

# Practica\_1\_R

Cristian Zaldaña

2022-07-28

## Índice

<b>CONSOLA DE R (R-CONSOLE)</b>	<b>2</b>
<b>TRABAJANDO CON SCRIPT</b>	<b>4</b>
<b>VECTORES DE DATOS</b>	<b>5</b>
VECTORES NUMÉRICOS . . . . .	5
OPERACIONES CON VECTORES NUMÉRICOS . . . . .	7
OPERACIONES DE FUNCIONES SOBRE VECTORES NUMÉRICOS. . . . .	7
VECTORES DE CARACTERES . . . . .	8
<b>CREACIÓN Y MANEJO DE MATRICES.</b>	<b>10</b>
CREACIÓN DE MATRICES NUMÉRICAS. . . . .	10
OPERACIONES CON MATRICES NUMÉRICAS. . . . .	12
CREACIÓN DE UNA MATRIZ DE CADENAS . . . . .	16
<b>CREACIÓN Y MANEJO DE MATRICES INDEXADAS (ARRAY).</b>	<b>16</b>

## CONSOLA DE R (R-CONSOLE)

Ejemplo 1. Encontrar el resultado de operar: 2 más 10 por 3 entre 5 Escriba en la Consola de R:  $2+10*3/5$  y oprima la tecla ENTER

```
2+10*3/5
```

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

Note que en R se respecta el mismo orden de preferencia de la mayoría de los lenguajes de programación, la multiplicación y la división tienen prioridad a la suma y resta

Ejemplo 2. Encontrar el resultado de operar: 3 elevado a la potencia 100  $3^{100}$  o también `format(3^100, sci = FALSE)`

```
format(3^100, sci = FALSE)
```

```
## [1] "515377520732011324202202224420402268886864624842"
```

`Sci=FALSE` le indica a R que muestre todos los dígitos del resultado, de lo contrario (`Sci=TRUE`) solamente se mostrará la representación científica.

Ejemplo 3. Encontrar el resultado anterior con 15 cifras decimales y guardarlo en la variable `y` y `<-format(3^100, digits = 15);y` o `y = format(3^100, digits = 15)`

```
y=format(3^100, digits = 15)
```

```
y
```

```
## [1] "5.15377520732011e+47"
```

Note que en R, la asignación de valores a una variable puede hacerse con `"="` o con `"<="`.

Ejemplo 4. Redondear el valor de  $\pi$  a 4 dígitos decimales `round(pi, 4)` Aplique las funciones: `trunc(pi)`, `floor(pi)` y `ceiling(pi)`

```
round(pi, 4)
```

```
## [1] 3.1416
```

```
trunc(pi)
```

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

```
floor(pi)
```

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

```
ceiling(pi)
```

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

Ejemplo 5. Guardar en la variable `n` el valor 150 y luego calcular el valor de `n = 150 factorial(n)`

```
n=150
```

```
factorial(n)
```

```
## [1] 5.713384e+262
```

Ejemplo 6. Operar el complejo  $(2+3i)$  elevado a la potencia 10  $(2+3i)^{10}$  o también `format((2+3i)^10, sci = TRUE)`

```
(2+3i)^10
```

```
## [1] -341525-145668i
```

Ejemplo 7. Calcular la integral entre 0 y  $\pi$  de la función Seno(x) f <- function(x) {sin(x)} integrate(f, lower = 0, upper = pi)

```
f=function(x) {sin(x)}  
integrate(f,lower = 0,upper = pi)
```

```
## 2 with absolute error < 2.2e-14
```

## TRABAJANDO CON SCRIPT

A medida que estemos realizando un trabajo de pequeña, mediana o de gran complejidad, será muy útil manejar todas las entradas que solicitemos a R en un entorno donde podamos corregirlas, retocarlas, repetirlas, guardarlas para continuar el trabajo en otro momento, con otros datos, etc. Esta es la función del editor de R, a los archivos creados en este editor se les conoce como Script. Es posible incluir comentarios que R no leerá si utilizamos líneas que comiencen con el carácter # (o en cualquier parte de la línea). Por el contrario, si escribimos cualquier orden no antecedida de # R (sin importar en que parte se encuentre) lo reconocerá como instrucciones que deben ejecutarse.

Este archivo se enviara por a parte. . .

## VECTORES DE DATOS

Este tipo de objetos se denominan estructuras atómicas ya que todos sus elementos son del mismo tipo o modo: character (carácter) o numeric (numérico) que puede ser integer (entero), double (real), complex (complejo), logical (lógico).

