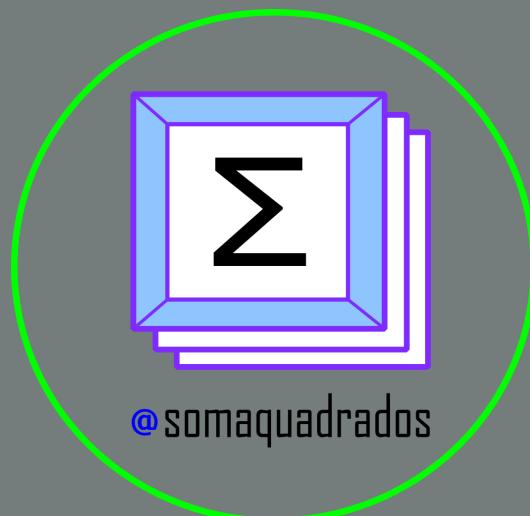


# Programación con R

## Clase 1

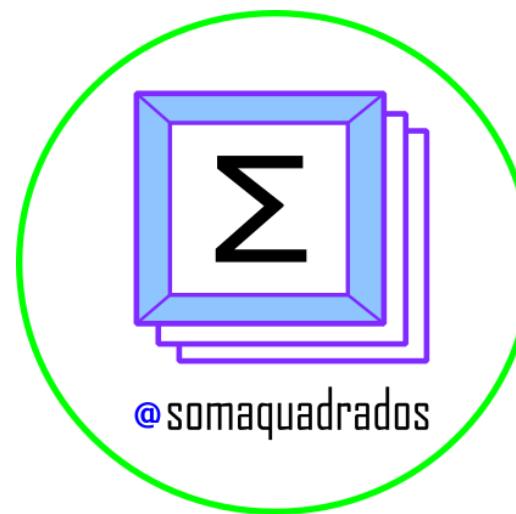


Marília Melo Favalesso

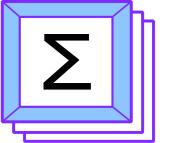
# Bienvenid@!!

## ¡Somos la **Soma dos Quadrados!**

¡Nuestro proyecto tiene como objetivo promover la bioestadística y una programación sencilla para los profesionales del gran área de las ciencias biológicas!

