

# PRACTICA\_02

Nelson de Jesus Magaña Godinez

30/7/2022

## Contents

1.1 VECTORES NUMÉRICOS . . . . .	2
1.1.1 OPERACIONES CON VECTORES NUMÉRICOS. . . . .	4
1.1.2 OPERACIONES DE FUNCIONES SOBRE VECTORES NUMÉRICOS. . . . .	5
OTRAS OPERACIONES: . . . . .	6
1.2 VECTORES DE CARACTERES. . . . .	6
2.1 CREACIÓN DE MATRICES NUMÉRICAS. . . . .	7
2.2 OPERACIONES CON MATRICES NUMÉRICAS. . . . .	10
MULTIPLICACIÓN DE MATRICES MATRICES NUMÉRICAS: . . . . .	10
OPERACIONES DE FUNCIONES SOBRE MATRICES NUMÉRICAS: . . . . .	10
OTRAS OPERACIONES: . . . . .	13
2.3 CREACIÓN DE UNA MATRIZ DE CADENAS . . . . .	13

## 1. CREACIÓN Y MANEJO DE 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).

### 1.1 VECTORES NUMÉRICOS

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

- Ejemplo 1:

```
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 # Asigna el valor de 17 en la tercera posición del vector v.
```

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

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

-Ejemplo 1:

```
x <- c(2,4,3.1,8,6)  
x #Mostrar el vector
```

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

```
is.integer(x) #¿El vector contiene datos tipo enteros?
```

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

```
is.double(x) #¿ El vector contiene datos de tipo doubles?
```

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

```
length(x) # Tamaño del vector
```

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

- Ejemplo 2: Modifique el vector agregándole el valor 9 en la posición 3.

```
#x <- edit(x) #Este comando muestra una ventana para agregar el valor
# en la posición en donde se requiera
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; # crea un vector de valores enteros en que su primer elemento es 1 su último es 4
y
```

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

- Ejemplo 2: Modificación de los elementos de un vector, (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; (vector de tamaño 5 con elementos de las posiciones pares de u)

```
u <- 1:12;
u
```

```
## [1] 1 2 3 4 5 6 7 8 9 10 11 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: (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); #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.  
s2
```

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

- Ejemplo 3:

```
s3 <- seq(to=2, from=-2); # note que puede invertir el orden de "to" y de "from".  
s3
```

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

- Ejemplo 4: Secuencia con incremento o decremento:

```
s4 <-seq(from=-3, to=3, by=0.2); # crea una secuencia que inicia en -3 y termina en 3  
# con incrementos de 0.2 en 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
```

- Ejemplo 5: Repetición de una secuencia

```
s5 <-rep(s3, times=3); #repite el vector s3, tres veces consecutivas  
s5
```

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

### 1.1.1 OPERACIONES CON VECTORES NUMÉRICOS.

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

```
x
```

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

```
1/x
```

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

- *Ejemplo 2:*  $v=2x+z+1$ ;  $v$  (genera un nuevo vector,  $v$ , de longitud 11, construido sumando, elemento a elemento, el vector  $2x$  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:* (calcula el producto interno entre dos vectores. Ambos deben tener el mismo número de elementos.

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

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

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

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

### 1.1.2 OPERACIONES DE FUNCIONES SOBRE VECTORES NUMÉRICOS.

- *Ejemplo 1:* Vector transpuesto del vector  $x$ :  $x_t = t(x)$ ;  $x_t$ .

```
x
```

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

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

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

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

```
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:

```
resun <- c(length(y), sum(y), prod(y), min(y), max(y));  
resun # tamaño, suma, producto, mínimo y máximo. todo del vector y
```

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

- Ejemplo 2: Ordenamiento de un vector

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

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

## 1.2 VECTORES DE CARACTERES.

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

- Ejemplo 1: `S<-character()`

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

```
## character(0)
```

**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
```

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

- Ejemplo 2: Agregue el elemento “Ahuachapán” en la cuarta posición. (R Permite incrementar el tamaño del vector en cualquier instante).

```
deptos[4]="Ahuachapan";  
deptos
```

```
## [1] "Santa Ana" "Sonsonate" "san Salvador" "Ahuachapan"
```

**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"); # Asigna el código
# correspondiente a cada caso.
codDeptos
```

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

