Introducción a R

Email: delval@decsai.ugr.es

Carmen Biedma Rodriguez

1. Matrices

* Ejecuta los siguientes comandos.

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

El primer comando genera una matriz de 2 filas (nr=2) y dos columnas (nc=2) en la que todas las celdas contienen el valor 5.

```
matrix(1:6, 2, 3)
```

En el segundo creamos una matriz 2x3 con los valores del 1 al 6. Como no hemos especificado nada, la matriz se rellenará por columnas de izquierda a derecha y de arriba a abajo, por lo que obtendremos lo siguiente:

```
> matrix(1:6, 2, 3)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

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

En éste último caso los valores y el tamaño del a matriz son los mismos que en el caso anterior, la diferencia es que especificamos que queremos que se rellene por filas, por lo que el resultado será el siguiente.

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

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

```
> m <- matrix(z,nr=3,nc=10)
     [,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

```
> vector <- m[,3]
> vector
[1] 7 8 9
```

* Create in R the matrices

```
x =3  21
-1   1
```

```
> x <- matrix(c(3,-1,21,1),nr=2,nc=2)
     [,1] [,2]
[1,]    3   21
[2,]   -1    1
```

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

```
   ➢  x <- matrix(c(1,0,4,1,0,-1),nr=2,nc=3)
     [,1] [,2] [,3]
[1,]    1    4    0
[2,]    0    1   -1
```

Y calcula los efectos de los siguientes comandos

```
(a) x[1,]
(b) x[2,]
(c) x[,2]
(d) y[1,2]
(e) y[,2:3]
```

Los comandos a) y b) devuelven las filas 1 y 2 completas respectivamente. El comando que vemos en el apartado c) devuelve la columna 2 completa. El comando d) devuelve el elemento en la fila 1 columna 2. El último comando que vemos nos devuelve las columnas 2 y 3 completas.

* 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() nos devuelve la dimensión del objeto que le pasamos como argumento. La función que necesitamos para crear un array multidimensional es array(). Los argumentos que necesita ésta función son: los elementos que queremos que almacene (en éste caso la matriz creada anteriormente) y la dimensión del array (expresada como un vector de enteros si queremos que sea de más de una dimensión).

```
> array(x,c(2,3))
     [,1] [,2] [,3]
[1,]    1    4    0
[2,]    0    1   -1
```

* Crea un array de 5 x 5 y rellénalo con valores del 1 al 25. Investiga la función array(). Llama al array x

```
> y = array(1:25,c(5,5))
> y
     [,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
```

* Dadas las matrices m1 y m2 usa rbind() ycbind() 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)
```

Las función rbind() se utiliza para añadir filas a una matriz, hay que especificar la matriz original y el valor que queremos que tengan los elementos del a nueva matriz.

```
> M1= rbind(m1,3)
> M1
     [,1] [,2]
[1,]    1    1
[2,]    1    1
[3,]    3    3
```

La función cbind() añade una columna a la matriz, el uso es el mismo que el de rbind y el resultado es el siguiente:

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

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

```
> m1%*%m2
     [,1] [,2]
[1,]    4    4
[2,]    4    4
```

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

Genera la traspuesta de la matriz que pasamos como argumento.

```
> t(M1)
     [,1] [,2] [,3]
[1,]    1    1    3
[2,]    1    1    3
```

* 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?

Ésta función sirve para trabajar con la diagonal de una matriz. Por ejemplo, podemos obtener el valor del a misma pasándole como argumento una matriz (como en el caso de los dos primeros comandos). Además, podemos cambiar dichas componentes de la matriz asignando un nuevo valor a diag(matriz). Por ejemplo en el tercer comando de los ejemplos asigna el número 10 a todas las componentes de la diagonal de m1.

Por otra parte, sirve para crear matrices identidad del tamaño que le pasamos como argumento. En comando diag(3) crea una matriz identidad 3x3. Además, si le pasamos como argumento un vector, creará una matriz con todos los valores a 0 excepto los de la diagonal, que tomarán el valor de los elementos del vector.

Otra aplicación curiosa que tiene éste comando es la que vemos en el último comando del ejemplo. En este caso crea una matriz con 3 filas y 5 columnas, y asigna en diagonal (empezando por el elemento [1,1]) el valor que le hayamos especificado en el primer argumento (2.1 en éste caso).

