

# Introducción al software estadístico

## Módulo IV

Nicolás Schmidt

`nschmidt@cienciassociales.edu.uy`

Departamento de Ciencia Política  
Unidad de Métodos y Acceso a Datos  
Facultad de Ciencias Sociales  
Universidad de la República

# Estructura de la presentación

## 1 Matrices

- Creación de matrices
- Indexación
- `rbind()` y `cbind()`
- Funciones matriciales básicas
- Funciones básicas con matrices
- `apply()`
- `matplot()`

# Estructura de la presentación

## 1 Matrices

- Creación de matrices
- Indexación
- `rbind()` y `cbind()`
- Funciones matriciales básicas
- Funciones básicas con matrices
- `apply()`
- `matplot()`

## Matriz y matrix()

$$M = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ x_{31} & x_{32} & x_{33} & \dots & x_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{d1} & x_{d2} & x_{d3} & \dots & x_{dn} \end{bmatrix} \quad M = \begin{bmatrix} 1 & 6 & 11 & 16 & 21 \\ 2 & 7 & 12 & 17 & 22 \\ 3 & 8 & 13 & 18 & 23 \\ 4 & 9 & 14 & 19 & 24 \\ 5 & 10 & 15 & 20 & 25 \end{bmatrix}$$

Ejemplo:

```
matrix(data = 1:25, nrow = 5, ncol = 5)
```

```
##      [,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
```

# Definición

Una matriz es un vector atómico con el atributo (`attributes()`) `dim()`. Las matrices son vectores con dos dimensiones: filas (`nrow()`) y columnas (`ncol()`).

## Ejemplo:

```
vector.m <- 1:4
m <- matrix(vector.m)
m

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

dim(m)

## [1] 4 1
```

## dim()

La función `dim()` sirve para consultar la dimensión de un objeto o para crear una matriz a partir de un vector atómico. En ambos casos, la entrada, como la salida de la función (en el caso de una matriz) es un vector de dos números enteros que corresponden a la cantidad de filas y a la cantidad de columnas.

### Ejemplo:

```
length(m) == dim(m)[1]*dim(m)[2]  
## [1] TRUE
```

### Modificando la dimensión de la matriz `m`

```
dim(m) <- c(2, 2)  
m  
##      [,1] [,2]  
## [1,]    1    3  
## [2,]    2    4
```

## matrix()

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

data	Vector de datos. Si no se ingresan datos se genera una matriz de NA's
nrow	Número entero que indica la cantidad de filas.
ncol	Número entero que indica la cantidad de columnas.
byrow	Valor lógico que indica si la matriz se completa por columnas o por filas.
dimnames	Lista de longitud 2 que contiene los nombres de filas y columnas.

- Si en el argumento data se ingresa un vector de longitud 1, pero la dimensión de la matriz es mayor a `c(1,1)` R completa la totalidad de la matriz repitiendo los valores (**reciclaje!**).

## Ejemplos:

```
matrix(1,2,2)                                # matriz con un solo valor de entrada

##      [,1] [,2]
## [1,]    1    1
## [2,]    1    1

matrix(nrow = 2, ncol = 2)                   # matriz sin vector de entrada

##      [,1] [,2]
## [1,]   NA   NA
## [2,]   NA   NA

matrix(1:4, 2, 2)                             # matriz con vector de entrada por columna

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

matrix(1:4, 2, 2, byrow = TRUE)               # matriz con vector de entrada por filas

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



## dimnames()

La función (y el argumento) `dimnames()` es para incorporar nombres a las filas y columnas de la matriz. Adicionalmente, también están las funciones `rownames()` y `colnames()`.

```
matrix(1:4, 2, 2, dimnames = list(c("Fila 1", "Fila 2"),  
                                   c("Columna 1", "Columna 2")))
```

```
##           Columna 1 Columna 2  
## Fila 1           1          3  
## Fila 2           2          4
```

```
dimnames(m) <- list(c("Fila 1", "Fila 2"), c("Columna 1", "Columna 2")); m
```

```
##           Columna 1 Columna 2  
## Fila 1           1          3  
## Fila 2           2          4
```

```
m2 <- matrix(c(100, 200), 1, 2)  
colnames(m2) <- c("Columna 1", "Columna 2")  
rownames(m2) <- "Fila 1"; m2
```

```
##           Columna 1 Columna 2  
## Fila 1          100         200
```

# Indexación

Al igual que la indexación con vectores, vamos a usar los corchetes '['. Pero como las matrices tienen el atributo `dim()` (dimensiones) hay que especificar si queremos indexar por filas, por columnas o por ambas.

