Programación en R:

Estructuras de datos bidimensionales, matrices y arrays

Autor: Luis Javier Sánchez Martínez
luisja02@ucm.es

Departamento de Biodiversidad, Ecología y Evolución

Unidad de Antropología Física.

Universidad Complutense de Madrid (UCM)

Introducción

El concepto de matriz en R se puede entender como un vector con un atributo adicional, la dimensión (dim):

La dimensión de una matriz tiene dos componentes:

- El número de filas (nrow)
- El número de columnas (ncol)

Es decir, una matriz es un vector con dimensión (dim). Sin embargo, un vector en R no es una matriz, es decir, en R los vectores no son ni vectores fila ni vectores columna.

- Las matrices en R tienen la dimensión como atributo.
- Los vectores en R no tienen "dimensión".

```
v <- c(1,2,3,4)
dim(v)
NULL
M <- matrix(c(1,2,3,4), nrow=2, ncol=2)
dim(M)
[1] 2 2
```

Como vectores especiales que son las matrices, todos los elementos de la matriz deben ser del mismo tipo (mode) y este puede ser principalmente, al igual que en los vectores normales:

- numérico
- complejo
- cadena
- lógicos

Ejemplos:

Matriz de números reales:

```
M <- matrix(1:20, nrow=4)
M
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
```

Matriz de números complejos:

```
M <- c(1+2i, 3+4i, 0+1i, 6+0i)
dim(M) <- c(2,2)
M
     [,1]  [,2]
[1,] 1+2i  0+1i
[2,] 3+4i  6+0i
```

Matriz de cadenas:

```
M <- matrix( month.name, nrow=3)
M
          [,1]       [,2]    [,3]        [,4]
[1,] "January"   "April"   "July"      "October"
[2,] "February"  "May"     "August"    "November"
[3,] "March"     "June"    "September" "December"
```

Matriz de booleanos:

```
M <- matrix(rep(c(TRUE,FALSE), 4), nrow=4)
M
     [,1]  [,2]
[1,]  TRUE  TRUE
[2,] FALSE FALSE
[3,]  TRUE  TRUE
[4,] FALSE FALSE
```

Crear y deshacer matrices

Antes puntualicemos como cambia la indexación en matrices:

- Los índices en las matrices comienzan en 1. Y aunque se puede acceder a las posiciones de una matriz como con los vectores M[5], lo más correcto es usar [,].

• Es decir, los índices de una matriz entre corchetes (no entre paréntesis) y separados por una coma que nos separa la información para las filas (izquierda), de la información para las columnas (derecha).

Creación de matrices

Hay tres formas básicas de crear matrices:

1. La función matrix().
2. Añadiendo el atributo dimensión a un vector.
3. Uniendo dos o más vectores.

- **La función matrix()**

Sintaxis:

```
matrix(data = …, nrow = 1, ncol = 1, byrow = FALSE,
       dimnames = NULL)
```

donde:

data	vector de datos de longitud n
nrwo	número de filas de la matriz
ncol	número de columnas de la matriz
byrow	forma de llenar el espacio de la matriz. Por defecto, por columnas.
dimnames	nombres de las filas y de las columnas de la matriz (atributo).

En R las matrices por defecto se rellenan y almacenan columna a columna.

```
y <- matrix(c(1,2,3,4), nrow=2, ncol=2)
y
     [,1] [,2]
[1,]    1    3
[2,]    2    4

M <- matrix(c(1,2,3,4),2,2)
M
     [,1] [,2]
```

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

Si se quiere rellenar la matriz por filas hay que utilizar la opción byrow=TRUE.

```
A = matrix(
      c(2, 4, 3, 1, 5, 7),      # Los datos de la matriz
      nrow=2,                   # número de filas
      ncol=3,                   # número de columnas
      byrow = TRUE)             # rellenar por filas

A
     [,1] [,2] [,3]
[1,]    2    4    3
[2,]    1    5    7

dimnames(A) = list(
                c("Fila1", "Fila"),          # nombre filas
                c("Col1", "Col2", "Col3"))   # nombre columnas

A
      Col1 Col2 Col3
Fila1    2    4    3
Fila     1    5    7
```