```
>diag(m1)

>diag(rbind(m1, m2) %*% cbind(m1, m2))

>diag(m1) <- 10

>diag(3)

> v <- c(10, 20, 30)
>diag(v)

>diag(2.1, nr = 3, nc = 5)
```

* 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[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

c. Ordena solo la primeracolumna de x de forma descendente

* Crea los siguientes vectores:

```
# Box office StarWars: In Millions (!)  Firstelement: US, Secondelement:
# 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

```
> stars_wars_matrix = rbind(new_hope,empire_strikes,return_jedi)
> stars_wars_matrix
                   [,1]  [,2]
new_hope       460.9980 314.4
empire_strikes 290.4751 247.9
return_jedi    309.3062 165.8
```

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

```
> colnames(stars_wars_matrix) <- c("USA","Resto")
```

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

```
> worldwide_vector <- c(sum(stars_wars_matrix[1,]),sum(stars_wars_ma
trix[2,]),sum(stars_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(stars_wars_matrix,worldwide_vector)
> all_wars_matrix
                   USA Resto worldwide_vector
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(stars_wars_matrix)
     USA    Resto
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.`

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

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)

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

2. Subsetting matrices yarrays

* Como hemos visto en teoría la sintásis para acceder tanto a matrices como a arraysbidimiensionales 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?

Nos da la dimensión del array, primero las filas y luego las columnas. Nrow proporciona el número de filas y ncol el número de columnas.

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

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

```
> a <- array(c(1:5,5:1),dim=c(5,2))
> a
     [,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

Este comando devuelve lo que hay en las posiciones correspondientes al vector i en la matriz x. Es decir, la primera fila de la matriz i tiene 2 valores, por lo que tomará el primer valor (1) como la fila y el segundo (6) como la columna. Lo que devuelve es el valor 51 correspondiente a la fila 1 columna 6, y así sucesivamente.

```
> 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
> i
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10


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

* ¿y el comando `x[i] <- 0`?

En este caso escribe un 0 en todas las posiciones que especifique el vector i (de la misma forma que lo hacía en el apartado anterior).

* Descárgate el fichero array_datos.txt de PRADO (Datos/) e impórtalo en tu workspace 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 tabseparated.

3. Factors

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

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

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

* 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?

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

a. 11
b. 22
c. 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"

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

```
> levels(z)[1] <- "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().

```
> f <-factor(table(cut(iris$Sepal.Length,5)))
> f
(4.3,5.02] (5.02,5.74] (5.74,6.46] (6.46,7.18]  (7.18,7.9]
        32          41          42          24          11
Levels: 11 24 32 41 42
```

- 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:

```
setosaversicolorvirginica
  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,"StronglyDisagree", 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
```

```
> levels(responses) <- c(levels(responses),"Strongly Disagree")
> responses
[1] Agree          Agree          Strongly Agree Disagree       Agre
e
Levels: Agree Disagree Strongly Agree Strongly Disagree
```

```
> ordered.responses <- factor(responses[],c("Strongly Agree","Agree"
,"Disagree","Strongly Disagree"),ordered = TRUE)
> ordered.responses
[1] Agree          Agree          Strongly Agree Disagree       Agre
e
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:

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

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

```
> data.frame(levels = unique(x), value = as.numeric(unique(x)))
  levels value
1   high     1
2    low     2
3 medium     3
```

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

Tiene 50 filas y 4 columnas.

```
> dim.data.frame(USArrests)
[1] 50  4
```

- La longitud del dataframe (filas o columnas)

Con el comando length obtenemos que tiene longitud 4.

- Numero de columnas

```
> ncol(USArrests)
[1] 4
```

- ¿Cómo calcularías el número de filas?

Con el comando nrow()

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

```
> row.names(USArrests)
 [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
     "California"     "Colorado"
 [7] "Connecticut"    "Delaware"       "Florida"        "Georgia"
     "Hawaii"         "Idaho"
[13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
     "Kentucky"       "Louisiana"
[19] "Maine"          "Maryland"       "Massachusetts"  "Michigan"
     "Minnesota"      "Mississippi"
[25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
     "New Hampshire"  "New Jersey"
[31] "New Mexico"     "New York"       "North Carolina" "North Dakot
a"   "Ohio"           "Oklahoma"
[37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carol
ina" "South Dakota"   "Tennessee"
[43] "Texas"          "Utah"           "Vermont"        "Virginia"
     "Washington"     "West Virginia"
[49] "Wisconsin"      "Wyoming"
> colnames(USArrests)
[1] "Murder"   "Assault"  "UrbanPop" "Rape"
```

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

```
> USArrests[1:6,]
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
Arkansas      8.8     190       50 19.5
California    9.0     276       91 40.6
Colorado      7.9     204       78 38.7
```

- 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.