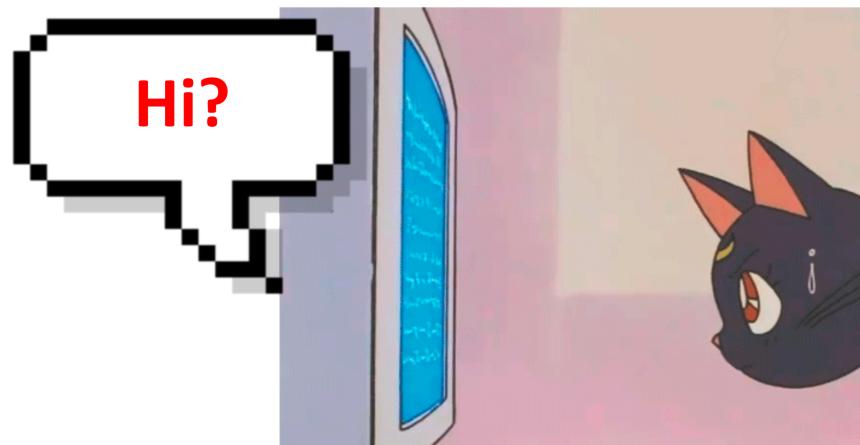


Para más informaciones: <https://www.somaquadrados.com/>

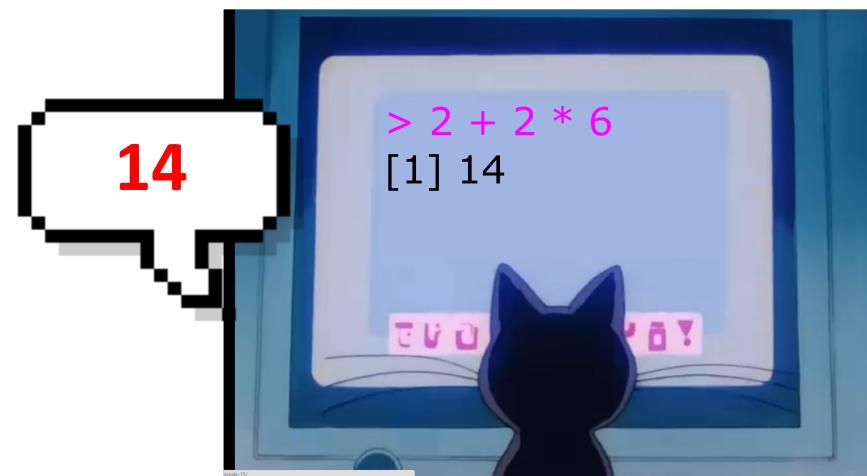
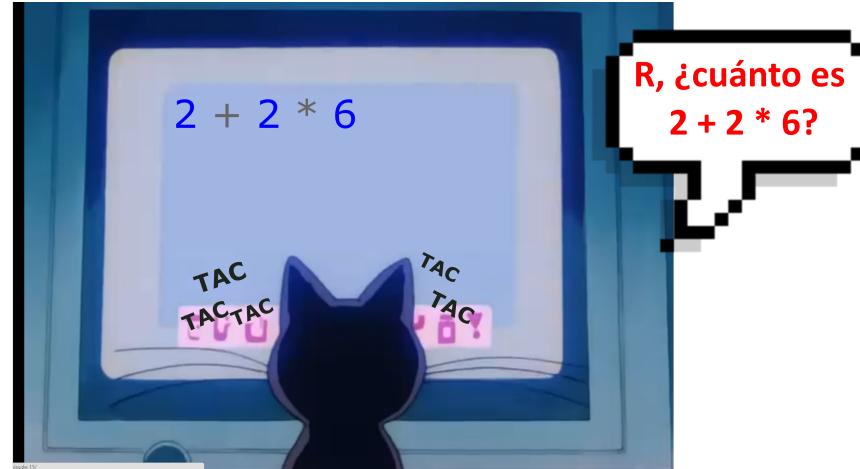


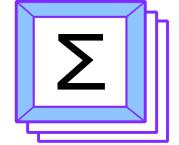
@somaquadrados

# Introducción



# Introducción

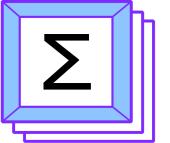




# Introducción

The screenshot shows the RStudio interface with the 'clase\_1.R' script open in the left pane. The code is a series of comments explaining basic R concepts:

```
1 # Introducción al análisis de datos mediante con R - Clase 1
2 #
3 # Sistemas de datos
4 # - tipos y operaciones
5 # - gráficas, manipulación
6 #
7 # Sistemas de control
8 #
9 #
10 # - objetos y visualizaciones
11 #
12 # en los objetos con variables capaces de almacenar cualquier valor
13 # es el valor.
14 #
15 # Guardar el valor 'x' en 'dato' con 'x'
16 # significa que 'dato' = 'x'.
17 # Es decir, se establece el valor en 'dato' mediante comando 'x'.
18 #
19 # El comando 'x' más utilizado = 'lugar de almacenamiento'.
20 # Ejemplo: 'x' = dato
21 # o
22 #
23 # Los comentarios dentro del código son en color lila...
```



@somaquadrados

# Introducción

## En nuestras diapositivas

En RStudio

The screenshot shows the RStudio interface. On the left, the script editor (R Script tab) contains the following code:

```
# donde escribimos y  
# le pedimos a R que  
# ejecute nuestro código  
2 + 2 * 6
```

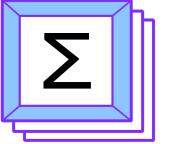
In the bottom-left corner of the script editor, there is a small red callout box containing the text "En la diapositiva".

In the bottom-right corner of the script editor, there is a larger blue callout box containing the following text:

*# donde escribimos y  
# le pedimos a R que  
# ejecute nuestro código  
2 + 2 \* 6*

