Introducción a R

Email: delval@decsai.ugr.es

Alberto Armijo Ruiz

1. Matrices

* Ejecuta los siguientes comandos.

```
matrix(data=5, nr=2, nc=2)

matrix(1:6, 2, 3)

matrix(1:6, 2, 3, byrow=TRUE)
```

```
> matrix(data=5, nr=2, nc=2)
     [,1] [,2]
[1,]    5    5
[2,]    5    5
> matrix(1:6, 2, 3)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> matrix(1:6, 2, 3, byrow=TRUE)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

El primer argumento determina un vector con datos que se van a introducir en la matriz, si se incluyen menos datos que filas y columnas se aplica la regla del reciclaje. Después se especifica el número de filas y el número de columnas; opcionalmente se puede elegir si se quiere introducir los datos por columnas o por filas.

* Crea un vector z con los 30 primeros números y crea con el una matriz m con 3 filas y 10 columnas.

```
> z=1:30
> m=matrix(z,nrow=3,ncol=10)
> m
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    4    7   10   13   16   19   22   25    28
[2,]    2    5    8   11   14   17   20   23   26    29
[3,]    3    6    9   12   15   18   21   24   27    30
```

* Escribe la tercera columna en un vector

```
> vec=m[,3]
> vec
[1] 7 8 9
```

* Create in R the matrices

```
x =3 2l
  −1  1

y =1 4    0
   0 1    -1
```

```
> m_x = matrix(data=c(3,1,'2l',1),nrow = 2,ncol=2)
> m_x
     [,1] [,2]
[1,] "3"  "2l"
[2,] "1"  "1"

> m_y = matrix(data = c(1,0,4,1,0,-1),nrow=2,ncol=3)
> m_y
     [,1] [,2] [,3]
[1,]    1    4    0
[2,]    0    1   -1
```

Y calcula los efectos de los siguientes comandos

(a) x[1,]

```
> m_x[1,]
[1] "3"  "2l"
```

(b) x[2,]

```
> m_x[2,]
[1] "1" "1"
```

(c) x[,2]

```
> m_x[,2]
[1] "2l" "1"
```

(d) y[1,2]

```
> m_y[1,2]
[1] 4
```

(e) y[,2:3]

```
> m_y[,2:3]
     [,1] [,2]
[1,]    4    0
[2,]    1   -1
```

* Transforma la matriz m que creaste en el ejercicio anterior en un array multidimensional. (Pista: averigua lo que puedas de la función dim().)

La función dim() devuelve un vector con las dimensiones del objeto, primero las filas, después las columnas, y el resto de dimensiones de un array.

```
> dim(m)
[1]  3 10
> m_array = array(data=m,dim=dim(m))
> m_array
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    4    7   10   13   16   19   22   25    28
[2,]    2    5    8   11   14   17   20   23   26    29
[3,]    3    6    9   12   15   18   21   24   27    30
```

* Crea un array de 5 x 5 x 2 y rellénalo con valores del 1 al 50. Investiga la función array(). Llama al array x

```
> x = array(1:50,dim=c(5,5,2))
> x
, , 1

     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
[2,]    2    7   12   17   22
[3,]    3    8   13   18   23
[4,]    4    9   14   19   24
[5,]    5   10   15   20   25

, , 2

     [,1] [,2] [,3] [,4] [,5]
[1,]   26   31   36   41   46
[2,]   27   32   37   42   47
[3,]   28   33   38   43   48
[4,]   29   34   39   44   49
[5,]   30   35   40   45   50
```

* Dadas las matrices m1 y m2 usa rbind() y cbind() para crear matrices nuevas utilizando estas funciones, llamalas M1 y M2. ¿En que se diferencian las matrices creadas?