Los nombres de las filas y las columnas de una matriz se pueden especificar de dos formas distintas, con el argumento dimnames dentro de la función matrix(), o bien utilizando las funciones rownames() y colnames() fuera de la función matrix().

```
M <- matrix(c(1,2,3,4),2,2)
M
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

```
rownames(M) <- c('Fila1', 'Fila2')
M
      [,1] [,2]
Fila1    1    3
Fila2    2    4

colnames(M) <- c('Columna1', 'Columna2')
 M
      Columna1 Columna2
Fila1        1        3
Fila2        2        4
```

Para borrar los nombres de filas y columnas:

```
rownames(M) <-c()
colnames(M) <-c()
M
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

- **Añadir el atributo dimensión a un vector**

Otra manera de crear una matriz sería a partir de un vector, añadiéndole el atributo de dimensión, el cual es propio de las matrices.

```
vector <- 1:20
class(vector)
[1] "integer”
dim(vector)
NULL
dim(vector) <- c(5,4)
```

```
vector
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20

class(vector)
[1] "matrix"
```

- **Unir dos o más vectores**

Otra manera muy usada de crear una matriz es mediante la unión de dos o más vectores con las funciones:

• cbind() los une como vectores columna

• rbind() los une como vectores fila

```
vector1 <- 1:5
vector2 <- 6:10
vector3 <- 11:15

cbind(vector1, vector2, vector3)
     vector1 vector2 vector3
[1,]       1       6      11
[2,]       2       7      12
[3,]       3       8      13
[4,]       4       9      14
[5,]       5      10      15

rbind(vector1, vector2, vector3)
        [,1] [,2] [,3] [,4] [,5]
vector1    1    2    3    4    5
vector2    6    7    8    9   10
vector3   11   12   13   14   15
```

Cómo deshacer matrices

Una matriz se convierte en vector, es decir pierde el atributo dimensión, aplicando la función c() a la matriz.

Ejemplo:

Veamos como pasar de la matriz M al vector M1

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

dim(M)
[1] 2 2
is.matrix(M)
[1] TRUE

M1 <- c(M)
dim(M1)
NULL
is.matrix(M1)
[1] FALSE
```

Forma de seleccionar elementos de una matriz

Como en el caso de los vectores hay diferentes formas de seleccionar elementos de una matriz.

- **Selección especificando los índices**

Veamos diferentes situaciones:

Para seleccionar un único elemento se emplea la sintaxis M[i,j] que toma el elemento de la i-ésima fila y la j-ésima columna.

Para seleccionar una fila la sintaxis es M[i,] que selecciona toda la i-ésima fila.

Para seleccionar una columna la sintaxis es M[,j] que selecciona toda la i-ésima columna.

Para seleccionar varias columnas y filas:

- M[3:7,] selecciona las filas consecutivas 3, 4, 5, 6, 7
- M[,seq(8,12)] selecciona las columnas consecutivas 8, 9, 10, 11, 12
- M[c(3,9,13),] selecciona las filas 3, 9, 13

Cuando el índice es negativo, el efecto que se consigue es excluir los elementos referenciados. Por ejemplo, y[-2,]elimina la segunda fila del la matriz y.

```
y <- matrix(1:10, nrow=5)
y
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10

y[-2,]            # Eliminamos la segunda fila de la matriz y
     [,1] [,2]
[1,]    1    6
[2,]    3    8
[3,]    4    9
[4,]    5   10
```

Por defecto, los elementos seleccionados de una matriz se convierten en vectores. Para que los elementos seleccionados conserven el atributo de la dimensión es necesario especificar el argumento drop.