## [1] 14

En la diapositiva

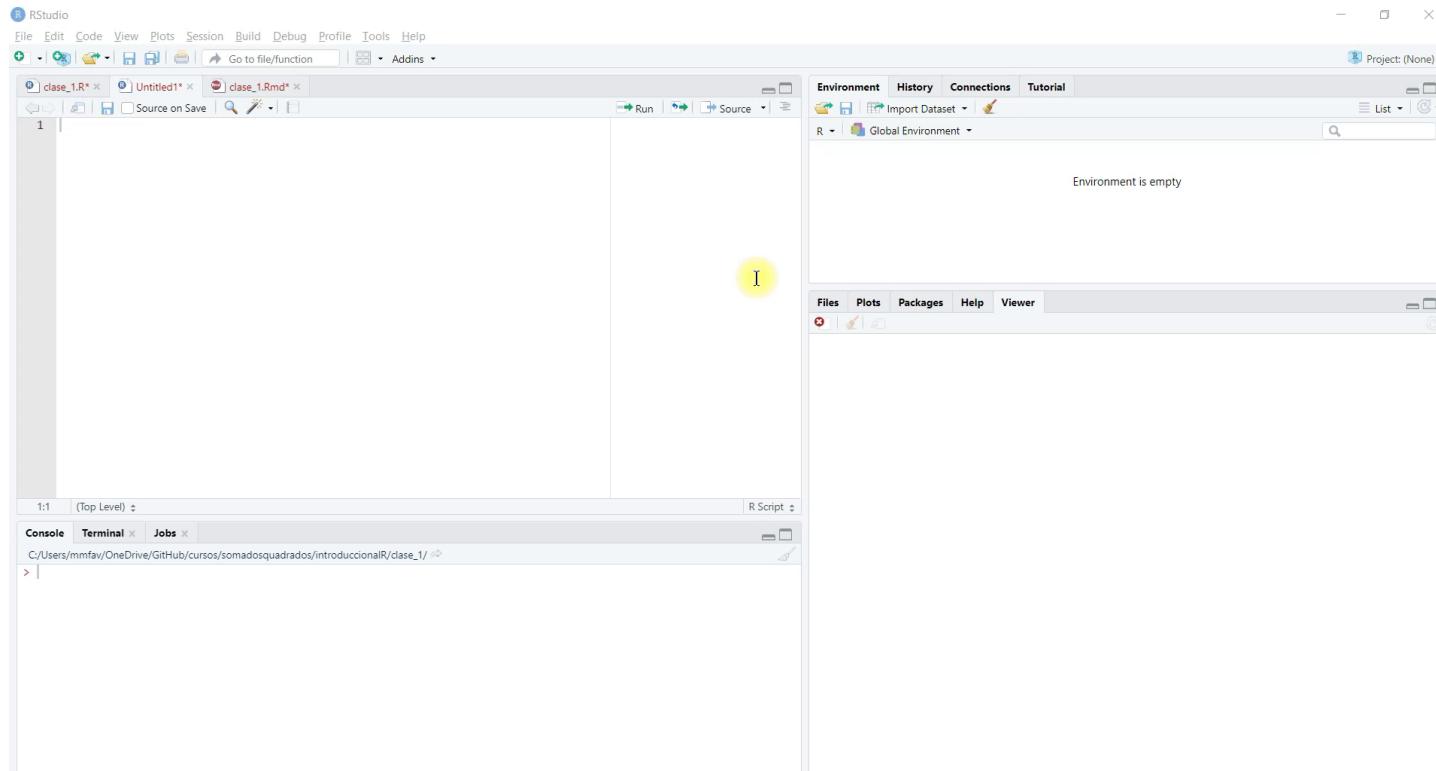


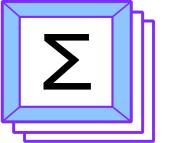
@somaquadrados

# Introducción

## ¿Script mal configurado?

Tools > Global options > Code > Saving > Default text encoding > select UTF-8



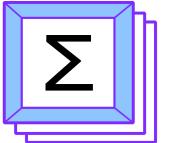


@somaquadrados

# Introducción

## Sistema de puntuación

	Como usamos	Cómo R usa
Separador decimal	0,10	0.10
Separador de miles	1.000	1,000



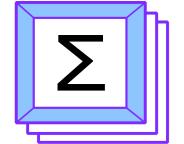
@somaquadrados

# Script

- `clase_1.R`

## Contenido de hoy

- Objetos y atribuciones
- Clases
- Tipos de objetos
- Manejo de datos
- Funciones
- Paquetes