```
nombre_de_la_matriz [ número_de_fila , número_de_columna ]
```

Detalles:

- Si dentro de los corchetes no hay una coma, se hará una indexación del vector que compone la matriz. Es la ubicación en el `length()`.
- Si no se indica ningún valor en filas o en columnas R interprete que se refiere a todas las filas o todas las columnas.
- El número de filas y el de columnas son un vector numérico. Esto indica que se puede usar cualquier función que devuelva un vector numérico de números enteros dentro de los corchetes.

## Ejemplos: indexar para seleccionar

```
(m <- matrix(sample(1:100, 12), 4, 3))
```

```
##      [,1] [,2] [,3]
## [1,]  34   2  95
## [2,]  96  89  58
## [3,]  84  94  49
## [4,]  63   1  69
```

```
m[1,2] # fila 1 y columna 2.
```

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

```
m[1:2,] # filas 1 y 2 y todas las columnas
```

```
##      [,1] [,2] [,3]
## [1,]  34   2  95
## [2,]  96  89  58
```

```
m[-c(3:4),] # quito fila 3 y 4
```

```
##      [,1] [,2] [,3]
## [1,]  34   2  95
## [2,]  96  89  58
```

## Ejemplos: indexar usando funciones

```
m
```

```
##      [,1] [,2] [,3]  
## [1,]  34   2  95  
## [2,]  96  89  58  
## [3,]  84  94  49  
## [4,]  63   1  69
```

```
m[3:nrow(m), 2:ncol(m)]
```

```
##      [,1] [,2]  
## [1,]  94  49  
## [2,]   1  69
```

```
m[m[,2] > 50,]
```

```
##      [,1] [,2] [,3]  
## [1,]  96  89  58  
## [2,]  84  94  49
```

## Ejemplos: indexar una matriz con otra matriz.

```
m2 <- matrix(1:20, 4, 5, byrow = TRUE)
```

```
m2
```

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

```
i <- matrix(c(1:4,5:2), 4 , 2)
```

```
i
```

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

```
m2[i]
```

```
## [1]  5  9 13 17
```

## Ejemplos: completar una matriz indexando valores de distintas maneras.

```
mat <- matrix(nrow = 4, ncol = 3)
mat

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

mat[1,1:3] <- 10:12      # usando ":"
mat[2,1:3] <- 20         # usando ":" y un solo valor (que se recicla)
mat[3,1:ncol(mat)] <- 30:32 # usando la función ncol()
j <- 1:3                # creo un vector de las columnas de interés
mat[4,j] <- 40:42       # indexo con el objeto "j"
mat

##      [,1] [,2] [,3]
## [1,]   10   11   12
## [2,]   20   20   20
## [3,]   30   31   32
## [4,]   40   41   42
```

## Ejemplos: indexar por los nombres de las filas o columnas.

```
M <- matrix(1:4, 2, 2, dimnames=list(c("Fila 1", "Fila 2"), c("Col. 1", "Col. 2")))
M

##           Col. 1 Col. 2
## Fila 1         1     3
## Fila 2         2     4

M["Fila 1", "Col. 2"]

## [1] 3

M["Fila 1", 2]

## [1] 3

M["Fila 1", c("Col. 1", "Col. 2")]

## Col. 1 Col. 2
##      1     3

M[-c(rownames(M)=="Fila 1"),]

## Col. 1 Col. 2
##      2     4
```

**Recordar:** una matriz es una estructura de datos homogénea. Es decir, solo puede contener datos de un único tipo.

### Ejemplos:

```
M <- matrix(1:4, 2, 2, dimnames=list(c("Fila 1", "Fila 2"), c("Col. 1", "Col. 2")))
M
```

```
##           Col. 1 Col. 2
## Fila 1      1      3
## Fila 2      2      4
```

```
typeof(M)
```

```
## [1] "integer"
```

```
M[1,1] <- "1"           # Solo cambio un dato de los cuatro que tiene la matriz
M
```

```
##           Col. 1 Col. 2
## Fila 1    "1"    "3"
## Fila 2    "2"    "4"
```

```
typeof(M)
```

```
## [1] "character"
```



# Función drop()

Si alguna de las dimensiones en la extracción va a tener una longitud de 1 se elimina el atributo `dim()`.

Ejemplo.