### VECTORES NUMÉRICOS

**FORMA 1-Crear un vector numérico vacío y añadirle luego sus elementos.**

- Ejemplo 1: `v <- numeric(3);v`

```
v <- numeric(3)
v
```

```
## [1] 0 0 0
```

El vector tiene longitud 3 y sus componentes serán NA (Not Available/“Missing” Values) que es la forma como R maneja los datos omitidos o faltantes.

- Ejemplo 2: `v[3] <- 17; v`

```
v[3] <- 17
v
```

```
## [1] 0 0 17
```

asigna el valor de 17 en la tercera posición del vector v.

**FORMA 2-Crear un vector numérico asignándole todos sus elementos o valores.**

- Ejemplo 1: `x <- c(2, 4, 3.1, 8, 6)`, revise el modo con `is.integer(x)` y `is.double(x)`; encuentre la longitud con: `length(x)`

```
x <- c(2,4,3.1,8,6)
is.integer(x)
```

```
## [1] FALSE
```

```
is.double(x)
```

```
## [1] TRUE
```

```
length(x)
```

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

- Ejemplo 2: Modifique el vector agregándole el valor 9 en la posición 3, use la siguiente la función de edición: `x <- edit(x)`

```
edit(x)
```

```
## [1] 2.0 4.0 3.1 8.0 6.0
```

**FORMA 3-Crear un vector numérico dando un rango de valores.**

- Ejemplo 1: `y = 1:4; y`

```
y = 1:4
y
```

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

Crea un vector de valores enteros en que su primer elemento es 1 su último es 4

- Ejemplo 2: Modificación de los elementos de un vector: `y[2] <- 5` (para modificar un elemento de un vector se escribe su nombre (del vector) y entre corchetes el índice del elemento que se quiera modificar).

```
y[2] <- 5
y
```

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

- Ejemplo 3: Crear un vector con elementos de otro; `u <- 1:12; u1=u[2 * 1:5]` (vector de tamaño 5 con elementos de las posiciones pares de u)

```
u <- 1:12
u1=u[2*1:5]
u1
```

```
## [1] 2 4 6 8 10
```

**FORMA 4-Crear un vector numérico utilizando la función `assign()`.**

- Ejemplo 1: `assign("z", c(x, 0, x)); z` (crea un vector en dos copias de x con un cero entre ambas)

```
assign("z", c(x,0,x))
z
```

```
## [1] 2.0 4.0 3.1 8.0 6.0 0.0 2.0 4.0 3.1 8.0 6.0
```

**FORMA 5-Crear un vector numérico generando una sucesión de valores.**

- Ejemplo 1: `s1 <- seq(2, 10); s1` (compárese a como fue generado el vector y y u)

```
s1 <- seq(2,10)
s1
```

```
## [1] 2 3 4 5 6 7 8 9 10
```

- Ejemplo 2: `s2 = seq(from=-1, to=5); s2`

```
s2 = seq(from=-1, to=5)
s2
```

```
## [1] -1 0 1 2 3 4 5
```

Crea un vector cuyo elemento inicial es -1 y su elemento final es 5, y cada dos elementos consecutivos del vector tienen una diferencia de una unidad.

- Ejemplo 3: `s3<-seq(to=2, from=-2); s3`

```
s3<-seq(to=2, from=-2)
s3
```

```
## [1] -2 -1 0 1 2
```

Note que puede invertir el orden de “to” y de “from”.

- Ejemplo 4: Secuencia con incremento o decremento: `s4=seq(from=-3, to=3, by=0.2); s4`

```
s4 = seq(from=-3, to=3, by=0.2)
s4
```

```
## [1] -3.0 -2.8 -2.6 -2.4 -2.2 -2.0 -1.8 -1.6 -1.4 -1.2 -1.0 -0.8 -0.6 -0.4 -0.2
## [16] 0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8
```

```
## [31] 3.0
```

Crea una secuencia que inicia en -3 y termina en 3 con incrementos de 0.2 en 0.2.