```
m1 <- matrix(1, nr = 2, nc = 2)
m2 <- matrix(2, nr = 2, nc = 2)
```

```
> m1 = matrix(1, nrow=2, ncol=2); m1
     [,1] [,2]
[1,]    1    1
[2,]    1    1

> m2 = matrix(2, nrow=2, ncol=2); m2
     [,1] [,2]
[1,]    2    2
[2,]    2    2

> M1 = rbind(m1,m2); M1
     [,1] [,2]
[1,]    1    1
[2,]    1    1
[3,]    2    2
[4,]    2    2

> M2 = cbind(m1,m2); M2
     [,1] [,2] [,3] [,4]
[1,]    1    1    2    2
```

```
[2,]    1    1    2    2
```

La diferencia entre la función cbind() y rbind() es como concatena las matrices, vectores, etc. La función cbind() concatena los objetos seleccionados por columnas, es decir, crea una matriz en la que primero están las columnas de la primera matriz y después las columnas de la segunda matriz; en cambio, la función rbind() concactena las filas de ambas matrices, primero las filas de la primera matriz y después las filas de la segunda matrix.

* El operador para el producto de dos matrices es ‘ %* %’. Por ejemplo, considerando las dos matrices creadas en el ejercicio anterior utilízalo.

```
> M12 = M1 %*% M2; M12
     [,1] [,2] [,3] [,4]
[1,]    2    2    4    4
[2,]    2    2    4    4
[3,]    4    4    8    8
[4,]    4    4    8    8
```

* Usa la matriz M1 del ejercicio anterior y aplica la función t(). ¿qué hace esa función?

```
> t_M1 = t(M1); t_M1
     [,1] [,2] [,3] [,4]
[1,]    1    1    2    2
[2,]    1    1    2    2
```

La función t() calcula la traspuesta de la matriz seleccionada.

* Ejecuta los siguientes comandos basados en la función diag() sobre las matrices creadas anteriormente m1 y m2. ¿Qué tipo de acciones puedes ejecutar con ella?

La función diag() calcula la diagonal de una matriz, o construye una matriz diagonal. Dependiendo de la información que se le pase a la función, esta realiza diferentes funciones. Si se le pasa una matriz, la función extrae la diagonal de dicha matriz. Si se le pasa un entero, la función devuelve la matriz identidad de dimesiones de dicho número entero. Si se le pasa un vector, la matriz devuelve una matriz donde la diagonal son los elementos del vector. También se puede cambiar el valor de la diagonal asignandole un valor nuevo a dicha diagonal.

```
> diag(m1)
> diag(m1)
[1] 1 1
```

```
> diag(rbind(m1, m2) %*% cbind(m1, m²))
> diag(rbind(m1,m2) %*% cbind(m1,m2))
[1] 2 2 8 8
```

```
> diag(m1) <- 10
> diag(m1) = 10; diag(m1)
[1] 10 10
```

```
> diag(3)
> diag(3)
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
```

```
[3,]    0    0    1
```

```
> v <- c(10, 20, 30)
> diag(v)
> v=c(10,20,30)
> diag(v)
     [,1] [,2] [,3]
[1,]   10    0    0
[2,]    0   20    0
[3,]    0    0   30
```

```
> diag(2.1, nr = 3, nc = 5)
> diag(2.1, nrow=3, ncol=5)
     [,1] [,2] [,3] [,4] [,5]
[1,]  2.1  0.0  0.0    0    0
[2,]  0.0  2.1  0.0    0    0
[3,]  0.0  0.0  2.1    0    0
```

* Ordena la matriz `x <- matrix(1:100, ncol=10):`

a. en orden descendente por su segunda columna y asigna el resultado a una nueva matrix x1. Pista: función order()

```
> x1 = x
> x1 = x[order(x[,2],decreasing = T),]
> x1
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]   10   20   30   40   50   60   70   80   90   100
 [2,]    9   19   29   39   49   59   69   79   89    99
 [3,]    8   18   28   38   48   58   68   78   88    98
 [4,]    7   17   27   37   47   57   67   77   87    97
 [5,]    6   16   26   36   46   56   66   76   86    96
 [6,]    5   15   25   35   45   55   65   75   85    95
 [7,]    4   14   24   34   44   54   64   74   84    94
 [8,]    3   13   23   33   43   53   63   73   83    93
 [9,]    2   12   22   32   42   52   62   72   82    92
[10,]    1   11   21   31   41   51   61   71   81    91
```

b. en orden descendente por su segunda fila y asigna el resultado a una nueva matrix x2

