# Tema 3 - Estructuras de datos con R

Curso Estadística Descriptiva

2022-09-18

## Vectores

**Vector.**Es una secuencia de datos ordenada y homogénea.

### Algunos tipos de datos en R

`R` dispone de diferentes tipos de datos como:

- **logical**: lógicos (`True` or `False`)
- **integer**: Números entero, Z
- **numeric**: números reales, R
- **complex**: complejos, C
- **character**: palabras

Los objetos del vector tienen que ser del mismo tipo; O todo enteros, o todo palabras, por ejemplo. Si se puede crear vectores con diferentes tipos de datos, pero para ello hay que utilizar las listas generalizadas.

### Funciones básicas para vectores

- **c()**: para definir un vector. Concatena todo lo que esté entre paréntesis (incluso variables)

```
c(1, 2, 3)
```

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

- **scan()**: se utilizará cuando escaneemos datos escritos por consola

```
scan()
```

```
## numeric(0)
```

```
#nos dará la opción de escanear por consola cuantos elementos queramos
```

- **fix()**: para modificar visualmente el vector. Mostrará una ventana emergente

```
x <- c(1,2,3)
fix(x) #
```

- **rep()**: para definir un vector constante que tien un dato $a$, repetido $n$ veces

```
rep("Matemáticas", 5)
```

```
## [1] "Matemáticas" "Matemáticas" "Matemáticas" "Matemáticas" "Matemáticas"
```

Para ver el tipo de dato del vector, utilizamos la función **class(nombre_vector)**

## Funciones de Vectores

Para aplicar funciones a un vector x:

```
x <- 1:5

x*pi
x *2
x^2
sqrt(x)
```

```
## [1]  3.141593  6.283185  9.424778 12.566371 15.707963
## [1]  2  4  6  8 10
## [1]  1  4  9 16 25
## [1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

Habrá veces en que no se podrá aplicar la función a cada uno de los elementos del vector, para esos casos, existe:

- **sapply(nombre_vector, FUN = nombre_funcion):** para aplicar dicha función a todos los elementos del vector.
  - Por ejemplo, para calcular la raíz cuadrada de cada uno de los elementos del vector x

```
x
```

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

```
sapply(x,FUN = function(y){sqrt(y)})
```

```
## [1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

```
#De esta forma también es válido

cuadrado = function(x){x^2}
v = c(1,2,3,4,5,6)
sapply(v, FUN = cuadrado)
```

```
## [1]  1  4  9 16 25 36
```

**Funciones aplicadas a vectores relacionadas al mundo de la estadística (medidas estadísticas):**

- **length(x)**: calcula la longitud del vector x

```
length(x)
```

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

- **max(x) y min(x)**: calcula el valor máximo y mínimo respectivamente del vector x

```
max(x)
```

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

```
min(x)
```

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

- **sum(x) y prod(x)**: calcula la suma y el producto respectivamente de las entradas del vector x

```
sum(x)
```

```
## [1] 15
```

```
prod(x)
```

```
## [1] 120
```

- **mean(x)**: calcula la media aritmética de las entradas del vector x

```
mean(x)
```

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

- **diff(x)**: calcula el vector formado por las diferencias sucesivas entre entradas del vector original x

```
diff(x)
```

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

- **cumsum(x)**: calcula el vector formado por las sumas acumuladas de las entradas del vector original $x$
  - Permite definir sucesiones descritas mediante sumatorios
  - Cada entrada de `cumsum(x)` es la suma de las entradas de $x$ hasta su posición

## Orden de Vectores

- **sort(x)**:: ordena el vector en orden natural de los objetos que lo forman: el orden numérico creciente, orden alfabético...
- **rev(x)**: invierte el orden de los elementos del vector $x$

```r
v = c(1,7,5,2,4,6,3)
sort(v)
```

```
## [1] 1 2 3 4 5 6 7
```

```r
rev(v)
```

```
## [1] 3 6 4 2 5 7 1
```

## Subvectores

**Vector[i]**: da la i-ésima entrada del vector

- Los índices en R empiezan en 1
- **vector[length(vector)]**: nos da la última entrada del vector

```r
v = c(2,4,5,8,10,12,15,20,21)
v[2]
v[length(v)-1]
v[v%%2==0]
```

```
## [1] 4
## [1] 20
## [1]  2  4  8 10 12 20
```

- **vector[a:b]**: si $a$ y $b$ son dos números naturales, nos da el subvector con las entradas del vector original que van de la posición $a$-ésima hasta la $b$-ésima.

```r
v[2:4]
v[4:2]
```

```
## [1] 4 5 8
## [1] 8 5 4
```

- **vector[-i]**: si $i$ es un número, este subvector está formado por todas las entradas del vector original menos la entrada $i$-ésima. Si $i$ resulta ser un vector, entonces es un vector de índices y crea un nuevo vector con las entradas del vector original,cuyos índices pertenecen a $i$

```r
v[-3]
v[c(2,3)]
v[seq(2,length(v),by = 2)] #obtener los elementos de posición par
v[seq(1,length(v),by = 2)] #obtener los elementos de posición impar
v[-seq(1,length(v),by = 2)]#de esta forma también se obtiene los de pos. par
```

```
## [1]  2  4  8 10 12 15 20 21
## [1] 4 5
## [1]  4  8 12 20
## [1]  2  5 10 15 21
## [1]  4  8 12 20
```

- **vector[-x]**: si $x$ es un vector (de índices), entonces este es el complementario de vector[$x$]. Es decir, aparecerá el vector sin los valores de los índices de x

```
v[-c(2,3)]
```

```
## [1]  2  8 10 12 15 20 21
```

También podemos utilizar **operadores lógicos**:

```
- ‘‘‘==‘‘‘: =
- ‘‘‘!=‘‘‘: $\neq$
- ‘‘‘>=‘‘‘: $\ge$
- ‘‘‘<=‘‘‘: $\le$
- ‘‘‘<‘‘‘: $<$
- ‘‘‘>‘‘‘: $>$
- ‘‘‘!‘‘‘: NO lógico
- ‘‘‘&‘‘‘: Y lógico
- ‘‘‘|‘‘‘: O lógico
```

```
v[v!= 2 & v>4]
```

```
## [1]  5  8 10 12 15 20 21
```

```
v[v>=1 & v<=4]
```

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

Lo se hace los operadores lógicos en vectores es crear un nuevo vector de valores FALSE y TRUE de las posiciones en las que la condición se cumple o no, por lo tanto nos da las entradas del vector original que corresponden a los valores TRUE.