- Ejemplo 5. Repetición de una secuencia `s5 <- rep(s3, times=3)`; `s5`

```
s5 <- rep(s3, time=3)
s5
```

```
## [1] -2 -1 0 1 2 -2 -1 0 1 2 -2 -1 0 1 2
```

## OPERACIONES CON VECTORES NUMÉRICOS

- Ejemplo 1: `1/x` (observe que calcula el inverso de cada elemento del vector)

```
1/x
```

```
## [1] 0.5000000 0.2500000 0.3225806 0.1250000 0.1666667
```

- Ejemplo 2: `v=2*x+z+1`; `v` (genera un nuevo vector, `v`, de longitud 11, construido sumando, elemento a elemento, el vector `2*x` repetido 2.2 veces, el vector `y`, y el número 1 repetido 11 veces **“Reciclado en R es repetir las veces necesarias un vector cuando en una operación intervienen vectores de distinta longitud”**).

```
v=2*x+z+1
```

```
## Warning in 2 * x + z: longitud de objeto mayor no es múltiplo de la longitud de
## uno menor
```

```
v
```

```
## [1] 7.0 13.0 10.3 25.0 19.0 5.0 11.0 11.2 20.1 21.0 11.0
```

- Ejemplo 3: `e1 <- c(1, 2, 3, 4)`; `e2<-c(4, 5, 6, 7)`; `crossprod(e1, e2)` ó `t(e1)`

```
e1<-c(1,2,3,4)
e2<-c(4,5,6,7)
crossprod(e1,e2)
```

```
##      [,1]
## [1,]    60
```

```
t(e1)%*%e2
```

```
##      [,1]
## [1,]    60
```

## OPERACIONES DE FUNCIONES SOBRE VECTORES NUMÉRICOS.

- Ejemplo 1: Vector transpuesto del vector `x`: `xt = t(x)`; `xt`.

```
xt=t(x)
xt
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    2    4  3.1    8    6
```

- Ejemplo 2: `u = exp(y)`; `u` (crea un nuevo vector de la misma longitud que `y`, en el cual cada elemento es la exponencial elevando a su respectivo elemento en `y`).

```
``r
u=exp(y)
u
``
``
## [1] 2.718282 148.413159 20.085537 54.598150
``
```

```
u=exp(y)
u
```

```
## [1] 2.718282 148.413159 20.085537 54.598150
```

- options(digits=10); u Permite visualizar un mínimo de 10 dígitos

```
options(digits = 10)
u
```

```
## [1] 2.718281828 148.413159103 20.085536923 54.598150033
```

### OTRAS OPERACIONES:

- Ejemplo 1: resum <- c(length(y), sum(y), prod(y), min(y), max(y)); resum

```
resum<-c(length(y), sum(y), prod(y), min(y), max(y))
resum
```

```
## [1] 4 13 60 1 5
```

- Ejemplo 2: Ordenamiento de un vector: yo <- sort(y); yo

```
yo <- sort(y)
yo
```

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

## VECTORES DE CARACTERES

### FORMA 1-Crear un vector de caracteres vacío y añadirle luego sus elementos.

- Ejemplo 1: S<-character()

```
s<-character()
```

### FORMA 2-Crear un vector de caracteres asignándole todos sus elementos.

- Ejemplo 1: Crear el vector de caracteres: deptos <- c("Santa Ana", "Sonsonate", "San Salvador"); deptos

```
deptos <- c("Santa Ana", "Sonsonate", "San Salvador")
deptos
```

```
## [1] "Santa Ana" "Sonsonate" "San Salvador"
```

- Ejemplo 2: Agregue el elemento "Ahuachapán" en la cuarta posición. deptos[4]="Ahuachapán"; deptos (R Permite incrementar el tamaño del vector en cualquier instante).



```
deptos[4]="Ahuachapán"  
deptos
```

```
## [1] "Santa Ana" "Sonsonate" "San Salvador" "Ahuachapán"
```

**FORMA 3-Crear un vector de caracteres dándole nombres a los elementos para identificarlos más fácilmente.**

- Ejemplo 1: `codDeptos <- c(11, 12, 13, 14)` `names(codDeptos) <- c("Usulután", "San Miguel", "Morazán", "La Unión");``codDeptos Oriente <- codDeptos [c("La Unión", "San Miguel")];``Oriente`

```
codDeptos <- c(11,12,13,14)  
names(codDeptos) <- c("Usulután", "San Miguel", "Morazán", "La Unión")  
codDeptos
```

```
## Usulután San Miguel Morazán La Unión  
## 11 12 13 14
```

```
Oriente <- codDeptos [c("La Unión", "San Miguel")]  
Oriente
```

```
## La Unión San Miguel  
## 14 12
```

- Ejemplo 2: Crear un vector con las etiquetas X1, Y2, ... , X9, Y10. `etiqs<-paste(c("X", "Y"), 1:10, sep="");` `etiqs`

```
etiqs <- paste(c("X","Y"), 1:10, sep = "")  
etiqs
```

```
## [1] "X1" "Y2" "X3" "Y4" "X5" "Y6" "X7" "Y8" "X9" "Y10"
```

Crea un vector de caracteres resultado de la unión de “X” o de “Y” con uno de los número comprendidos entre 1 y 10, `sep=""` indica que no se deja espaciado en la unión.