```
> x2 = x
> x2 = x[order(x[2,],decreasing = T),]
> x2
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]   10   20   30   40   50   60   70   80   90   100
 [2,]    9   19   29   39   49   59   69   79   89    99
 [3,]    8   18   28   38   48   58   68   78   88    98
 [4,]    7   17   27   37   47   57   67   77   87    97
 [5,]    6   16   26   36   46   56   66   76   86    96
 [6,]    5   15   25   35   45   55   65   75   85    95
 [7,]    4   14   24   34   44   54   64   74   84    94
 [8,]    3   13   23   33   43   53   63   73   83    93
 [9,]    2   12   22   32   42   52   62   72   82    92
[10,]    1   11   21   31   41   51   61   71   81    91
```

c. Ordena solo la primera columna de x de forma descendente

```
> x[,1] = x[order(x[,1],decreasing = T),1]
> x
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]   10   11   21   31   41   51   61   71   81    91
 [2,]    9   12   22   32   42   52   62   72   82    92
 [3,]    8   13   23   33   43   53   63   73   83    93
 [4,]    7   14   24   34   44   54   64   74   84    94
 [5,]    6   15   25   35   45   55   65   75   85    95
 [6,]    5   16   26   36   46   56   66   76   86    96
 [7,]    4   17   27   37   47   57   67   77   87    97
 [8,]    3   18   28   38   48   58   68   78   88    98
 [9,]    2   19   29   39   49   59   69   79   89    99
[10,]    1   20   30   40   50   60   70   80   90   100
```

* Crea los siguientes vectores:

```
# Box office Star Wars: In Millions (!)  First element: US, Second element:
# Non-US
new_hope = c(460.998007, 314.4)
empire_strikes = c(290.475067, 247.9)
return_jedi = c(309.306177, 165.8)
```

Los datos se corresponden con las ventas en millones de la trilogía de la guerra de las galaxias. El primer numero corresponde a las ventas en US y el segundo al resto de países.

a) Construye la matriz star_wars_matrix con esos vectores

```
> star_wars_matrix = matrix(rbind(new_hope,empire_strikes,return_jedi),
ncol=2); star_wars_matrix
          [,1]  [,2]
[1,] 460.9980 314.4
[2,] 290.4751 247.9
[3,] 309.3062 165.8
```

b) Añádele nombres a las columnas y filas de la matriz según las descripciones dadas anteriormente de los datos

```
> colnames(star_wars_matrix)=c('US','Non-US')

> rownames(star_wars_matrix)=c('new_hope','empire_strikes','return_jedi')

> star_wars_matrix
                      US Non-US
new_hope       460.9980  314.4
empire_strikes 290.4751  247.9
return_jedi    309.3062  165.8
```

c) Calcula las ganacias mundiales de cada película y guardalas en un vector que se llame `worldwide_vector.`

```
>
worldwide_vector=c(sum(star_wars_matrix[1,]),sum(star_wars_matrix[2,]),sum(star
_wars_matrix[3,]))

> worldwide_vector
[1] 775.3980 538.3751 475.1062
```

d) Añade éste ultimo vector como una columna nueva a la matriz `star_wars_matrix` y asigna el resultado a `all_wars_matrix.` Usa para ello la función `cbind().`

```
> all_wars_matrix = cbind(star_wars_matrix,worldwide_vector)

> colnames(all_wars_matrix)=c('US','Non-US','Total'); all_wars_matrix
                        US Non-US    Total
new_hope       460.9980  314.4 775.3980
empire_strikes 290.4751  247.9 538.3751
return_jedi    309.3062  165.8 475.1062
```

e) Calcula las ganancias totals en USA y fuera de USA para las tres películas. Puedes usar para ello la función `colSums()`

```
> colSums(star_wars_matrix)
       US   Non-US
1060.779  728.100
```

f) Calcula la media de ganancias para todas las películas fuera de los estados unidos. Asigna esa media la variable `non_us_all.`

```
> non_us_all = mean(star_wars_matrix[,2]); non_us_all

[1] 242.7
```

g) Haz lo mismo pero solo par alas dos primeras películas . Asigna el resultado a la variable `non_us_some`.

