, write functions to do the vector cross product and scalar triple product.