## CREACIÓN Y MANEJO DE MATRICES.

### CREACIÓN DE MATRICES NUMÉRICAS.

**FORMA 1-Crear una matriz numérica vacía y añadirle luego sus elementos.**

- Ejemplo 1: `M <- matrix(numeric(), nrow = 3, ncol=4)`

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

- Ejemplo 2: Asignación de los elementos de una matriz: `M[2,3] <- 6`; M. similar a la de un vector pero considerando que deben utilizarse dos índices para indicar fila y columna.

```
M[2,3] <- 6
```

```
M
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   NA   NA   NA   NA
## [2,]   NA   NA    6   NA
## [3,]   NA   NA   NA   NA
```

**FORMA 2-Crear una matriz numérica asignándole todos sus elementos o valores.**

- Ejemplo 1: `A <- matrix(c(2, 4, 6, 8, 10, 12), nrow=2, ncol=3)`; A Observe que R almacena los elementos por columna. Se pueden explorar algunas características de la matriz A, por ejemplo: `mode(A)`; `dim(A)`; `attributes(A)`; `is.matrix(A)`; `is.array(A)`

```
A <- matrix(c(2,4,6,8,10,12), nrow = 2, ncol = 3)
```

```
A
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    4    8   12
```

```
mode(A)
```

```
## [1] "numeric"
```

```
dim(A)
```

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

```
attributes(A)
```

```
## $dim
```

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

```
is.matrix(A)
```

```
## [1] TRUE
```

```
is.array(A)
```

```
## [1] TRUE
```

**FORMA 3-Crear una matriz numérica dando un rango de valores**

- Ejemplo 1: `B <- matrix(1:12, nrow=3, ncol=4); B`

```
B <- matrix(1:12, nrow = 3, ncol = 4)
```

```
B
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

#### FORMA 4-Crear una matriz a partir de la unión de vectores

##### I. Crear tres vectores

```
x1 <- seq(0, 10, 2); x1
```

```
x2 <- seq(1, 11, 2); x2
```

```
x3 <- runif(6); x3 Vector con valores de una uniforme(0,1)
```

```
```r
x1<-seq(0,10,2)
x1
```

```
## [1]  0  2  4  6  8 10
```
```

```
```r
x2<-seq(1,11,2)
x2
```

```
## [1]  1  3  5  7  9 11
```
```

```
```r
x3<-runif(6)
x3
```

```
## [1] 0.30543221976 0.88818790112 0.94454932632 0.15252033062 0.33599477191
## [6] 0.09657459054
```
```

##### II. Unir los tres vectores en una matriz por columnas.

```
Xcol <- cbind(x1, x2, x3); Xcol
```

```
```r
Xcol <- cbind(x1,x2,x3)
Xcol
```

```
```

```
##      x1 x2      x3
## [1,]  0  1 0.30543221976
## [2,]  2  3 0.88818790112
## [3,]  4  5 0.94454932632
## [4,]  6  7 0.15252033062
## [5,]  8  9 0.33599477191
## [6,] 10 11 0.09657459054
```
```

III. Unir los tres vectores en una matriz por filas.

```
Xfil <- rbind(x1, x2, x3); Xfil
```

```
```r
Xfil<-rbind(x1,x2,x3)
Xfil
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## x1 0.0000000000 2.0000000000 4.0000000000 6.0000000000 8.0000000000
## x2 1.0000000000 3.0000000000 5.0000000000 7.0000000000 9.0000000000
## x3 0.3054322198 0.8881879011 0.9445493263 0.1525203306 0.3359947719
##           [,6]
## x1 10.0000000000
## x2 11.0000000000
## x3  0.09657459054
```
```

IV. Acceso a las filas y columnas de una matriz.

$X \leftarrow Xfil[1:3, c(2, 3)]$ ;  $X$  (crea una submatriz de dimensión  $3 \times 2$  (el 3 se indica por 1:3), las columnas están conformadas por la segunda y tercera columna de la matriz  $Xfil$  (se indica por  $C(2,3)$ ))