```
> USArrests.ordered <- USArrests[c(order(USArrests[,3],decreasing =
TRUE)),]
```

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

Ordenando de nuevo el vector que nos da order() y poniendo un segundo criterio.

- Muestra por pantalla la columna con los datos de asesinato

```
> USArrests$Murder
 [1] 13.2 10.0  8.1  8.8  9.0  7.9  3.3  5.9 15.4 17.4  5.3  2.6 10.
4  7.2  2.2  6.0  9.7 15.4  2.1 11.3
[21]  4.4 12.1  2.7 16.1  9.0  6.0  4.3 12.2  2.1  7.4 11.4 11.1 13.
0  0.8  7.3  6.6  4.9  6.3  3.4 14.4
[41]  3.8 13.2 12.7  3.2  2.2  8.5  4.0  5.7  2.6  6.8
```

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

```
> USArrests[c(2,3,4),2]
[1] 263 294 190
```

- Muestra las primeras cinco filas de todas las columnas

```
> USArrests[1:5,]
```

- Muestra todas las filas para las dos primeras columnas

```
> USArrests[,c(1,2)]
```

- Muestra todas las filas de las columnas 1 y 3

```
> USArrests[,c(1,3)]
```

- Muestra solo las primeras cinco filas de las columnas 1 y 2

```
> USArrests[c(1:5),c(1,2)]
```

- Extrae las filas para el índice Murder

```
> USArrests[,"Murder"]
```

Vamos con expresiones un poco mas complicadas:…

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

```
> which(USArrests == min(USArrests[,"Murder"]))
[1] 34
```

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

```
> row.names(USArrests[which(USArrests < 4),])
 [1] "Connecticut"   "Idaho"         "Iowa"          "Maine"
"Minnesota"     "New Hampshire"
 [7] "North Dakota"  "Rhode Island"  "South Dakota"  "Utah"
"Vermont"       "Wisconsin"
```

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

```
> summary(USArrests)
     Murder          Assault         UrbanPop          Rape
 Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30
 1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07
 Median : 7.250   Median :159.0   Median :66.00   Median :20.10
 Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23
 3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18
 Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
```

* 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 <- read.delim("student.txt")
```

```
      >students
```

-Imprime solo los nombres de la columnas

```
> names(students)
[1] "height"     "shoesize"   "gender"     "population"
```

–Llama a la columna height solo

```
> students["height"]
 [1] 181 160 174 170 172 165 161 167 164 166 162 158 175 181 180 177
 173
```

-¿Cuantas observaciones hay en cada grupo?. Utiliza la funcióntable(). 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.

```
> sexo <- c(ifelse(students$gender == "male","M","F"))
> sexo
 [1] "M" "F" "F" "M" "M" "F" "F" "F" "F" "F" "F" "F" "M" "M" "M" "M"
 "M"
> colours <- ifelse(students$population == "kuopio","Blue","Red")
> colours
 [1] "Blue" "Blue" "Blue" "Blue" "Blue" "Blue" "Blue" "Red"  "Red"
"Red"  "Red"  "Red"  "Red"  "Red"

[15] "Red"  "Red"  "Red"
```

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

```
> students.new <- data.frame(students$height,students$shoesize,sym,c
olours)
```

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

```
> class(students.new)

[1] "data.frame"
```

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

Tenemos 2 formas de hacer este ejercicio.

```
> students.male <- subset(students,students$gender == "male")

> students.male <- students[which(students$gender == "male"),]
```

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

```
> students.male <- subset(students,students$gender == "male")

> students.male <- students[which(students$gender == "male"),]
```

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

```
> students.female <- subset(students,students$gender == "female")
```

```
> students.female <- students[which(students$gender == "female"),]
```

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

```
> write.table(students.new,file="studentsnew.txt")
```