Matrices y Vectores

Introducción:

Bienvenido a este recurso de instrucción básica en R sobre vectores y matrices, primero comenzaremos con un pequeño repaso de álgebra. Tanto las matrices como los vectores son conjuntos de datos numéricos o alfanuméricos. Los vectores son conjuntos de datos unidimensionales, mientras que las matrices son bidimensionales. Ademas las matrices se organizan en filas y columnas que se pueden identificar como vectores. Gracias a estas características, podremos aplicar nuestros conocimientos en R para ordenar y combinar la información y así obtener el resultado de distintas operaciones algebraicas, gráficos, estadísticas…

Vectores:

1.

Un vector siempre debe estar asociado a un título o variable que nosotros definamos en la plataforma, un ejemplo sería x. Nuestro vector siempre lo definiremos mediante el comando =c seguido de un paréntesis con los valores que queramos introducir. Como ya hemos mencionado se pueden combinar valores numéricos y de caracteres, pero si introducimos un carácter no podremos realizar operaciones ya que nuestro programa detectará el vector como alfanumérico. Deberemos usar comillas para los valores de caracteres que introduzcamos, si no se reconocerán como variables previamente definidas.

```
x=c(1, 2, “azul”, 3)
```

Las componentes de nuestro vector vendrán definidas por el nombre del vector seguido de un corchete que encierre el número de componente en cuestión. Cuando nuestro vector haya sido registrado ya podremos disponer de él para posteriores operaciones y análisis. Si nosotros por ejemplo quisiéramos analizar todo el contenido de nuestro vector simplemente deberíamos escribir el nombre de nuestro vector (x) y el programa nos mostraría todos sus valores. Si por el contrario quisiéramos observar sólo una de sus componentes utilizaríamos el nombre del vector seguido del corchete con su número de componente.

```
x
[1] “1” “2” “azul” “3”
```

```
x[4]
[1] 3
```

2.
Las operaciones con vectores son las que siempre se han trabajado en álgebra, suma, resta y multiplicación. Para realizar dichas operaciones deberemos definir al menos dos vectores anteriormente y utilizaremos el signo + para realizar su suma componente a componente, el signo - para su resta componente a componente, el signo %*% para el producto vectorial y por último usaremos * para llevar a cabo un producto escalar componente a componente. Para almacenar estos nuevos valores definiremos un nuevo vector (con diferente nombre o no) como la operación que queremos realizar.

```
x=c(1, 2, 3)
y=c(4, 5, 6)
```

```
Z=x+y
Z
[1] 5 7 9
```

```
Z=x-y
Z
[1] -3 -3 -3
```

```
Z=x*y
Z
[1] 4 10 18
```

```
Z=x%*%y
Z
[1] 32
```

También podemos seleccionar componentes concretas de cada vector y realizar diferentes operaciones entre componentes. Uno de los usos más comunes es un *bucle for* (se recomienda leer el recurso que trata los bucles for) en el que un vector nos da diferentes datos o medidas de un sistema y se realiza la misma operación con los diferentes valores del vector hasta que sea deseado.

```
x=c(1, 2, 3)
y=c(4, 5, 6)
```

```
Z=0
for(i in 1:3){
a[i]=y[i] + z[i]
}
a
[1] 5 7 9
```

Matrices:

1- Almacenaje y análisis de vectores
2- Ordenación de vectores
3- Operaciones con matrices

1.
Una matriz también deberá estar asociada siempre a un título como podría ser A, la definiremos de manera similar a los vectores, introduciendo el nombre de la matriz seguido del comando =matrix para luego abrir un paréntesis en el que asignaremos los valores y dimensiones de la matriz. Los valores los introduciremos con el comando c seguido de todas las componentes. Cada una de las dimensiones tiene su propio comando específico: para designar el número de filas y columnas usaremos los comandos nrow y ncol

respectivamente. Solo usando estos dos comandos (e incluso uno) la matriz ya se podrá definir, rellenará sus posiciones con las componentes establecidas siguiendo el número filas y columnas (rellenando por columnas en orden). Al igual que con los vectores, si queremos visualizar la matriz completa escribiremos el nombre de la matriz y si solo queremos una componente, el nombre seguido del corchete con la posición de la componente.

```
A=matrix( c(1, 2, 3, 4), nrow=2, ncol=2)
A
     [,1] [,2]
[1,]  1    3
[2,]  2    4
```

2.

