

UNIVERSIDAD DE EL SALVADOR  
FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE  
DEPARTAMENTO DE MATEMÁTICA



Licenciatura en Estadística

Control Estadístico del Paquete R

”UNIDAD UNO”

PRÁCTICA 02 - Tipos de objetos, operadores y funciones que operan sobre ellos: Vectores, matrices y arreglos (matrices indexadas). .

Alumna:  
Martha Yoana Medina Sánchez

Fecha de elaboración  $\tilde{A}^3_n$   
Santa Ana - 27 de noviembre de 2015

**Practica 02 - Tipos de objetos, operadores y funciones que operan sobre ellos:  
Vectores, matrices y arreglos (matrices indexadas).**

**1. CREACIÓN Y MANEJO DE VECTORES DE DATOS.**

**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 serían NA (datos omitidos o faltantes).
```

Ejemplo 2:

```
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);is.integer(x);is.double(x);length(x);x
## [1] FALSE
## [1] TRUE
## [1] 5
## [1] 2.0 4.0 3.1 8.0 6.0
```

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

```
x <- edit(x)
```

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

Ejemplo 1:

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

Ejemplo 3: Crear un vector con elementos de otro

```
u <- 1:12; u1=u[2 * 1:5];u1  
## [1] 2 4 6 8 10  
  
# Vector de tama~no 5 con elementos de las posiciones pares de u
```

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

Ejemplo 1

```
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  
  
# Crea un vector en dos copias de x con un cero entre ambas
```

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

Ejemplo 1:

```
s1 <- seq(2, 10); s1  
## [1] 2 3 4 5 6 7 8 9 10  
  
# Comp~arese a como fue generado el vector y y u
```

Ejemplo 2:

```
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
```

Ejemplo 3:

```
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

## [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
## [15] -0.2  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
## [29]  2.6  2.8  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

## [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;x

## [1] 0.5000000 0.2500000 0.3225806 0.1250000 0.1666667
## [1] 2.0 4.0 3.1 8.0 6.0

# Observe que calcula el inverso de cada elemento del vector
```

Ejemplo 2:

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

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

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

#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"
```

Ejemplo 3:

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

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

```
t(e1)%*%e2

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

# Calcula el producto interno entre dos vectores. Ambos deben tener el
# mismo n\umero de elementos.
```

### 1.1.2 OPERACIONES DE FUNCIONES SOBRE VECTORES NUMÉRICOS.

Ejemplo 1: Vector transpuesto del vector x:

```
xt = t(x); xt

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

Ejemplo 2:

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

## [1] 1 5 3 4
## [1]  2.718282 148.413159  20.085537  54.598150

# 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.
```

```
options(digits=10); u

## [1]  2.718281828 148.413159103  20.085536923  54.598150033

# Permite visualizar un m\inimo de 10 d\igitos
```

### OTRAS OPERACIONES:

Ejemplo 1:

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

## [1] 1 5 3 4
## [1]  4 13 60  1  5
```

Ejemplo 2: Ordenamiento de un vector

```
yo <- sort(y); y;yo
## [1] 1 5 3 4
## [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()
```

**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.<sup>en</sup> la cuarta posición.

```
deptos[4]="Ahuachap'an"; deptos
## [1] "Santa Ana"      "Sonsonate"      "San Salvador" "Ahuachap'an"
# R Permite incrementar el tamaño del vector en cualquier instante.
```

**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", "Moraz'an", "La Uni'on");codDeptos
## Usulután San Miguel Moraz'an La Uni'on
##      11      12      13      14
Oriente <- codDeptos [c("La Uni'on", "San Miguel)];Oriente
## La Uni'on 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 uni\ 'on de "X" o de "Y"
#con uno de los n\ 'umero comprendidos entre 1 y 10, sep=""
#indica que no se deja espaciado en la uni\ 'on.
```

## 2. CREACIÓN 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

##      [,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 Índicespara indicar fila y columna.
```

**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;

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

mode(A); dim(A); attributes(A); is.matrix(A); is.array(A)
```

```
## [1] "numeric"
## [1] 2 3
## $dim
## [1] 2 3
## [1] TRUE
## [1] TRUE

#Observe que R almacena los elementos por columna.
```

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

Ejemplo 1:

```
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.

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); x3

## [1] 0.7392216821 0.1308083450 0.2109663961 0.2474708515 0.2450251312
## [6] 0.1640773728

# Vector con valores de una uniforme(0,1)
```

Unir los tres vectores en una matriz por columnas.

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

##      x1 x2      x3
## [1,]  0  1 0.7392216821
## [2,]  2  3 0.1308083450
## [3,]  4  5 0.2109663961
```



```
## [4,] 6 7 0.2474708515
## [5,] 8 9 0.2450251312
## [6,] 10 11 0.1640773728
```

Unir los tres vectores en una matriz por filas.

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

##           [,1]      [,2]      [,3]      [,4]      [,5]
## x1 0.0000000000 2.000000000 4.000000000 6.000000000 8.000000000
## x2 1.0000000000 3.000000000 5.000000000 7.000000000 9.000000000
## x3 0.7392216821 0.130808345 0.2109663961 0.2474708515 0.2450251312
##           [,6]
## x1 10.0000000000
## x2 11.0000000000
## x3 0.1640773728
```

Acceso a las filas y columnas de una matriz.

```
X <- Xfil[1:3, c(2, 3)]; X

##           [,1]      [,2]
## x1 2.000000000 4.000000000
## x2 3.000000000 5.000000000
## x3 0.130808345 0.2109663961

# Crea una submatriz de dimensi\on 3x2 (el 3 se indica por 1:3), las
#columnas est\an conformadas por la segunda y tercera columna de la
#matriz Xfill (se indica por C(2,3))
```

## 2.2 OPERACIONES CON MATRICES NUMÉRICAS.

### MULTIPLICACIÓN DE MATRICES MATRICES NUMÉRICAS:

Ejemplo 1: Multiplicacion 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:

```
2*A

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

```
2%*%A

## Error in 2%*% A: argumentos no compatibles

#ñtese que al usar 2%*%A se obtiene un error pues las dimensiones no son compatibles
```

## OPERACIONES DE FUNCIONES SOBRE MATRICES NUMÉRICAS:

Ejemplo 1: Longitud o número de elementos:

```
length(A)

## [1] 6
```

Ejemplo 2:

```
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

#Observe que la raíz se saca a cada elemento de la matriz.
```

Ejemplo 3: Transpuesta de una matriz:

```
t(A)

##           [,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
```

o tambien:

```
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)

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

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

```
diag(A)

## [1] 2 8
```

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

```
diag(X)
```

```
## [1] 2 5
```

Ejemplo 9: La función `diag(escalar)`, devuelve la matriz identidad de tamaño  $n \times n$ .

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

Ejemplo 2:

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

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

*#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*

## 2.3 CREACIÓN DE UNA MATRIZ DE CADENAS.

Ejemplo 1:

```
nombres <- matrix(c("Carlos", "JosÃ©", "Ana", "Ren'e", "Mar'ia", "Mario"),  
nrow=3, ncol=2); nombres  
##      [,1] [,2]  
## [1,] "Carlos" "Ren'e"  
## [2,] "JosÃ©" "Mar'ia"  
## [3,] "Ana" "Mario"
```

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

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)); 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
```

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

```
# M es un array o matriz.
```

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

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

## , , 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

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