```
Mat <- matrix(1:20, nrow=5)
Mat[,c(2,4)]
Mat[,2]
Mat[,2, drop=FALSE]
Mat[3,, drop=FALSE]
```

- **Selección estableciendo filtros sobre matrices**

Se pueden establecer filtros sobre matrices utilizando operadores relacionales, veamos que selección hacen los siguientes trozos de código:

```
Mat[Mat[,2]>=10,]
```

```
j = Mat[1,]!=6
Mat[,j]
```

Añadir, borrar filas y columnas de una matriz

- **Funciones rbind() y cbind()**

Las funciones rbind() y cbind() las hemos utilizado para crear matrices a partir de un conjunto de vectores. Otra de las acciones que realizan estas dos funciones es combinar dos o más matrices para obtener una tercera matriz ampliada.

- La función cbind() crea una nueva matriz combinando dos matrices, añade las columnas de la segunda a la primera.
- La función rbind() crea una nueva matriz combinando dos matrices, añade las filas de la segunda a la primera.

```
mat1 <- matrix(1:16, ncol = 4)
mat1
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

```
mat2 <- matrix(30:53, ncol = 4)
mat2
     [,1] [,2] [,3] [,4]
[1,]   30   36   42   48
[2,]   31   37   43   49
[3,]   32   38   44   50
```

```
[4,]   33   39   45   51
[5,]   34   40   46   52
[6,]   35   41   47   53
```

```
mat3 <- matrix(100:119, nrow = 4)
mat3
     [,1] [,2] [,3] [,4] [,5]
[1,]  100  104  108  112  116
[2,]  101  105  109  113  117
[3,]  102  106  110  114  118
[4,]  103  107  111  115  119
```

```
# Función rbind: una matriz encima de la otra
rbind(mat1, mat2)
      [,1] [,2] [,3] [,4]
 [1,]    1    5    9   13
 [2,]    2    6   10   14
 [3,]    3    7   11   15
 [4,]    4    8   12   16
 [5,]   30   36   42   48
 [6,]   31   37   43   49
 [7,]   32   38   44   50
 [8,]   33   39   45   51
 [9,]   34   40   46   52
[10,]   35   41   47   53
```

```
rbind(mat1, mat3)
Error in rbind(mat1, mat3) :
  el número de columnas de las matrices debe coincidir (vea arg 2)
```

```
# Función cbind: una matriz al lado de la otra
cbind(mat1, mat3)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]    1    5    9   13  100  104  108  112  116
```

```
[2,]    2    6   10   14  101  105  109  113  117
[3,]    3    7   11   15  102  106  110  114  118
[4,]    4    8   12   16  103  107  111  115  119
```

```
cbind(mat1, mat2)
Error in cbind(mat1, mat2) :
  el número de filas de las matrices debe coincidir (vea arg 2)
```

Funciones sobre matrices. Función apply()

El objetivo de esta función consiste en la aplicación de otra segunda función por filas o por columnas.

Sintaxis:

apply(matriz, código de dimensión, función, argumentos de la función)

Ejemplo: función mean()

Dada la matriz m se quiere calcular la media de cada una de sus columnas.

Para resolver este problema vamos a aplicar la función mean() a cada una de las columnas por medio de la función apply().

```
m <- matrix(1:20, nrow=4, ncol=5)
m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
```

```
# Se aplica la función mean() por columnas
apply(m,2,mean)
[1]  2.5  6.5 10.5 14.5 18.5
```

```
# La función colMeans() también obtiene este resultado:
colMeans(m)
[1]  2.5  6.5 10.5 14.5 18.5

m <- matrix(1:20, nrow=4, ncol=5)
m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20

apply(m,1,mean)
[1]  9 10 11 12
rowMeans(m)
[1]  9 10 11 12
```

Ejemplo 2:

```
# Definimos la función
fun <- function(x){(mean(x) + median(x))/2}
# Matriz m2
m2 <- matrix(seq(1,6), nrow=3)
m2
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

# Aplicamos la función a las filas de la matriz m2
y <- apply(m2,1,fun)
y
[1] 2.5 3.5 4.5
```

Operaciones con matrices

- Transpuesta de una matriz

La transpuesta de una matriz cualquiera sería aquella matriz en la que las filas se corresponden con las columnas de la anterior matriz, y las columnas con las filas de la anterior matriz.

