

Summary of R Syntax and Commands

1 Syntax

1.1 Operations

`+`, `-`, `*`, `/` addition, subtraction, multiplication, division.
`%%` matrix multiplication.

1.2 Assignments

Each of the following place the contents of `x` into `y`

```
y <- x
x -> y
y = x
```

To place a group of elements into `x` use the concatenate function `c()`, e.g.
`x <- c(-1, 0.33, 104)` puts the elements `-1, 0.33, 104` into `x`.

1.3 Logical Tests

To compare the elements in `x` and `y`, returning TRUE when condition is met or FALSE otherwise:

Equality: `x == y` (Note the ‘double’ equals!)
Greater than or equal to: `x >= y`
Strictly greater than: `x > y`
(similar for less than `<`)

1.4 Brackets

Use square brackets `[]` to access elements in a vector or matrix.
Use round brackets `()` for function arguments (which includes the arguments for ‘if’ statements and ‘for’ loops).
Use curly brackets `{ }` to group statements into blocks, e.g. to execute all statements in a ‘for’ loop or all statements after an ‘if’ command.

1.5 Creating Functions

Functions are created using `function()`, e.g.

```
f <- function(x) x ^ 2
g <- function(x)
{ z <- f(x)
  z + 1
}
```

would produce the functions $f(x) = x^2$ and $g(x) = x^2 + 1$ respectively.

1.6 Accessing Elements of a Vector or Matrix

Suppose we have elements in a vector \mathbf{x} . Then,

$\mathbf{x}[\mathbf{n}]$ returns the n th element.

$\mathbf{x}[-\mathbf{n}]$ returns a *vector* with the n th element removed.

$\mathbf{x}[\mathbf{n}:\mathbf{m}]$ returns a *vector* consisting of elements in the range n to m .

Notes: $\mathbf{n}:\mathbf{m}$ is itself a vector consisting of the integers p , such that $n \leq p \leq m$.

The syntax is logical, so $\mathbf{x}[\mathbf{y}]$ would select elements in \mathbf{x} indexed using a vector \mathbf{y} . For example:

```
y <- c(3, 10, 1)
x [y]
```

would return the 3rd, 10th and 1st element (in that order) of \mathbf{x} .

Now suppose we have elements in a matrix \mathbf{x} . Then,

$\mathbf{x}[\mathbf{i}, \mathbf{j}]$ returns the element in the i th row and j th column.

$\mathbf{x}[\mathbf{i},]$ returns a *vector* which consists of the elements from the i th row.

$\mathbf{x}[, \mathbf{j}]$ returns a *vector* which consists of the elements from the j th column.

$\mathbf{x}[-\mathbf{i},]$ returns a *matrix* with the i th row removed.

$\mathbf{x}[, -\mathbf{j}]$ returns a *matrix* with the j th column removed.

2 Functions

2.1 Vectors and Matrices

`vector(length = n)` creates a vector of length n .

`matrix(x, nr = n, nc = m)` creates an $n \times m$ matrix using the elements in \mathbf{x} . Note: by default the matrix is filled column wise. To fill the matrix row-by-row include `byrow=T` as a parameter.

2.2 Basic Statistical Functions

`mean()`, `var()`, `sd()`, `median()`, `quantile()`

2.3 Plotting Functions

`plot()`, `qqplot()`, `hist()`, `boxplot()`, `abline()`, `points()`

Note `points()` can be used for adding points or a line to an existing plot. `abline(a, b)` adds the straight line 'y = a + bx' to a plot.

2.4 Random Numbers

`runif()`, `rnorm()`, `rbinom`, `rpois()`, `rgamma`, `rweibull`

2.5 Loops

```
for (i in 1:n) { statements }  
while ( condition ) { statements }
```

2.6 Reading and Writing Data

`scan()`, `read.table()` read in unformatted and tabulated data respectively; `write()` writes out data to a file or the terminal.

2.7 Miscellaneous Functions

`apply()` - apply a function to rows/columns in a matrix;

`sort()` - sort elements in a vector or matrix;

`sum()` - sum elements in a vector or matrix. Note if the vector/matrix consists of boolean elements (i.e. each element is either TRUE or FALSE), then 'sum' gives the number of elements with the value TRUE. `ifelse()` - a function for testing conditions on elements of a vector.

`summary()` - can be used on almost any R object.

`source()` - read in R commands from a source file.