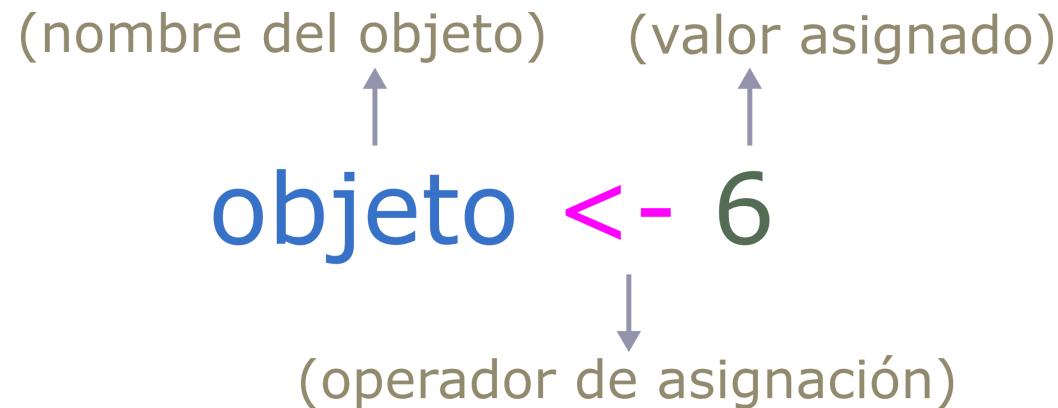


@somaquadrados

# Objetos y atribuciones

# Objetos y atribuciones

- Los **objetos** son variables capaces de almacenar cualquier valor o estructura de datos.



```
objeto <- 6 # Guardamos el valor '6' en 'objeto' con '<-'  
objeto # Siempre que evaluamos el `objeto`, la R devolverá el valor 6
```

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

El símbolo **=** se puede utilizar en lugar de **<-** pero no se recomienda.

# Objetos y atribuciones

Podemos almacenar el valor de un objeto `k` dentro de un objeto `w`:

```
k <- 10  
k
```

```
w <- k  
w
```

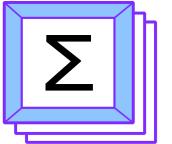
```
## [1] 10 ## [1] 10
```

Podemos usar objetos para realizar operaciones matemáticas...

```
(k + w) / 2  
## [1] 10
```

... y podemos asignar esta operación matemática a un nuevo objeto.

```
j <- (k + w) / 2  
j  
## [1] 10
```



@somaquadrados

# Objetos y atribuciones

!!Cada objeto solo puede almacenar una estructura a la vez (un número o una secuencia de valores)!!

```
a <- 5  
a
```

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

```
a <- 18  
a
```

```
## [1] 18
```

```
a <- 36  
a
```

```
## [1] 36
```

# Reglas para nombrar objetos

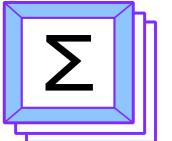
1. Pueden estar formados por *letras, números, "\_" y ":"*
2. **No** se puede empezar con un número y/o un punto
3. **No** puede contener espacios
4. Evitar el uso de acentos
5. Evitar el uso de nombres de funciones como:
  - sum, diff, df, var, pt, data, C, etc
6. La **R** distingue entre mayúsculas y minúsculas, por lo que:
  - obj ≠ Obj ≠ OBJ

```
obj <- 2; obj
```

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

```
OBJ <- 3; OBJ
```

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



@somaquadrados

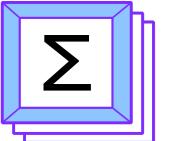
# Reglas para nombrar objetos

## Permitido

```
a <- 5
a1 <- 5
obj <- 10
mi_obj <- 15
mi.obj <-15
```

## No permitido

```
1a <- 1
a 1 <- 5
_obj <-15
mi-obj <- 15
```



@somaquadrados

# Gestionar el lugar de trabajo

## Observación:

La pestaña "**Environmental**" de RStudio muestra los objetos creados en la sesión actual.