Ejemplo:

```
# Matriz B
B <- matrix(c(1,0,0,-1,0,1,0,0,1,0), ncol=2, byrow=TRUE) # se rellena por filas
B
     [,1] [,2]
[1,]    1    0
[2,]    0   -1
[3,]    0    1
[4,]    0    0
[5,]    1    0

# transpuesta de B
t(B)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    1
[2,]    0   -1    1    0    0
```

- Determinante de una matriz cuadrada

La función determinante (det()) es de gran importancia en el álgebra matricial, ya que, por ejemplo, nos permite saber si un matriz es regular (si tiene inversa) y, por tanto, si un sistema de ecuaciones lineales tiene solución.

```
m <- matrix(c(2,1,4,2),ncol=2, byrow=TRUE)
m
     [,1] [,2]
[1,]    2    1
[2,]    4    2
```

```
det(m)
[1] 0
```

El valor de 0 nos indica una dependencia lineal entre las filas de la matriz

- Suma de matrices

En R esta operación se realiza simplemente utilizando la función suma '+'

Ejemplo:

```
A <- matrix(1:20, nrow=5)
B <- matrix(21:40, nrow=5)
```

```
A
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
```

```
B
     [,1] [,2] [,3] [,4]
[1,]   21   26   31   36
[2,]   22   27   32   37
[3,]   23   28   33   38
[4,]   24   29   34   39
[5,]   25   30   35   40
```

```
A+B
     [,1] [,2] [,3] [,4]
[1,]   22   32   42   52
[2,]   24   34   44   54
[3,]   26   36   46   56
[4,]   28   38   48   58
[5,]   30   40   50   60
```

- Producto de un escalar por una matriz

En R esta operación se realiza utilizando la función producto '*' que multiplica el escalar por cada uno de los elementos de la matriz.

Ejemplo:

```
A <- matrix(1:20, nrow=5)
lambda <- 10
lambda*A
```

```
     [,1] [,2] [,3] [,4]
[1,]   10   60  110  160
[2,]   20   70  120  170
[3,]   30   80  130  180
[4,]   40   90  140  190
[5,]   50  100  150  200
```

- Producto elemento a elemento de dos matrices

Dadas dos matrices de igual dimensión se define el producto elemento a elemento de dos matrices como el resultado de: $a_{ij} \cdot b_{ij}$

En R esta operación se realiza utilizando la función'*'

Ejemplo:

```
A <- matrix(1:20, nrow=5)
B <- matrix(21:40, nrow=5)

A
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
```

```
B
```

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

```
[1,]  21  26  31  36
[2,]  22  27  32  37
[3,]  23  28  33  38
[4,]  24  29  34  39
[5,]  25  30  35  40
```

```
A*B
     [,1] [,2] [,3] [,4]
[1,]   21  156  341  576
[2,]   44  189  384  629
[3,]   69  224  429  684
[4,]   96  261  476  741
[5,]  125  300  525  800
```

- Producto matricial de dos matrices

La matriz resultante del producto de dos matrices M_{nxp} y M_{pxm} es una tercera matriz M_{nxm}

En R esta operación se realiza utilizando la función producto '%*%'.

