MATRICES CON R

Ramon Ceballos

20/2/2021

1. Definir Matrices en R

Para crear una matriz fila, se emplea el comando **matrix(c(...), nrow = 1)**.

```
row = matrix(c(1,2,3,4), nrow = 1)
row
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
```

Para crear una matriz columna, se emplea el comando **matrix(c(...), ncol = 1)**.

```
col = matrix(c(1,2,3), ncol = 1)
col
```

```
##      [,1]
## [1,]    1
## [2,]    2
## [3,]    3
```

Para crear matrices se utiliza la instrucción **`matrix()`**.

- Los datos de la matriz deben ir en un vector **`c()`**.
- Hay que indicar el número de filas y/o columnas con **`nrow`** o **`ncol`**, respectivamente.
- Debemos utilizar el parámetro lógico **`byrow`** para indicar si hemos escrito los números del vector por filas o por columnas.

```
A = matrix(c(1,1,3,5,2,4,3,-2,-2,2,-1,3), nrow = 3, ncol = 4, byrow = TRUE)
A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    1    3    5
## [2,]    2    4    3   -2
## [3,]   -2    2   -1    3
```

```
B = matrix(c(1,0,2,3,3,2,1,-2,3), nrow = 3, byrow = FALSE)
B
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    1
## [2,]    0    3   -2
## [3,]    2    2    3
```

También podemos crear matrices con las funciones **bind()**.

```
#Por filas
C = rbind(c(1,2,3),c(4,5,6),c(7,8,9))
C
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

```
#Por columnas
D = cbind(c(1,2,3),c(4,5,6),c(7,8,9))
D
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

1.1. Acceder a la matriz definida en R

Para acceder a un elemento/fila/columna de una matriz se sigue la sintaxis **A[i,j]**, donde i indica la fila y j, la columna

```
A[3,3] #Elemento a_33
```

```
## [1] -1
```

```
A[1,] #Primera fila
```

```
## [1] 1 1 3 5
```

```
B[,2] #Segunda columna
```

```
## [1] 3 3 2
```

Observación. Tal y como podemos ver en la anterior diapositiva, si dejamos el parámetro de las filas vacío, estamos llamando a todas las filas. Lo mismo ocurre con las columnas si dejamos ese parámetro vacío.

Además, no necesariamente tiene por qué ser simplemente un número lo introducido por parámetro, también puede ser un vector de posiciones con el cual llamar a varias filas o columnas a la vez según pertoque.

1.2. Definir matrices específicas en R

Para crear una **matriz de ceros**, se realiza:

```r
O = matrix(0, nrow = 3, ncol = 3)
O
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    0    0    0
## [3,]    0    0    0
```

Para crear una **matriz de unos**, se realiza:

```r
ones = matrix(1, nrow = 3, ncol = 3)
ones
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    1    1    1
## [3,]    1    1    1
```

Para crear una **matriz diagonal**, utilizamos la función `diag()`.

```r
E = diag(c(1,2,3,4,5,6))
E
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    0    0    0    0    0
## [2,]    0    2    0    0    0    0
## [3,]    0    0    3    0    0    0
## [4,]    0    0    0    4    0    0
## [5,]    0    0    0    0    5    0
## [6,]    0    0    0    0    0    6
```

1.3. Obtener la diagonal, filas, columnas y dimensión de una matriz en R

Para **obtener los elementos de la diagonal de una matriz**, utilizamos la función `diag()` introduciendo por parámetro la matriz pertinente:

```r
M = rbind(c(1,0,1),c(2,-1,5),c(3,3,2))
diag(M)
```

```
## [1]  1 -1  2
```

Para **obtener el número de filas o columnas de una matriz**, utilizamos las funciones `nrow()` o `ncol()`, respectivamente:

```r
nrow(M)
```

```
## [1] 3
```

```r
ncol(M)
```

```
## [1] 3
```

Y, si queremos la **dimensión de la matriz**, utilizamos la función `dim()`, la cual nos devuelve un vector de dos entradas. El primer elemento del vector es el número de filas y, el segundo, el de columnas.

```r
dim(M)
```

```
## [1] 3 3
```

2. Manipulación de matrices en R

La función `sum()` aplicada a una matriz calcula la suma de todos los elementos de dicha matriz:

```r
sum(M)
```

```
## [1] 16
```

Las sumas por filas o por columnas se calculan del siguiente modo:

```r
rowSums(M) #sumas por filas
```

```
## [1] 2 6 8
```

```r
colSums(M) #sumas por columnas
```

```
## [1] 6 2 8
```