Usando el comando cbind o rbind podremos generar una matriz a base de unos vectores previamente definidos (todos deben ser de iguales dimensiones o por lo menos se deber haber definido la matriz como nula). Con cbind la matriz tendrá cada una de sus columnas definida por cada vector y con rbind lo mismo pero cada vector corresponderá a una fila.

```
x=c(1, 2, 3)
y=c(4, 5, 6)
```

```
A=cbind(x, y)
A
     x  y
[1,] 1  4
[2,] 2  5
[3,] 3  6
```

```
A=rbind(x, y)
A
  [,1] [,2] [,3]
x    1    2    3
y    4    5    6
```

Al definir las dimensiones de la matriz podríamos utilizar el comando length(…) para que esta tuviera el número de filas o columnas igual al número de componentes de cierto vector.

3.

En este caso las operaciones posibles también serán las que se han utilizado siempre en el álgebra. Para la suma y resta utilizaremos + y - respectivamente, para el producto entre dos matrices %*% y para un producto componente a componente *. Además R nos permite programar operaciones más complejas de manera directa con los siguientes comandos: t(…) para realizar la traspuesta de la matriz, det(…) para realizar el determinante, solve(…) para determinar la inversa, solve(…1,…2) para hallar la solución al sistema …1x=…2, svd(…) para su descomposición en valores singulares, qr(…) para su factorización QR, eigen(…) para obtener sus vectores propios, diag(a) para obtener la matriz diagonal en la que a es una matriz y diag(A) en donde A es un vector.

```
A=matrix(c(1,2,3,4,5,6,7,8,9), nrow=3)
> B=matrix(c(10,11,12,13,14,15,16,17,18), nrow=3)
> a=c(1,2,3,4)
>
> z=A+B
> z
     [,1] [,2] [,3]
[1,]   11   17   23
[2,]   13   19   25
[3,]   15   21   27
> z=A-B
> z
     [,1] [,2] [,3]
[1,]   -9   -9   -9
[2,]   -9   -9   -9
[3,]   -9   -9   -9
> z=A*B
> z
     [,1] [,2] [,3]
[1,]   10   52  112
[2,]   22   70  136
[3,]   36   90  162
> z=A%*%B
> z
     [,1] [,2] [,3]
[1,]  138  174  210
[2,]  171  216  261
[3,]  204  258  312
> z=t(A)
> z
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> z=det(A)
> z
[1] 0
> z=svd(A)
> z
$d
[1] 1.684810e+01 1.068370e+00 5.039188e-17

$u
           [,1]       [,2]       [,3]
[1,] -0.4796712  0.77669099  0.4082483
[2,] -0.5723678  0.07568647 -0.8164966
[3,] -0.6650644 -0.62531805  0.4082483

$v
           [,1]       [,2]       [,3]
[1,] -0.2148372 -0.8872307 -0.4082483
[2,] -0.5205874 -0.2496440  0.8164966
[3,] -0.8263375  0.3879428 -0.4082483
```

```
> z=qr(A)
> z
$qr
           [,1]      [,2]          [,3]
[1,] -3.7416574 -8.552360 -1.336306e+01
[2,]  0.5345225  1.963961  3.927922e+00
[3,]  0.8017837  0.988693  1.776357e-15

$rank
[1] 2

$qraux
[1] 1.267261e+00 1.149954e+00 1.776357e-15

$pivot
[1] 1 2 3

attr(,"class")
[1] "qr"
> z=eigen(A)
> z
eigen() decomposition
$values
[1]  1.611684e+01 -1.116844e+00
-5.700691e-16

$vectors
           [,1]       [,2]       [,3]
[1,] -0.4645473 -0.8829060  0.4082483
[2,] -0.5707955 -0.2395204 -0.8164966
[3,] -0.6770438  0.4038651  0.4082483

> z=diag(A)
> z
[1] 1 5 9
> z=diag(a)
> z
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    2    0    0
[3,]    0    0    3    0
[4,]    0    0    0    4
```