Ejemplo:

```
# Matriz A
A <- matrix(c(1,0,0,0,3,0,2,0,1,0,1,0,-1,0,1), ncol=5)    # se rellena por columnas
A
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    2    0   -1
[2,]    0    3    0    1    0
[3,]    0    0    1    0    1

# Matriz B
B <- matrix(c(1,0,0,-1,0,1,0,0,1,0), ncol=2, byrow=TRUE) # se rellena por filas
B
     [,1] [,2]
[1,]    1    0
[2,]    0   -1
```

```
[3,]    0    1
[4,]    0    0
[5,]    1    0
```

```
#Matric C = A%*%B
C <- A%*%B
C
     [,1] [,2]
[1,]    0    2
[2,]    0   -3
[3,]    1    1
```

La matriz C es resultado de la operación de la matriz A_{3x5} y B_{5x2}, por lo tanto la matriz C resultante presentará 3 filas y 2 columnas (C_{3x2})

- Resolución de sistemas de ecuaciones:

En R tenemos la función solve que nos resuelve sistemas de ecuaciones en forma matricial, veamos de forma práctica como podemos resolver mediante R un sistema de ecuaciones dado:

$$2x + 3y – z = 2$$
$$3x – y – z = 5$$
$$2x + 4y - 2z = 10$$

El primer paso sería transformar en matriz tanto la parte de las incógnitas, como la de los resultados, para que la función solve() pueda trabajar con ello:

```
incognitas = matrix(c(2,3,2,3,-1,4,-1,-1,-2), ncol=3)
res = matrix(c(2,5,10))
```

Una vez ya tenemos ambas matrices creadas, aplicamos la función solve() sobre ellas, dando en el primer argumento la matriz de las incógnitas y en el segundo la matriz de los resultados:

```
solve(incognitas, res)
     [,1]
[1,] -1.8
[2,] -1.2
[3,] -9.2
```

Arrays

Un array en R es un tipo de objeto concreto que se puede entender como la generalización de una matriz de dos dimensiones al caso multidimensional. Su definición de manera forman a nivel de sintaxis es de la forma:

array(datos, dimensiones)

En este caso las dimensiones hay que rellenarlas en forma de vector indicando en este orden: Número de filas, número de columnas y número de matrices. De esa manera si quisiéramos crear un array formado por 3 matrices de 3 filas y 2 columnas cada una:

```
array(1:18,c(3,2,3))
, , 1

     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

, , 2

     [,1] [,2]
```

```
[1,]    7   10
[2,]    8   11
[3,]    9   12

, , 3

     [,1] [,2]
[1,]   13   16
[2,]   14   17
[3,]   15   18
```

Los comandos para manejar arrays son similares a los que manejan matrices. Sin embargo cambian ciertos aspectos de aplicación, por ejemplo con la indexación, pues ahora tenemos una dimensión más que manejar. Vamos a crear un array supuesto:

```
datos<-array(c(45,46,65,55,170,167,48,49,68,56,169,165),c(2,3,2))
dimnames(datos)<-list(c("hombres", "mujeres"), c("edad" ,"peso", "altura"), c("Madrid", "Barcelona"))
datos

, , Madrid

         edad peso altura
hombres   45   65    170
mujeres   46   55    167

, , Barcelona

         edad peso altura
hombres   48   68    169
mujeres   49   56    165
```

Con la función dimnames tenemos un resumen de lo que podemos encontrar en cada una de las posiciones de nuestro objeto:

```
dimnames(datos)
[[1]]
[1] "hombres" "mujeres"

[[2]]
[1] "edad"   "peso"   "altura"

[[3]]
[1] "Madrid"    "Barcelona"
```

Accedamos a distintas posiciones de nuestro Array.
Por ejemplo podemos acceder solo a los datos de hombres:

```
datos["hombres",,]
       Madrid Barcelona
edad       45       48
peso       65       68
altura    170      169
```

Podemos acceder a los datos de edad en función del sexo y de la ciudad:

```
datos[,"edad",]
         Madrid Barcelona
hombres     45       48
mujeres     46       49
```

Podemos acceder solo a los datos de una de las ciudades:

```
datos[,,"Madrid"]
         edad peso altura
hombres   45   65    170
mujeres   46   55    167
```

¿Practicamos un poco?