```
> non_us_some = mean(star_wars_matrix[1:2,2]); non_us_some
[1] 281.15
```

h) Calcula cuantos visitantes hubo para cada película en cada área geográfica. Ya tienes las ganancias totales en `star_wars_matrix`. Asume que el precio de las entradas es de cinco euros/dólares (Nota: el numero total de visitantes para cada pelicula dividido por el precio del ticket te da el numero de visitantes)

```
> tickets_sold_us_1 = star_wars_matrix[1,1]*1000000/5 ; tickets_sold_us_1

[1] 92199601

> ticket_sold_us_2 = star_wars_matrix[2,1]*1000000/5; ticket_sold_us_2
[1] 58095013
```

```
> ticket_sold_us_3 = star_wars_matrix[3,1]*1000000/5; ticket_sold_us_3
[1] 61861235

> tickets_sold_non_us_1 = star_wars_matrix[1,2]*1000000/5;
tickets_sold_non_us_1
[1] 62880000

> ticket_sold_non_us_2 = star_wars_matrix[2,2]*1000000/5; ticket_sold_non_us_2
[1] 49580000

> ticket_sold_non_us_3 = star_wars_matrix[3,2]*1000000/5; ticket_sold_non_us_3
[1] 33160000
```

i) Calcula la media de visitantes en territorio USA y en territorio noUS.

```
> mean_visitors_us =
mean(c(ticket_sold_us_2,tickets_sold_us_1,ticket_sold_us_3)); mean_visitors_us
[1] 70718617

> mean_visitors_non_us =
mean(c(ticket_sold_non_us_2,tickets_sold_non_us_1,ticket_sold_non_us_3));
mean_visitors_non_us
[1] 48540000
```

2. Subsetting matrices y arrays

* Como hemos visto en teoría la sintásis para acceder tanto a matrices como a arrays bidimiensionales es la siguiente.

```
array[rows, columns]
```

Muchas funciones de R necesitan una matriz como dato de entrada. Si algo no funciona recuerda convertir el objeto a una matriz con la función `as.matrix(iris)`

* Crea un array `i <- array(c(1:10),dim=c(5,2))`. ¿Que información te dan los siguientes comandos?

```
dim(i);
nrow(i);
ncol(i)
```

La función dim() devuelve las dimensiones del array, primero devuelve las filas, después devuelve las columnas, si hay más dimensiones las devuelve también después de estas dos. La función nrow() devuelve las filas de la matriz. La función ncol() devuelve las columnas de la matriz.

* Crea un array de dimensiones 5 filas y dos columnas y rellénalo con valores del 1-5 y del 5 al 1

```
> x <- array(c(1:5,5:1),dim=c(5,2))
> x
     [,1] [,2]
[1,]    1    5
[2,]    2    4
[3,]    3    3
[4,]    4    2
[5,]    5    1
```

* ¿Qué hace el comando `x[i]` ? Comprueba que tienes en x antes

```
> x
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1   11   21   31   41   51   61   71   81    91
 [2,]    2   12   22   32   42   52   62   72   82    92
 [3,]    3   13   23   33   43   53   63   73   83    93
 [4,]    4   14   24   34   44   54   64   74   84    94
 [5,]    5   15   25   35   45   55   65   75   85    95
 [6,]    6   16   26   36   46   56   66   76   86    96
 [7,]    7   17   27   37   47   57   67   77   87    97
 [8,]    8   18   28   38   48   58   68   78   88    98
 [9,]    9   19   29   39   49   59   69   79   89    99
[10,]   10   20   30   40   50   60   70   80   90   100

> x[i]
[1] 51 62 73 84 95
```

Devuelve los elementos dentro de la matriz indicada por los datos del array 'i', los cuales toman la primera columna de 'i' para indicar la fila y la segunda columna de 'i' para indicar la columna.

* ¿y el comando `x[i] <- 0`?

