Matrices con R

Daniel Eduardo Macias Estrada

6/8/2020

Introducción

En este documento se explicará acerca del manejo más básico de las matrices en R, empezando con su definición, hasta terminar con el rango y la inversa.

Toda la información recabada está basado enteramente de la obra de Juan Gabrial Gomila Salas, CEO de Frogames, Matemático, Data Scientist & Game Designer

Definición de matrices Empezando con lo más sencillo, hablemos sobre como declarar una matriz en una variable.

Para crear un vector fila:

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

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

Para crear una matriz columna:

```
col <- matrix(c(1,2,3,4),ncol=1)
col
```

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

Como notamos a primera vista, con la función **matrix()** creamos un objeto que será nuestra matriz. De manera más general, ésta se usa de la siguiente manera.

```
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(1,4,7,2,5,8,3,6,9), nrow = 3, byrow = FALSE)
B
```

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

En la función, los argumentos **nrow** y **ncol** indican el número de filas y columnas respectivamente. Por otra parte **byrow** indica como se leeran los elementos del vector: por filas (TRUE) o por columnas (FALSE). Al mismo tiempo, en caso de solo aclarar el número de filas o el de columnas, se calculara el número de elementos por columna o fila, respectivamente. En caso de que no sea posibla, se lanzará un error.

Existe otra manera de declarar una matriz, con las funciones **bind()**

```
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
```

```
D = cbind(c(1,2,-1), c(0,5,10), c(17,8,2))
D
```

```
##      [,1] [,2] [,3]
## [1,]    1    0   17
## [2,]    2    5    8
## [3,]   -1   10    2
```

Acceder a un elemento de la matriz Para lograr este fin se sigue la siguiente sintáxis: **A[i,j]**, en donde se indica la fila con la variable i, y la columna con la variable j

```
A[3,3] #elemento: a_{33}
```

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

```
A[1,] #toda la primera fila
```

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

```
A[,1] #toda la primer columna
```

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

Como se observa en los 2 últimos ejemplos, si no aclaramos la fila o columna entonces nos dara toda la columna o fila entera. Además es posible pasar un vector con las posiciones de los elementos que requiramos

Matriz nula y matriz identidad Con la misma función **matrix()** podemos decarar una matriz con todos sus elementos nulos

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

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

Para que todos los elementos valgan un único valor, solo se pasa un solo valor

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

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

Matrices diagonales Para realizar una matriz diagonal de manera más sencilla, es posible usar la función **diag()**, en el que solo se pasará como argumento un vector con los elementos

```
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
```

Para visualizar los elementos de la diagonal principal de una matriz, usamos esta misma función pero pasando como parámetro la matriz.

```
diag(E)
```

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

Números de filas y columnas, dimensión Para obtener el número de filas y columnas, usamos las funciones **nrow()** y **ncol()**

```
nrow(A)
```

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

```
ncol(A)
```

```
## [1] 4
```

En caso de querer obtener el orden de la matriz usamos la función **dim()**, a la cual le pasaremos la matriz como parámetro. Este devolverá un vector con dos elementos, el primero como el número de filas y el segundo como el número de columnas

```
dim(A)
```

```
## [1] 3 4
```

Manipulación de matrices

Suma de elementos de una matriz Para sumar todos los elementos de la matriz, usamos la función **sum()**

Tomaremos el ejemplo de la matriz C

```
C
```

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

```
sum(C)
```

```
## [1] 45
```

Además se puede obtener un vector con el resultado de las sumas de los elementos de todas las filas de la matriz con la función **rowSums()** al igual que por columnas mediante **colSums()**

```
rowSums(C)
```

```
## [1]  6 15 24
```

```
colSums(C)
```

```
## [1] 12 15 18
```

Producto de los elementos de una matriz En caso de querer obtener el resultado del producto de todos los elementos de la matriz con la función **prod()**

```
prod(C)
```

```
## [1] 362880
```

Media aritmética de los elementos de una matriz Otra función útil es **mean()**, que obtiene la media aritmética de los elementos de la matriz

```
mean(C)
```

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

Además, es posible obtener la media de cada fila o columna, para ello usamos la función **rowMeans()** y **colMeans()**

```
rowMeans(C)
```

```
## [1] 2 5 8
```

```
colMeans(C)
```

```
## [1] 4 5 6
```

Operaciones con matrices

Transpuesta de una matriz La trasnspuesta de una matriz se consigue usando la función **t()**

```
D
```

```
##      [,1] [,2] [,3]
## [1,]    1    0   17
## [2,]    2    5    8
## [3,]   -1   10    2
```

```
t(D)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2   -1
## [2,]    0    5   10
## [3,]   17    8    2
```

Traza de una matriz La traza de una matriz es la suma de los elementos de su diagonal principal. Podemos usar la siguiente combinación de comandos para obtenerla

```
sum(diag(D))
```

```
## [1] 8
```

Suma de matrices Es fácil realizarla

```
A = rbind(c(4,2,-6), c(2,8,1), c(-6, 10, -3))
B = rbind(c(3,1,2), c(-2,12,5), c(7,4,2))

A+B
```

```
##      [,1] [,2] [,3]
## [1,]    7    3   -4
## [2,]    0   20    6
## [3,]    1   14   -1
```

```
B+A
```

```
##      [,1] [,2] [,3]
## [1,]    7    3   -4
## [2,]    0   20    6
## [3,]    1   14   -1
```

```
5*A
```

Producto de un escalar por una matriz

```
##      [,1] [,2] [,3]
## [1,]   20   10  -30
## [2,]   10   40    5
## [3,]  -30   50  -15
```

Producto de una matriz por otra matriz Para poder multiplicar matrices el asterístico debe ir entre signos de porcentaje, %*%

En caso de poner solamente el * se devolverá como resultado una matriz $C = (c_{ij})$ cuyos elementos son $c_{ij} = a_{ij} \cdot b_{ij}$. A este proceso se le conoce como producto tensorial, o producto elemento a elemento.

```
A%*%B
```

```
##      [,1] [,2] [,3]
## [1,]  -34    4    6
## [2,]   -3  102   46
## [3,]  -59  102   32
```

Igualdad entre matrices Para conocer si una matriz es igual a otra se usa el operador lógico ==. Al haber solamente un FALSE, las matrices serán diferentes

```
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
```

Potenciación Para calcular la potencia n-ésima (aproximada) de una matriz, podemos llamar a dos funciones de librerias distintas. Calcular esta potencia es costoso para la máquina, por ello que el resultado es aproximado. La primera es **mtx.exp()** del paquete **Biodem**

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

```
##      [,1]  [,2]  [,3]
## [1,] 2432 -4976  -192
## [2,] 2314  5170 -1003
## [3,] 2238  4550  2703
```

El otro caso presentado, elevar una matriz se indica con los simbolos %^% y el número del exponente. Es necesario la librería **expm** para usarlo

```
library(expm)
```

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

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

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

```
A%^%4
```

```
##      [,1]  [,2]  [,3]
## [1,] 2432 -4976  -192
## [2,] 2314  5170 -1003
## [3,] 2238  4550  2703
```

Rango e inversa

Rango de una matriz Para obtener el rango de una matriz se usa la función **qr()$rank**, qr es un método para operar una matriz, y lo que le prosigue es la propiedad de la función, la cual se accede con la sintáxis de $nombre_de_propiedad.

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

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