```
j <- matrix(1:12, 3, 4)
j[1,]

## [1] 1 4 7 10

j[1,,drop = FALSE]

##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7    10

class(j[1,])

## [1] "integer"

class(j[1,,drop = FALSE])

## [1] "matrix"
```

## rbind() y cbind()

Una manera alternativa de crear funciones es *pegando* vectores. Los vectores se pueden pegar por filas (`rbind()`) o por columnas (`cbind()`). Estas funciones también permiten *pegar* matrices o un vector a una matriz.

Ejemplo:

```
vec1 <- 1:5
vec2 <- 6:10
vec3 <- 11:15

(mat <- rbind(vec1, vec2, vec3))
```

##	[,1]	[,2]	[,3]	[,4]	[,5]
## vec1	1	2	3	4	5
## vec2	6	7	8	9	10
## vec3	11	12	13	14	15

## Ejemplos con rbind() y cbind()

```
(mat2 <- cbind(vec1, vec2, vec3))
```

```
##      vec1 vec2 vec3
## [1,]    1    6   11
## [2,]    2    7   12
## [3,]    3    8   13
## [4,]    4    9   14
## [5,]    5   10   15
```

```
(mat3 <- cbind(mat2, 1, 0))
```

*# el 1 y el 0 se reciclan!*

```
##      vec1 vec2 vec3
## [1,]    1    6   11 1 0
## [2,]    2    7   12 1 0
## [3,]    3    8   13 1 0
## [4,]    4    9   14 1 0
## [5,]    5   10   15 1 0
```

```
colnames(mat3)[4:5] <- c("vec4", "vec5"); mat3
```

```
##      vec1 vec2 vec3 vec4 vec5
## [1,]    1    6   11    1    0
## [2,]    2    7   12    1    0
## [3,]    3    8   13    1    0
## [4,]    4    9   14    1    0
## [5,]    5   10   15    1    0
```

## Ejemplos con rbind() y cbind()

```
mat4 <- cbind(mat2, vec4 = 1, vec5 = 0)
mat4
```

```
##      vec1 vec2 vec3 vec4 vec5
## [1,]    1    6   11    1    0
## [2,]    2    7   12    1    0
## [3,]    3    8   13    1    0
## [4,]    4    9   14    1    0
## [5,]    5   10   15    1    0
```

```
mat5 <- cbind(0, rbind(1, 1:3, 21:23))
colnames(mat5) <- letters[1:ncol(mat5)]
rownames(mat5) <- paste(colnames(mat5)[1:nrow(mat5)], 1:nrow(mat5), sep = "-")
mat5
```

```
##      a  b  c  d
## a-1 0  1  1  1
## b-2 0  1  2  3
## c-3 0 21 22 23
```

## Pregunta:

¿cuál es el problema con esta manera de nombrar filas y columnas?

## Funciones matriciales básicas

Función	Descripción
<code>t(x)</code>	Transpuesta de $x$
<code>det(x)</code>	Determinante de $x$
<code>solve(A, b)</code>	Resuelve la ecuación $Ax = b$ para $x$
<code>solve(x)</code>	Matriz inversa de $x$
<code>eigen(x)</code>	Valores propios y vectores propios de $x$
<code>diag(A)</code>	Crea una matriz identidad de $A^*A$
<code>diag(B)</code>	Elementos de la diagonal de $B$
<code>diag(X)</code>	Crea una matriz diagonal a partir del vector $X$
<code>lower.tri(x)</code>	Matriz triangular inferior de $x$
<code>upper.tri(x)</code>	Matriz triangular superior de $x$

## Ejemplos:

```
diag(3) # devuelve una matriz identidad de 3*3
```

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

```
diag(c(1:4)) # devuelve una matriz con esos valores en la diagonal
```

```
##      [,1] [,2] [,3] [,4]  
## [1,]    1    0    0    0  
## [2,]    0    2    0    0  
## [3,]    0    0    3    0  
## [4,]    0    0    0    4
```

```
diag(matrix(1:20, 5 , 4)) # devuelve un vector con los valores de la diagonal
```

```
## [1]  1  7 13 19
```

## Ejemplos:

```
(mat5 <- matrix(16, 4, 4))
```

```
##      [,1] [,2] [,3] [,4]  
## [1,]  16  16  16  16  
## [2,]  16  16  16  16  
## [3,]  16  16  16  16  
## [4,]  16  16  16  16
```

```
lower.tri(mat5)
```

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

```
upper.tri(mat5)
```

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

## Funciones básicas con matrices

Función	Descripción
<code>colSums(x)</code>	Suma por columna de una matriz
<code>rowSums(x)</code>	Suma por fila de una matriz
<code>colMeans(x)</code>	Media por columna de una matriz
<code>rowMeans(x)</code>	Media por fila de una matriz
<code>ncol(x)</code>	Número de columnas de la matriz
<code>nrow(x)</code>	Número de filas de la matriz
<code>colnames(x)</code>	Nombres de las columnas
<code>rownames(x)</code>	Nombre de las filas



## Ejemplos:

```
set.seed(2130)
mi_matriz <- matrix(sample(1:100, 20), 4, 5)
colnames(mi_matriz) <- LETTERS[1:ncol(mi_matriz)]
rownames(mi_matriz) <- paste("Fila", 1:nrow(mi_matriz))
mi_matriz2 <- mi_matriz
mi_matriz