```
```r
x<-Xfil[1:3,c(2,3)]
x
```

```
##           [,1]      [,2]
## x1 2.0000000000 4.0000000000
## x2 3.0000000000 5.0000000000
## x3 0.8881879011 0.9445493263
```
```

## OPERACIONES CON MATRICES NUMÉRICAS.

### MULTIPLICACIÓN DE MATRICES MATRICES NUMÉRICAS:

- Ejemplo 1: Multiplicación de un vector por una matriz:  $v \leftarrow c(1, 2)$ ;  $v \%*\% A$

```

```r
v<-c(1,2)
A
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    4    8   12
```

```

```

```r
v%*%A
```

```
##      [,1] [,2] [,3]
## [1,]   10   22   34
```

```

- Ejemplo 2: Multiplicación de matrices:  $P \leftarrow A \%*\% B$ ; P

```

```r
P<-A%*%B
A
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    4    8   12
```

```r
B
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```r
P
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   44   98  152  206
```

```

```
## [2,] 56 128 200 272
```
```

- Ejemplo 3: Multiplicación de un escalar por una matriz:  $2*A$  (nótese que al usar  $2\%*A$  se obtiene un error pues las dimensiones no son compatibles).

```
```r
2*A
```
```

```
```
##      [,1] [,2] [,3]
## [1,]    4   12   20
## [2,]    8   16   24
```
```

#### OPERACIONES DE FUNCIONES SOBRE MATRICES NUMÉRICAS:

- Ejemplo 1: Longitud o número de elementos: `length(A)`

```
length(A)
```

```
## [1] 6
```

- Ejemplo 2:  $T=\sqrt{B}$ ;  $T$  (observe que la raíz se saca a cada elemento de la matriz)

```
T=sqrt(B)
T
```

```
##      [,1]      [,2]      [,3]      [,4]
## [1,] 1.000000000 2.000000000 2.645751311 3.162277660
## [2,] 1.414213562 2.236067977 2.828427125 3.316624790
## [3,] 1.732050808 2.449489743 3.000000000 3.464101615
```

- Ejemplo 3: Transpuesta de una matriz: `t(A)`

```
t(A)
```

```
##      [,1] [,2]
## [1,]    2    4
## [2,]    6    8
## [3,]   10   12
```

- Ejemplo 4: Determinante de una matriz:  $C \leftarrow \text{matrix}(c(2, 1, 10, 12), \text{nrow}=2, \text{ncol}=2)$ ;  $\det(C)$

```
C<-matrix(c(2,1,10,12), nrow = 2, ncol = 2)
C
```

```
##      [,1] [,2]
## [1,]    2   10
## [2,]    1   12
```

```
det(C)
```

```
## [1] 14
```

- Ejemplo 5: Inversa de una matriz, resulta de resolver el sistema  $Ax = b$  con  $b=I$ :  $\text{InvC} \leftarrow \text{solve}(C)$ ;  $\text{InvC}$  O también:  $b=\text{diag}(2)$ ;  $\text{InvC} \leftarrow \text{solve}(C, b)$ ;  $\text{InvC}$

```
InvC<-solve(C)
InvC
```

```
##           [,1]      [,2]
## [1,]  0.85714285714 -0.7142857143
## [2,] -0.07142857143  0.1428571429
```

```
b=diag(2)
InvC<-solve(C,b)
InvC
```

```
##           [,1]      [,2]
## [1,]  0.85714285714 -0.7142857143
## [2,] -0.07142857143  0.1428571429
```

- Ejemplo 6: Autovalores y autovectores de una matriz simétrica: eigen(C)

```
eigen(C)
```

```
## eigen() decomposition
## $values
## [1] 12.916079783  1.083920217
##
## $vectors
##           [,1]      [,2]
## [1,] -0.6754894393 -0.99583021557
## [2,] -0.7373696613  0.09122599279
```

- Ejemplo 7: La función diag(nombMatriz), devuelve un vector formado por los elementos en la diagonal de la matriz nombMatriz.

```
diag(C)
```

```
## [1]  2 12
```

- Ejemplo 8: La función diag(nomVector), devuelve una matriz diagonal cuyos elementos en la diagonal son los elementos del vector nomVector.

```
diag(v)
```

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

- Ejemplo 9: La función diag(escalar), devuelve la matriz identidad de tamaño nxn.

```
diag(2)
```

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

#### OTRAS OPERACIONES:

- Ejemplo 1: c(length(A), sum(A), prod(A), min(A), max(A))

```
c(length(A), sum(A), prod(A), min(A), max(A))
```

```
## [1]      6    42 46080      2    12
```

A

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    4    8   12
```

- Ejemplo 2: `O <- matrix(sort(C), nrow=2, ncol=2)`; `O` (`sort()` genera un vector en los cuales sus elementos han sido ordenados de menor a mayor a partir de los elementos de la matriz `C`).