The screenshot shows the RStudio environment. In the top-left, there's a code editor with a script named 'clase\_1.Rmd' containing the following R code:

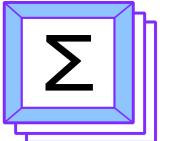
```
1
2
3 a <- 1
4
5 b <- 5
6
7 c <- a + b
8
```

In the bottom-left, the Console tab is active, showing the same R code being run in the terminal:

```
> a <- 1 # CTRL + ENTER
> a <- 1
> b <- a + 2
> c <- a + b + 3
> rm(a)
> a <- 1
> b <- 5
> c <- a + b
>
```

On the right side, the main workspace is visible. The "Environment" tab is highlighted with a red border. It displays the global environment with the following variables and their values:

Values	
a	1
b	5
c	6



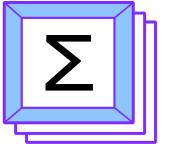
@somaquadrados

# Gestionar el lugar de trabajo

Para eliminar objetos: `rm()`:

```
rm(a, k) # elimina los objetos a y k  
ls() # ¿Qué objetos quedan?
```

```
## [1] "A"          "a1"         "a2"         "a3"         "a4"         "aa"  
## [7] "ar"         "area"       "b"          "B"          "bb"         "c"  
## [13] "ca"        "cb"        "com"       "datos"      "dtf"        "edad"  
## [19] "enfermedad" "estacion"   "j"          "l"          "lis"        "logf"  
## [25] "logt"       "ma"        "mb"        "mes"        "num"        "num_int"  
## [31] "obj"        "OBJ"       "objeto"    "presencia"  "sexo"       "temperatura"  
## [37] "v1"         "v2"        "vc"        "ve"        "vr"        "w"
```



@somaquadrados

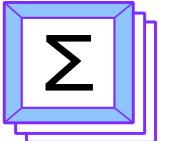
# Clases

# Clases y tipos de objetos

Los **objetos** tienen tres características:

a <- 1

1. **Nombre** que le damos al objeto (= a)
2. **Contenido** en sí del objeto (= 1)
3. **Atributo** del objeto
  - **Clase**: naturaleza del elementos (1 = numerico)
  - **Estructura**: Cómo están organizados los elementos (a = vector)



@somaquadrados

# Clases de objetos

La clase de un objeto es muy importante en R! Es a partir de ella que las funciones y los operadores pueden saber *exactamente* qué hacer con un objeto.

Por ejemplo, es posible sumar dos objetos numéricos,...

```
a <- 1  
b <- 2  
a + b
```

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

... pero no podemos sumar dos caracteres:

```
c <- "c"  
d <- "!"  
c + d
```

```
## simpleError in "c" + "d": argumento no numérico para el operador binario.
```

| R verificó la naturaleza de "c" y "d" y las identificó como no numéricas.

# Clases de objetos

## Objetos atómicos

R tiene 5 clases básicas de objetos, también llamados **objetos atómicos**:

1 - **numeric**: Números reales, punto flotante (decimales).

```
num <- 1.50
```

2 - **integer**: Números enteros (poner un L al final)

```
num_int <- 1L
```

3 - **logical**: booleano (True/False).

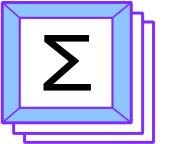
```
logt <- TRUE  
logf <- FALSE
```

4 - **character**: una cadena de caracteres, comúnmente utilizada para representar palabras, frases o texto. Poner entre comillas simple o doble.

```
ca <- "holla!"  
cb <- 'todo bien??'
```

5 - **complex**: Un número con partes reales e imaginarias.

```
com <- 1.5 + 2i
```



@somaquadrados

# Tipos de objetos

## Conversión

- Es posible intentar forzar a un objeto a tener una clase específica:

```
a <- 1; a
```

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

```
class(a)
```

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

---

```
a1 <- as.character(a)
class(a1)
```

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

```
a2 <- as.integer(a)
class(a2)
```

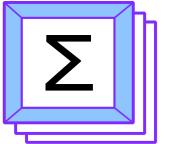
```
## [1] "integer"
```

```
a3 <- as.numeric(a)
class(a3)
```

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

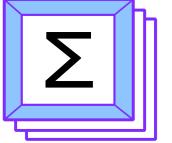
```
a4 <- as.logical(a)
class(a4)
```

```
## [1] "logical"
```