##           A  B  C  D  E
## Fila 1 69 51 17 68 18
## Fila 2 77 15  4 82 83
## Fila 3 60 41 37 45 56
## Fila 4 98 13 40 10 32

colSums(mi_matriz)

##      A      B      C      D      E
## 304 120  98 205 189

rowSums(mi_matriz)

## Fila 1 Fila 2 Fila 3 Fila 4
##    223    261    239    193
```

## Ejemplos:

```
mi_matriz <- rbind(mi_matriz, Total = colSums(mi_matriz))
mi_matriz
```

```
##           A    B    C    D    E
## Fila 1  69   51   17   68   18
## Fila 2  77   15    4   82   83
## Fila 3  60   41   37   45   56
## Fila 4  98   13   40   10   32
## Total  304  120   98  205  189
```

```
mi_matriz <- cbind(mi_matriz, Total = rowSums(mi_matriz))
mi_matriz
```

```
##           A    B    C    D    E Total
## Fila 1  69   51   17   68   18   223
## Fila 2  77   15    4   82   83   261
## Fila 3  60   41   37   45   56   239
## Fila 4  98   13   40   10   32   193
## Total  304  120   98  205  189   916
```

## Ejemplos:

```
colMeans(mi_matriz)
```

```
##      A      B      C      D      E Total  
## 121.6  48.0  39.2  82.0  75.6 366.4
```

```
mi_matriz <- rbind(mi_matriz, Promedio = colMeans(mi_matriz))  
mi_matriz
```

```
##           A      B      C      D      E Total  
## Fila 1    69.0  51 17.0  68  18.0 223.0  
## Fila 2    77.0  15  4.0  82  83.0 261.0  
## Fila 3    60.0  41 37.0  45  56.0 239.0  
## Fila 4    98.0  13 40.0  10  32.0 193.0  
## Total    304.0 120 98.0 205 189.0 916.0  
## Promedio 121.6  48 39.2  82  75.6 366.4
```

## Ejemplos:

```
addmargins(mi_matriz2)
```

```
##           A    B    C    D    E Sum
## Fila 1  69  51  17  68  18 223
## Fila 2  77  15   4  82  83 261
## Fila 3  60  41  37  45  56 239
## Fila 4  98  13  40  10  32 193
## Sum    304 120  98 205 189 916
```

```
addmargins(mi_matriz2, c(1, 2), mean , quiet = TRUE)
```

```
##           A    B    C    D    E mean
## Fila 1  69  51 17.0 68.00 18.00 44.6
## Fila 2  77  15   4.0 82.00 83.00 52.2
## Fila 3  60  41 37.0 45.00 56.00 47.8
## Fila 4  98  13 40.0 10.00 32.00 38.6
## mean   76  30 24.5 51.25 47.25 45.8
```

# apply()

```
apply(X, MARGIN, FUN, ...)
```

X	Matriz. Si es un data.frame va a forzarlo al modo matrix
MARGIN	Vector que indica si la FUN se va a realizar por filas MARGIN = 1, o por columnas MARGIN = 2 o por ambas MARGIN = c(1,2)
FUN	Número entero que indica la cantidad de columnas.
...	Argumentos opcionales a FUN.

## Ejemplos:

```
(matriz <- matrix(round(rnorm(21, 5, 3), 2), 3, 7))
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,] 8.58 13.80 0.43 7.79 4.81 3.09 5.83
## [2,] 5.88  6.86 4.04 1.09 7.76 4.92 5.87
## [3,] 6.86  6.10 7.20 9.64 6.44 3.64 3.27
```

```
apply(matriz, 1, median)           # devuelve un vector
```

```
## [1] 5.83 5.87 6.44
```

```
apply(matriz, 1, fivenum)         # devuelve una matriz
```

```
##      [,1] [,2] [,3]
## [1,] 0.430 1.09 3.27
## [2,] 3.950 4.48 4.87
## [3,] 5.830 5.87 6.44
## [4,] 8.185 6.37 7.03
## [5,] 13.800 7.76 9.64
```

```
cbind(matriz, t(round(apply(matriz, 1, fivenum), 2)))
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## [1,] 8.58 13.80 0.43 7.79 4.81 3.09 5.83 0.43 3.95  5.83  8.19 13.80
## [2,] 5.88  6.86 4.04 1.09 7.76 4.92 5.87 1.09 4.48  5.87  6.37  7.76
## [3,] 6.86  6.10 7.20 9.64 6.44 3.64 3.27 3.27 4.87  6.44  7.03  9.64
```

# matplot()

```
m <- matrix(1:12, 4, 3)
matplot(m, cex = 1.5, ylim = c(0,15))
text(m[,3]+1, labels = paste("Fila", 1:4), cex = 0.9)
```