1.- Se ha definido el vector: x = c(1,2,3,4,5,6), a partir de dicho vector se han construido las siguientes matrices:

```
> m1
     [,1] [,2] [,3]
[1,]  1    3    5
[2,]  2    4    6
> m2
     [,1] [,2]
[1,]  1    4
[2,]  2    5
[3,]  3    6
> m3
    [,1] [,2] [,3]
[1,]  1    2    3
[2,]  4    5    6
> m4
     [,1] [,2] [,3]
[1,]   1    4    1
[2,]   2    5    2
[3,]   3    6    3
```

Intenta reproducir el código necesario para obtener cada una de ellas.

2.- ¿Qué ocurre cuando definimos una matriz en R y sólo especificamos el número de filas o el número de columnas? ¿Qué ocurre cuándo los datos no se corresponden con la dimensión de la matriz que queremos definir? Compruébalo ejecutando los siguientes comandos:

```
> matrix(1:6,nrow=2)
> matrix(1:6,nrow=4)
> matrix(1:6,nrow=4,ncol=4)
```

3.- Sean A una matriz 2 × 3, B una matriz 3 × 4 y C una matriz 2 × 3. ¿De qué tipo y dimensión serán los objetos obtenidos de los siguientes comandos de R? ¿Alguno de los comandos produce mensajes de error? ¿Por qué?

a) A*B

b) A+2

c) A%*%B

d) exp(B) *Nota: exp() es la función exponencial.*

e) A*C

f) A%*%C

4.- Ante la siguiente matriz 5x5, que recoge la edad de un grupo de 5 amigas y cinco amigos:

```
> M <- matrix(c(33,28,31,29,30,45,36,28,30,22), ncol=2,nrow=5, byrow=TRUE)
> colnames(M)<-c("Chicos","Chicas")
> M
     Chicos Chicas
[1,]     33     28
[2,]     31     29
[3,]     30     45
[4,]     36     28
[5,]     30     22
```

¿Cuál es la edad media del grupo de chicos? ¿Y del grupo de chicas? Intenta pensar 3 maneras distintas de calcular los dos valores.

Solucionario

1

```
x <- c(1,2,3,4,5,6)

m1 <- matrix(x, nrow=2, ncol=3)
m2 <- matrix(x, nrow=3, ncol=2)
m3 <- matrix(x, nrow=2, ncol=3, byrow = TRUE)
m4 <- matrix(x, nrow=3, ncol=3)
```

2

```
matrix(1:6,nrow=2)
#En este primer caso, al crear la matriz, R autocompleta el número de filas o columnas no establecido para que encaje con el número de elementos

matrix(1:6,nrow=4)
matrix(1:6,nrow=4,ncol=4)
#En el caso en el que el número de elementos no es multiplo de filas · columnas, R repite los elementos hasta que consigue rellenar todas las posiciones (además de emitir un warning)
```

3

```
A <- matrix(1, nrow=2, ncol=3)
B <- matrix(1, nrow=3, ncol=4)
C <- matrix(1, nrow=2, ncol=3)

#a)
A*B #mensaje de error porque las matrices no tiene la misma cantidad de elemntos

#b)
A+2 #matriz 2x3, ya que unicamente se suma 2 a cada elemento

#c)
A%*%B #matriz 2x4, ya que es una multiplicación matricial de A (2X3) con B (3x4)

#d)
exp(B) #matriz 3x4, ya que exp simplemente calcula el exponencial de cada elemento

#e)
A*C #matriz 2x3, ya que se realiza una multiplicación elemento a elemento
```

#f)

A%*%C #mensaje de error ya que no coinciden las columnas de A (2X3) con las filas de C (2X3)

4

mean(M[,1]); mean(M[,2])

colMeans(M)

apply(M,2,mean)