La función `prod()` aplicada a una matriz calcula el producto de todos los elementos de dicha matriz:

```r
prod(M)
```

```
## [1] 0
```

La función `mean()` aplicada a una matriz calcula la media (aritmética) de todos los elementos de dicha matriz:

```r
mean(M)
```

```
## [1] 1.777778
```

Las medias por filas o por columnas se calculan del siguiente modo:

```r
rowMeans(M) #sumas por filas
```

```
## [1] 0.6666667 2.0000000 2.6666667
```

```r
colMeans(M) #sumas por columnas
```

```
## [1] 2.0000000 0.6666667 2.6666667
```

3. Operaciones con matrices en R

3.1. Obtener la transpuesta de una matriz

La transpuesta de una matriz se consIgue aplicando la función `t()`.

```r
M
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    1
## [2,]    2   -1    5
## [3,]    3    3    2
```

```r
t(M)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    0   -1    3
## [3,]    1    5    2
```

3.2. Obtener la traza de una matriz

Para calcular la traza de la matriz (la suma de los elementos de la diagonal):

```r
sum(diag(M))
```

```
## [1] 2
```

3.3. Suma de matrices

La suma de matrices:

```r
A = rbind(c(1,2,3),c(4,5,6),c(7,8,9))
B = rbind(c(1,0,2),c(3,0,4),c(5,0,6))
A+B
```

```
##      [,1] [,2] [,3]
## [1,]    2    2    5
## [2,]    7    5   10
## [3,]   12    8   15
```

```
B+A
```

```
##      [,1] [,2] [,3]
## [1,]    2    2    5
## [2,]    7    5   10
## [3,]   12    8   15
```

3.4. Producto de un escalar

El producto de un esclalar por una matriz:

```
5*A
```

```
##      [,1] [,2] [,3]
## [1,]    5   10   15
## [2,]   20   25   30
## [3,]   35   40   45
```

3.5. Producto de matrices

El producto de matrices:

```
A%*%B
```

```
##      [,1] [,2] [,3]
## [1,]   22    0   28
## [2,]   49    0   64
## [3,]   76    0  100
```

```
B%*%A
```

```
##      [,1] [,2] [,3]
## [1,]   15   18   21
## [2,]   31   38   45
## [3,]   47   58   69
```

Fijaos que el producto de matrices se consigue aplicando **%*%** y *no *.

En el segundo caso, lo que hace R es devolver como resultado una matriz $C = (c_{ij})$ cuyos elementos son $c_{ij} = a_{ij} \cdot b_{ij}$.

```
A*B
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    6
## [2,]   12    0   24
## [3,]   35    0   54
```

3.6. Comprobar igualdad de matrices

Para comprobar que dos matrices son iguales, utilizamos el operador lógico `==`

```
A+B == B+A
```

```
##      [,1] [,2] [,3]
## [1,] TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE
```

```
A%*%B == B%*%A
```

```
##       [,1]  [,2]  [,3]
## [1,] FALSE FALSE FALSE
## [2,] FALSE FALSE FALSE
## [3,] FALSE FALSE FALSE
```

Con que haya un `FALSE`, sabemos que son diferentes.

3.7. Calcular potencias de matrices

Para calcular la potencia n-ésima (aproximada) de una matriz, utilizamos la función **`mtx.exp()`, del paquete `Biodem`**.

```
library(Biodem)
mtx.exp(A,4)
```

```
##       [,1]  [,2]  [,3]
## [1,]  7560  9288 11016
## [2,] 17118 21033 24948
## [3,] 26676 32778 38880
```

Para calcular la potencia n-ésima (aproximada) de una matriz, utilizamos **`%^%`, del paquete `expm`**.

```
library(expm)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'expm'
```

```
## The following object is masked from 'package:Matrix':
##
##     expm
```

```
A%^%4
```

```
##       [,1]  [,2]  [,3]
## [1,]  7560  9288 11016
## [2,] 17118 21033 24948
## [3,] 26676 32778 38880
```

4. Rango e inversa de una matriz en R

4.1 Rango de la matriz

El rango de una matriz se calcula con la función **qr()$rank**.

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

```
## [1] 2
```

4.2 Inversa de una matriz

La inversa (aproximada) de una matriz (invertible) se calcula mediante la función **solve()**.

```
solve(M)
```

```
##        [,1]   [,2]   [,3]
## [1,]  2.125 -0.375 -0.125
## [2,] -1.375  0.125  0.375
## [3,] -1.125  0.375  0.125
```

Comprobamos si realmente da la identidad al multiplicar M por su inversa.

```
round(M%*%solve(M))
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

```
round(solve(M)%*%M)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```