```
Oriente <- codDeptos[c("La Unión", "San Miguel")]; #Muestra el código de los casos solicitados.
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
```

```
##      [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 números comprendidos entre 1 y 10, sep=""
# indica que no se deja espaciado en la unión.
```

## 2. CREACION Y MANEJO DE MATRICES

### 2.1 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)
M
```

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

- Ejemplo 2: Asignación de los elementos de una matriz

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

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

```
#similar a la de un vector pero considerando que deben
# utilizarse dos índices para indicar fila y columna.
```

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

- *Ejemplo 1:* 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) #tipo de variables que contiene.
```

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

```
dim(A) # Dimencion de la matriz, filas, columnas.
```

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

```
attributes(A) # Atributos
```

```
## $dim
## [1] 2 3
```

```
is.matrix(A) #Pregunta si es una matriz.
```

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

```
is.array(A) #Pregunta si es un arreglo.
```

```
## [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
```

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

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

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

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

```
## [1] 0.12578460900 0.59251501178 0.03905041097 0.44838090520 0.80107001704
## [6] 0.89410877437
```

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

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

```
##      x1 x2      x3
## [1,]  0  1 0.12578460900
## [2,]  2  3 0.59251501178
## [3,]  4  5 0.03905041097
## [4,]  6  7 0.44838090520
## [5,]  8  9 0.80107001704
## [6,] 10 11 0.89410877437
```

III. Unir los tres vectores en una matriz por filas. Xfil <- rbind(x1, x2, x3); Xfil

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

```
##      [,1]      [,2]      [,3]      [,4]      [,5]
## x1 0.000000000 2.000000000 4.000000000 6.000000000 8.000000000
## x2 1.000000000 3.000000000 5.000000000 7.000000000 9.000000000
## x3 0.125784609 0.592515011 0.039050410 0.448380905 0.801070017
##      [,6]
## x1 10.000000000
## x2 11.000000000
## x3  0.894108744
```

IV. Acceso a las filas y columnas de una matriz. (crea una submatriz de dimensión 3x2 (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))

```
x <- xfil[1:3, c(2,3)];  
x
```

```
##           [,1]      [,2]  
## x1 2.0000000000 4.0000000000  
## x2 3.0000000000 5.0000000000  
## x3 0.5925150118 0.03905041097
```

## 2.2 OPERACIONES CON MATRICES NUMÉRICAS.

### MULTIPLICACIÓN DE MATRICES MATRICES NUMÉRICAS:

- *Ejemplo 1:* Multiplicación de un vector por una matriz:

```
v <- c(1,2);  
v*%A
```

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

- *Ejemplo 2:* Multiplicación de matrices:

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

```
##      [,1] [,2] [,3] [,4]  
## [1,]   44   98  152  206  
## [2,]   56  128  200  272
```

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

```
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:* (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:

```
A #Matriz
```

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

```
t(A) #Transpuesta de la matriz
```

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

- Ejemplo 4: Determinante de una matriz:

```
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  $b=I$ :

```
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`.

```
C # Matriz
```

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

```
diag(C) #Diagonal de la matriz
```

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

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

```
v # vector
```

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

```
diag(v) # Matriz vector
```

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

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

```
diag(5) # el Escalar le da la dimencion de la matriz
```

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

## OTRAS OPERACIONES:

- Ejemplo 1:

```
A # Matriz
```

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

```
ej1 <- c(length(A), sum(A), prod(A), min(A), max(A)); #Operaciones en un mismo vector
ej1
```

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

- Ejemplo 2: (sort()) genera un vector en los cual sus elementos han sido ordenados de menor a mayor a partir de los elementos de la matriz C).

```
0 <- matrix(sort(C), nrow = 2, ncol = 2);
0
```

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

## 2.3 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"
```

## 3. 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
```

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

- Ejemplo 2:

```
Z <- array(1, c(3,3)); #Matriz en donde todos los elementos son 1, de dimension 3x3
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
```

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

- Ejemplo 4: Operaciones con funciones:

```
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
```

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

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

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

```
M <- a%o% b;
M # M es un array o matriz.
```

```
##      [,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.

```
c <- a*b;
c
```

```
## [1]  2  8 18
```

- 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.
```

```
## , , 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
```