

Práctica 1

Ecuaciones Lineales

Clara Simón de Blas (clara.simon@urjc.es)
Universidad Rey Juan Carlos

La descarga del archivo de instalación se realiza desde

<http://cran.es.r-project.org/>

- Comentarios en R: líneas que comiencen con el carácter #.
- Ejecutar comandos: mediante la tecla F5 situándonos en cualquier posición de esa línea (no necesariamente en el final) o la combinación de teclas Control+R para cualquier orden no antecedita de #. Asimismo, si seleccionamos con el ratón más de una línea, éstas pueden ser ejecutadas simultáneamente también con F5 o Control+R. **command + enter**
- Script o guión de trabajo: podemos modificar nuestras líneas de código con comodidad y guardarlas para el futuro. Para ello, utilizaremos la opción Guardar o Guardar como del menú Archivo de la consola.

Tipos de objetos de R

En el lenguaje de R, los elementos u objetos que se vayan definiendo, bien por nosotros mismos, bien como resultado del programa, pueden y deben ser distinguidos para su uso correcto.

Concretamente, vamos a hablar de:

- Vectores.
- Matrices.
- Hojas de datos.
- Variables indexadas (arrays),
- Funciones

Un vector en R puede contener una colección de números o de caracteres no numéricos. Para definir un vector, por ejemplo, el vector $x = (1; 3; 5)$, usaríamos la orden

```
x<-c(1,3,5)
```

Así podremos llamar al vector x en el futuro. Observemos que se utiliza el operador asignación `<-` (también podemos usar el operador `=`) y que es la función de concatenación `c()` la que construye el vector.

Equivalentemente

`c(1,3,5)->x`

Se puede emplear la función `assign` para crear un vector

`assign("x", c(10.4, 5.6, 3.1, 6.4, 21.7))`

Se puede utilizar el valor del vector `x` para crear nuevos vectores

`y <- c(x, 0, x)`

También es posible definir un vector de números consecutivos, por ejemplo, el vector (1; 2; 3; 4; 5) mediante

1:5

De forma más general, la función seq() permite definir secuencias desde un inicio hasta un n con una determinada separación entre ellos. Por ejemplo,

```
y<-seq(-3,3,by=0.5)
```

y

La función `rep()` para definir vectores como repetición de otros vectores. Por ejemplo,

`rep(0,100)`

`rep(1:3,3)`

Si queremos saber la longitud de un vector, usaremos `length()`. Por ejemplo,

`length(y)`

Un vector puede incluir caracteres en lugar de números, siempre que estos estén entre comillas.

Por ejemplo, podríamos definir el vector

```
genero<-c("Mujer","Hombre")
```

```
genero
```

Código 1:

```
x<-c(1,3,5)
c(1,3,5)->x
assign('x', c(10.4, 5.6, 3.1, 6.4, 21.7))
y <- c(x, 0, x)
1:5
y<-seq(-3,3,by=0.5)
y
```

Código 1:

```
rep(0,100)
rep(1:3,3)
length(y)
genero<-c("Mujer","Hombre")
genero
```

Operaciones con vectores

Suma

```
x<-c(1,2,2)
```

```
y<-c(3,1,1)
```

```
z<-x+y
```

Resta

```
z<-x-y
```

Multiplicación por una constante

```
z<-3*x
```

Operaciones con vectores

Multiplicación elemento por elemento

```
z<-x*y
```

Multiplicación vectorial

```
z<-x%*%y
```

Mínimo de un vector

```
z<-min(x)
```

Máximo de un vector

```
z<-max(x)
```

Código 2:

```
x<-c(1,2,2)
```

```
y<-c(3,1,1)
```

```
z<-x+y
```

```
z<-x-y
```

```
z<-3*x
```

```
z<-x*y
```

```
z<-x%*%y
```

```
z<-max(x)
```

```
z<-min(x)
```

Tipos de vectores

```
x<-c(1.5,2.5,3.5,4.2,7.8,9.3)
```

Conversión a entero

```
z<-as.integer(x)
```

Conversión a tipo caracter

```
z<-as.character(x)
```

Conversión a numérico

```
t<-as.numeric(z)
```

Código 2:

```
x<-c(1.5,2.5,3.5,4.2,7.8,9.3)
# Conversión a entero
z<-as.integer(x)
# Conversión a tipo caracter
z<-as.character(x)
# Conversión a numérico
t<-as.numeric(z)
```


Una matriz se define mediante la función `matrix()` a la que hay que especificar sus elementos y su dimensión.

$$\begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$$

```
matriz<-matrix(c(1,2,3,4,5,6,7,8,9),3,3)
```

Las dimensiones (número de filas y columnas) de la matriz pueden obtenerse mediante

`dim(matriz)`

Si queremos obtener elementos concretos de una matriz lo haremos utilizando corchetes para indicar las filas y columnas. Por ejemplo,

`matriz[2,3]`

`matriz[1:2,2:3]`

`matriz[,c(1,3)]`

Tanto para vectores como para matrices, funcionan las operaciones suma y diferencia sin más complicaciones. En el caso del producto, sin embargo, es importante distinguir entre la multiplicación elemento a elemento y el producto matricial.

$\text{matriz}_A + \text{matriz}_B$

$\text{matriz}_A - \text{matriz}_B$

$\text{matriz}_A * \text{matriz}_B$

$\text{matriz}_A \% * \% \text{matriz}_B$ (alternativa: `outer(matrizA, matrizB, "**")`)

Diagonal de una matriz

$\text{diag}(\text{matriz}_A)$

Determinante de una matriz

$\text{det}(\text{matriz}_A)$

Traspuesta de una matriz

$\text{t}(\text{matriz}_A)$

Ejemplo 1: Genera una matriz 5x4 con valores desde 1 hasta el número de celdas de la matriz.

Ejemplo 1:

```
x <- array(1:20, dim=c(5,4))
```

Sistemas de ecuaciones lineales

Sea A la matriz de coeficientes del sistema de ecuaciones lineales y b el vector del lado derecho, la instrucción

`solve(A,b)`

Resuelve el sistema de ecuaciones lineales

Ejemplo:

```
A<- array(c(0,1,-4,2,-2,5,-8,1,9), dim=c(3,3))  
b<- array(c(8,0,-9), dim=c(3,1))  
solve(A,b)
```