@somaquadrados

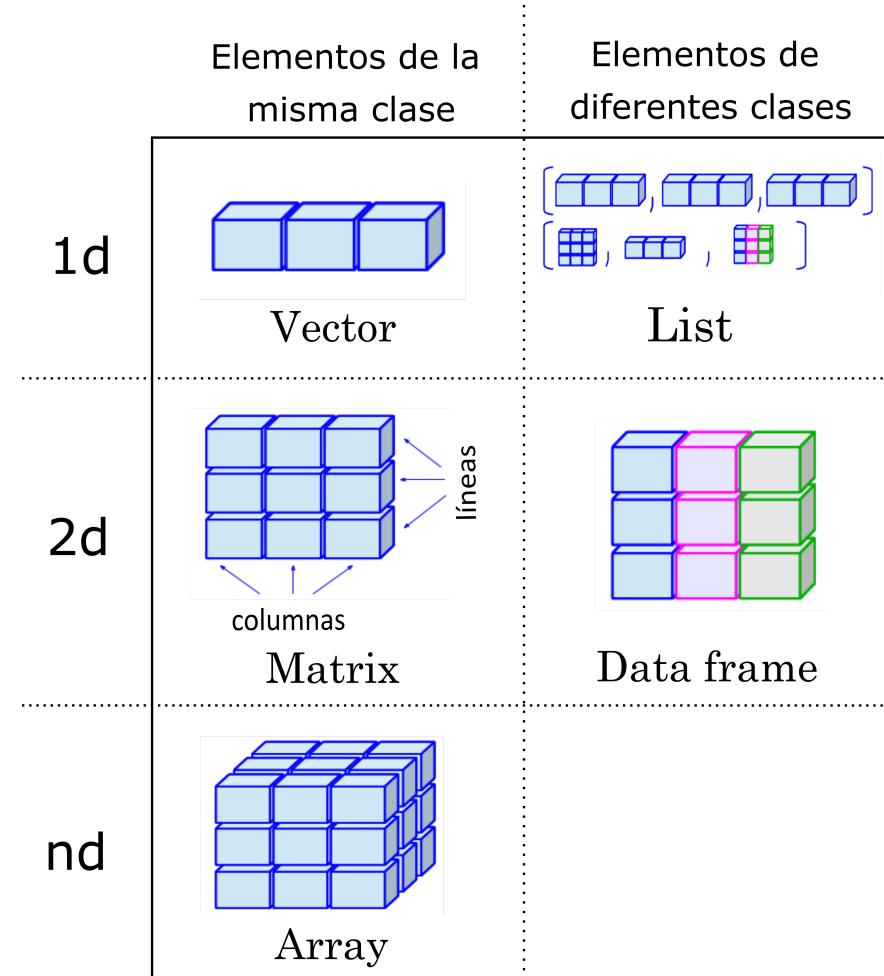
# Tipos de objetos

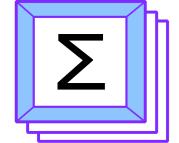


@somaquadrados

# Tipos de objetos

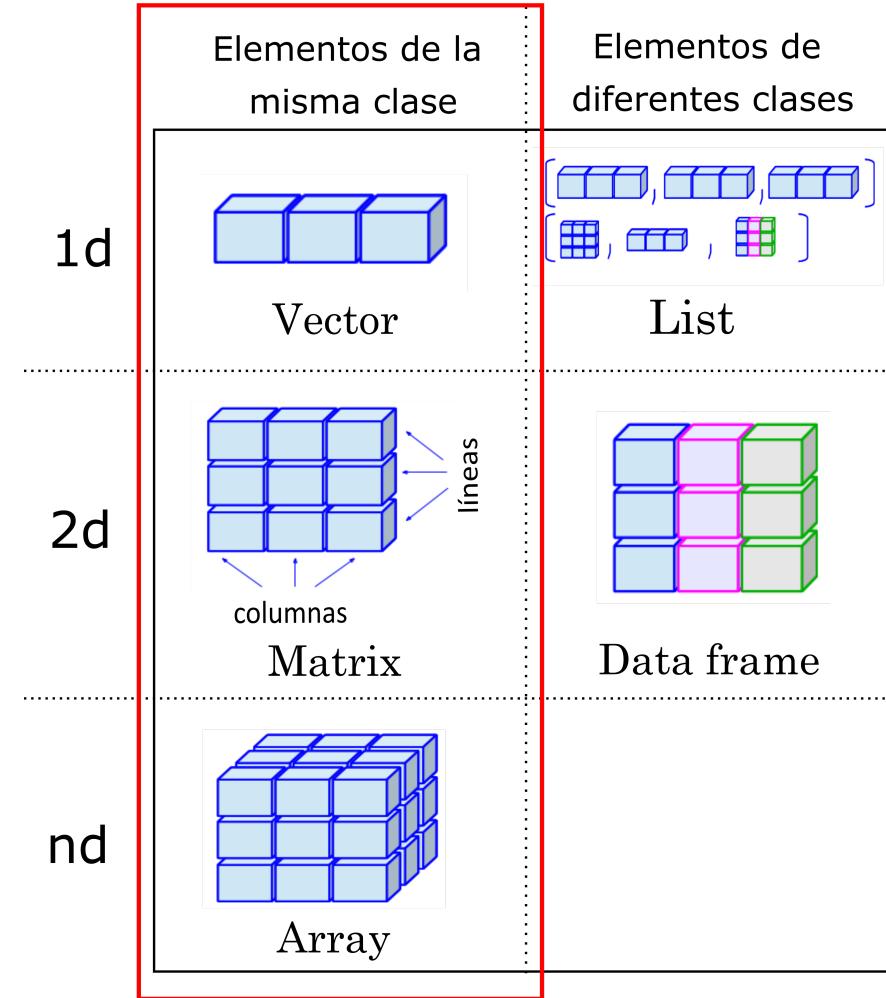
- El tipo del objeto está relacionado con la **clase** y la **estructura/organización**.
- Pueden estar formados por elementos de la misma clase o de clases diferentes.
- Pueden tener de una hasta n dimensiones.
- En **R** tenemos cinco estructuras:
  - **Vector**
  - **Matrix**
  - **Array**
  - **List**
  - **Data frame**