## Condicionales

- **which(x cumple condición)**: para obtener los índices de las entradas del vector $x$ que satisfacen la condición dada
- **which.min(x)**: nos da la primera posición en la que el vector $x$ toma su valor mínimo
- **which(x==min(x))**: da todas las posiciones en las que el vector $x$ toma sus valores mínimos
- **which.max(x)**: nos da la primera posición en la que el vector $x$ toma su valor máximo
- **which(x==max(x))**: da todas las posiciones en las que el vector $x$ toma sus valores máximos

```
which(v>4)
v[which(v>4)]
which.min(v) #posición del primer elemento más pequeño
which.max(v) #posición del primer elemento más pequeño
```

```
## [1] 3 4 5 6 7 8 9
## [1]  5  8 10 12 15 20 21
## [1] 1
## [1] 9
```

**Nota importante**: la función which te da la posición o los índices que ocupan los elementos que cumplan la determinada condición.

## Operaciones extras con vectores

- Modificar los valores de un vector con la sintaxis **vector[i] = x**

```
r = 1:11
r[3] = 2 #reescribir la posición 3 con el número 2
r[4:6] = r[4:6] + 1 #sumarle 1 a los elementos del rango de indice indicados
r[length(r)-2:length(r)] = 0 #hacer 0 los ultimos 3 elementos del vector
```

## Valores NA (Not Available)

Los valores **NA** son valores que no están disponibles. Pueden aparecer por diversos motivos como un error de cálculo, un dato faltante, o simplemente que el dato no existe.

R puede rellenar de valores NA un vector en el que no se le indicaron los valores anteriores a la ultima posición.

```
vector = 1:10

vector[length(vector) + 5] = 9
vector
```

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

Evidentemente algunas operaciones de vectores que ya conocemos como (sum, prod, mean, etc...) no se podrán realizar, para ello utilizaremos el siguiente parámetro:

- **na.rm**: removerá los valores NA a la hora de hacer la operación (no los eliminará del vector)

```
sum(vector, na.rm = TRUE)
prod(vector, na.rm = TRUE)
mean(vector, na.rm = TRUE)
```

```
## [1] 64
## [1] 32659200
## [1] 5.818182
```

### Extraer valores NA de un vector

La función **is.na** nos devolverá los indices cuyas entradas contienen na

```
y = vector
is.na(y) #vector de TRUE y FALSE
which(is.na(y)) #posiciones del vector que son NA
y[which(is.na(y))] #valores que son NA con which

# y[which(is.na(y))] = y [is.na(y)]


vector[is.na(vector)] #Valores del vector que son NA
vector[is.na(vector)] = mean(vector, na.rm = TRUE) #sustituir con la media (calculada sin NA) donde hay
vector[!is.na(vector)] #Valores del vector que no son NA
sum(vector[!is.na(vector)]) #sumar los valores del vector que no sean NA
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
## [13]  TRUE  TRUE FALSE
## [1] 11 12 13 14
## [1] NA NA NA NA
## [1] NA NA NA NA
##  [1]  1.000000  2.000000  3.000000  4.000000  5.000000  6.000000  7.000000
##  [8]  8.000000  9.000000 10.000000  5.818182  5.818182  5.818182  5.818182
## [15]  9.000000
## [1] 87.27273
```

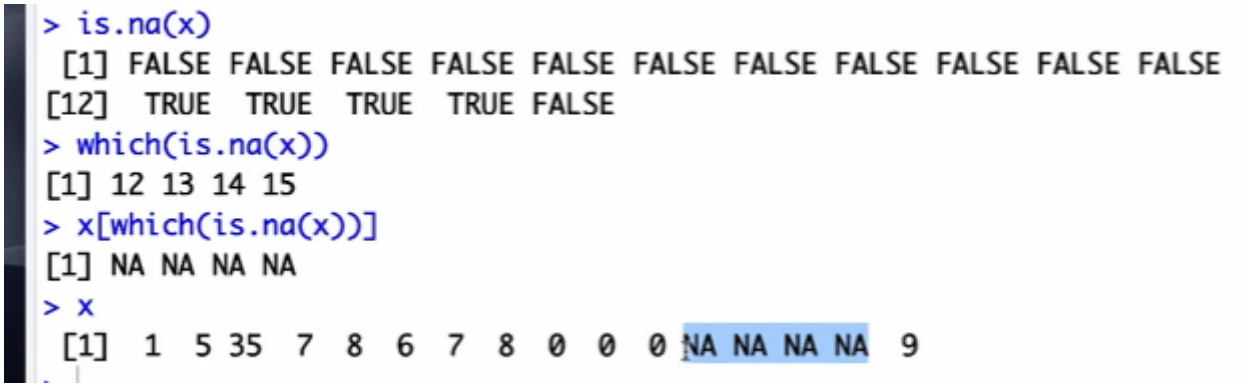


Figure 1: isnaejemplos