```
o<-matrix(sort(C), nrow = 2, ncol = 2)
o
```

```
##      [,1] [,2]
## [1,]    1   10
## [2,]    2   12
```

C

```
##      [,1] [,2]
## [1,]    2   10
## [2,]    1   12
```

## CREACIÓN DE UNA MATRIZ DE CADENAS

- Ejemplo 1: `nombres <- matrix(c("Carlos", "José", "Ana", "René", "María", "Mario"), nrow=3, ncol=2)`; `nombres`

```
nombres<-matrix(c("Carlos","Jose","Ana","Rene","Maria","Mario"), nrow = 3, ncol = 2)
nombres
```

```
##      [,1]      [,2]
## [1,] "Carlos" "Rene"
## [2,] "Jose"   "Maria"
## [3,] "Ana"    "Mario"
```

## CREACIÓN Y MANEJO DE MATRICES INDEXADAS (ARRAY).

Una variable indexada (array) es una colección de datos, por ejemplo numéricos, indexada por varios índices. R permite crear y manipular variables indexadas en general y en particular, matrices. Una variable indexada puede utilizar no sólo un vector de índices, sino incluso una variable indexada de índices, tanto para asignar un vector a una colección irregular de elementos de una variable indexada como para extraer una colección irregular de elementos.

Un vector es un array unidimensional y una matriz es un array bidimensional.

Una variable indexada se construye con la función `array()`, que tiene la forma general siguiente:

```
NombMatriz <- array(vector_de_datos, vector_de_dimensiones
```

- Ejemplo 1: `X <- array(c(1, 3, 5, 7, 9, 11), dim=c(2, 3))`; `X`

```
X<-array(c(1,3,5,7,9,11), dim = c(2,3))
X
```



```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    3    7   11
```

- Ejemplo 2: `Z <- array(1, c(3, 3)); Z`

```
Z<-array(1,c(3,3))
Z
```

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

- Ejemplo 3: Operaciones aritméticas: `W <- 2*Z+1; W`

```
W<-2*Z+1
W
```

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

- Ejemplo 4: Operaciones con funciones: `TX <- t(X); TX`

```
TX<-t(X)
TX
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    5    7
## [3,]    9   11
```

- Ejemplo 5: Producto exterior de dos vectores con: operador `%o%`  
`a <- c(2, 4, 6); a`  
`b <- 1:3;b`  
`M <- a %o% b; M` # M es un array o matriz.

```
a<-c(2,4,6)
a
```

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

```
b<-1:3
b
```

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

```
M<-a%o%b
M
```

```
##      [,1] [,2] [,3]
## [1,]    2    4    6
## [2,]    4    8   12
## [3,]    6   12   18
```

Nota: `c <- a * b`; `c` devuelve un vector con el producto de elemento por elemento

En R se distingue entre matrices y arrays: las matrices son colecciones de elementos indexados por filas

y columnas; los arrays son extensiones de ellas donde el conjunto de índices o dimensiones puede ser mayor que dos.

- Ejemplo 6. Una matriz de tres dimensiones (i, j, k)

```
Arreglo3 <- array(c(1:8, 11:18, 111:118), dim = c(2, 4, 3));
```

Arreglo3 # un arreglo de 3 matrices cada una de 2 filas y 4 columnas.

```
Arreglo3<-array(c(1:8, 11:18,111:118), dim = c(2,4,3))  
Arreglo3
```

```
## , , 1  
##  
##      [,1] [,2] [,3] [,4]  
## [1,]    1    3    5    7  
## [2,]    2    4    6    8  
##  
## , , 2  
##  
##      [,1] [,2] [,3] [,4]  
## [1,]   11   13   15   17  
## [2,]   12   14   16   18  
##  
## , , 3  
##  
##      [,1] [,2] [,3] [,4]  
## [1,]  111  113  115  117  
## [2,]  112  114  116  118
```