```
> x[i] = 10
> x
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]   10   11   21   31   41   10   61   71   81    91
 [2,]    9   12   22   32   42   52   10   72   82    92
 [3,]    8   13   23   33   43   53   63   10   83    93
 [4,]    7   14   24   34   44   54   64   74   10    94
 [5,]    6   15   25   35   45   55   65   75   85    10
 [6,]    5   16   26   36   46   56   66   76   86    96
 [7,]    4   17   27   37   47   57   67   77   87    97
 [8,]    3   18   28   38   48   58   68   78   88    98
 [9,]    2   19   29   39   49   59   69   79   89    99
[10,]    1   20   30   40   50   60   70   80   90   100
```

El comando asigna el valor 10 a las posiciones definidas por el array 'i'.

* Descárgate el fichero array_datos.txt de PRADO (Datos/) e impórtalo en tu work space de R teniendo en cuenta que es un texto tabulado. Después crea un documento con los mismos datos pero en formato csv en vez de tab separated.

```
> arr_tab = read.delim('array_datos.txt',header = T)

> str(arr_tab)
'data.frame':   3 obs. of  3 variables:
 $ edad  : int  20 22 19
 $ peso  : int  65 70 68
 $ altura: int  174 180 170

> write.csv(arr_tab,'salida_datos.csv')
```

3. Factors

* Dado x = c(1, 2, 3, 3, 5, 3, 2, 4, NA), ¿cuáles son los levels de factor(x)?

a) 1, 2, 3, 4, 5
b) NA
c) 1, 2, 3, 4, 5, NA

```
> x = c(1,2,3,3,5,3,2,4,NA)
> factor(x)
[1] 1    2    3    3    5    3    2    4    <NA>
Levels: 1 2 3 4 5
```

La respuesta correcta sería la a) 1,2,3,4,5.

* Dado x <- c(11, 22, 47, 47, 11, 47, 11) y la ejecución de la sentencia factor(x, levels=c(11, 22, 47), ordered=TRUE) ¿cuál es el cuarto elemento de la salida?

a. 11
b. 22
c. 47

```
> x <- c(11, 22, 47, 47, 11, 47, 11)
> factor(x,levels=c(11,22,47),ordered = T)
[1] 11 22 47 47 11 47 11
Levels: 11 < 22 < 47
```

El cuarto elemento de la salida es 47.

* Para el factor z <- c("p", "a" , "g", "t", "b"), reemplaza el tercer elemento de z por "b".

a. factor(z[3]) <- "b"
b. levels(z[3]) <- "b"
c. z[3] <- "b"

```
> z <- c("p", "a" , "g", "t", "b")
> z[3] <- 'b'; z
[1] "p" "a" "b" "t" "b"
```

La respuesta correcta es la c. c) z[3]<-'b'

* Dado z <- factor(c("p", "q", "p", "r", "q")) escribe una expresión de R que cambie el level "p" a "w"

* Usa el dataset “iris”

- escribe la expresión necesaria para convertir la variable “Sepal.Length” en un factor con cinco niveles (levels) . Pista(mira la función table() y la función cut().

- escribe la expresión necesaria para generar una tabla de frecuencias con dos filas y tres columnas . Las filas deben referirse a si la variable “Sepal.length” es menor que 5 y las columnas a las diferentes expecies. El resultado debe ser:

```
       setosa versicolor virginica
FALSE      30         49        49
TRUE       20          1         1
```

* El factor responses se define como:

```
responses <- factor(c("Agree", "Agree", "Strongly Agree",
"Disagree", "Agree")),
```

sin embargo nos damos cuenta que tiene un nuevo nivel, "Strongly Disagree", que no estaba presente cuando se creó. Añade el nuevo nivel al factor y conviértelo en un factor ordenado de la siguiente forma:

```
Levels: Strongly Agree < Agree < Disagree < Strongly
Disagree
```

* Dado el factor:

```
x <- factor(c("high", "low", "medium", "high", "high",
"low", "medium"))
```

escribe la expresión en R que permita dar valores numéricos únicos para los distintos niveles (levels) de x según el siguiente esquema:

```
level high   => value 1
level low    => value 2
level medium => value 3
```

Pista: investiga la función unique() y los parámetros de data.frame()

4. Acceso y selección de secciones de un data frames

La sintaxis general para acceder a un data frame es
`my_frame[rows, columns]`

* Vamos a trabajar con un ejemplo que viene por defecto en la instalación de R USArrests. Este data frame contiene la información para cada estado Americano de las tasas de criminales (por 100.000 habitantes). Los datos de las columnas se refieren a Asesinatos, violaciones y porcentaje de la población que vive en áreas urbanas. Los datos son de 1973. Contesta a las siguientes preguntas sobre los datos

- Las dimensiones del dataframe
- La longitud del dataframe (filas o columnas)
- Numero de columnas

- ¿Cómo calcularías el número de filas?

- Obtén el nombre de las filas y las columnas para este data frame

- échale un vistazo a los datos, por ejemplo a las seis primeras filas

- Ordena de forma decreciente las filas de nuestro data frame según el porcentaje de población en el área urbana. Para ello investiga la función order () y sus parámetros.

- ¿Podrías añadir un segundo criterio de orden?, ¿cómo?

- Muestra por pantalla la columna con los datos de asesinato

- Muestra las tasas de asesinato para el segundo, tercer y cuarto estado

- Muestra las primeras cinco filas de todas las columnas

- Muestra todas las filas para las dos primeras columnas

- Muestra todas las filas de las `columnas` 1 y 3
- Muestra solo las primeras cinco filas de las columnas 1 y 2
- Extrae las filas para el índice Murder

Vamos con expresiones un poco mas complicadas:…

-¿Que estado tiene la menor tasa de asesinatos? ¿qué línea contiene esa información?, obtén esa informaciónn

¿Que estados tienen una tasa inferior al 4%?, obtén esa informaciónn

¿Que estados estan en el cuartil superior (75) en lo que a poblacion en zonas urbanas se refiere?

* Vamos a trabajar con otro dataframe. Descarga el fichero student.txt de la plataforma PRADO, almacena la información en una variable llamada “students”. Ten en cuenta que los datos son tab-delimited y tienen un texto para cada columna. Comprueba que R ha leído correctamente el fichero imprimiendo el objeto en la pantalla

```
> students
```

-Imprime solo los nombres de la columnas

-Llama a la columna height solo

-¿Cuantas observaciones hay en cada grupo?. Utiliza la función table(). Este commando se puede utilizar para crear tablas cruzadas (cross-tabulations)

-Crea nuevas variables a partir de los datos que tenemos. Vamos a crear una variable nueva “sym” que contenga M si el genero es masculino y F si el genero es femenino. Busca en la ayuda información sobre la función ifelse(). Crea una segunda variable “colours” cuyo valor será “Blue” si el estudiante es de kuopio y “Red” si es de otro sitio.

- Con los datos anteriores de height y shoesize y las nuevas variables crea un nuevo data.frame que se llame students.new

- Comprueba que la clase de student.new es un dataframe

- Crea dos subsets a partir del dataset student. Dividelo dependiendo del sexo. Para ello primero comprueba que estudiantes son hombres (male). Pista: busca información sobre la función which.

-Basándote en esa selección dada por which() toma solo esas filas del dataset student para generar el subset stundent.male

- Repite el procedimiento para seleccionar las estudiantes mujeres (females)

- Utiliza la function write.table() para guarder el contenido de student.new en un archivo.

* Accede al dataset “women”.

- Primero confirma que los datos están ordenados de forma creciente según la altura (height) y el peso (weight) sin mirar los datos
- Crea una nueva variable “bmi”. Este valor responde a la siguiente fórmula: BMI = (Weight in Pounds / (Height in inches) x (Height in inches)) x 703
- Ordena el dataframe por el valor de bmi y lpor orden alfabético de la variable name

  investiga las funciones is.unsorted(), sort() and order()