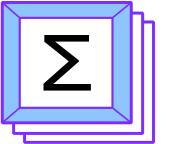




# Tipos de objetos

- El tipo del objeto está relacionado con la **clase** y la **estructura/organización**.
- Pueden estar formados por elementos de la misma clase o de clases diferentes.
- Pueden tener de una hasta n dimensiones.
- En **R** tenemos cinco estructuras:
  - **Vector**
  - **Matrix**
  - **Array**
  - **List**
  - **Data frame**



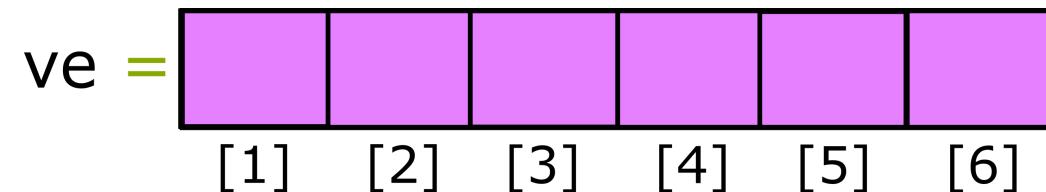


@somaquadrados

# Tipos de objetos

## Vetores

- Colección **unidimensional** de valores:



- Almacena datos de una misma clase.
- La forma más sencilla de crear un **vector** es enumerar los valores separados por comas dentro de una **c()**:

```
area <- c("urb", "rur", "urb", "rur", "urb", "r  
area  
  
## [1] "urb" "rur" "urb" "rur" "urb" "rur"
```

```
temperatura <- c(20, 23, 18, 20, 14, 17)  
temperatura  
  
## [1] 20 23 18 20 14 17
```

# Tipos de objetos

## Vetores

### Coerción

- No es posible mezclar datos de dos clases en un vector.
- Si lo intenta, R exhibirá el comportamiento conocido como **coerción**.

```
aa <- c(1, 2, 3, 4, "a")
class(aa)
```

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

```
bb <- c(1L, 2L, 3.50, 4.1)
class(bb)
```

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

| **DOMINANTE** character > numeric > integer > logical **RECESIVO**

# Tipos de objetos

## Factor

- Colección **unidimensional** de valores.
- Almacena datos de la clase **character**.
- El factor representa medidas de una variable *cualitativa*, que puede ser *nominal* u *ordinal*.

```
estacion <- c("verano", "verano", "primavera", "primavera", "primavera", "otono", "invierno", "invierno")
as.factor(estacion)
```

```
## [1] verano     verano     primavera primavera primavera otono      invierno  invierno
## Levels: invierno otono primavera verano
```

# Tipos de objetos

## Matrix

- Colección **bidimensional** de valores:
  - líneas (por ejemplo, unidades de muestreo)
  - columnas (variables cuantitativas o cualitativas, por ejemplo: horario, tubo de ensayo, ubicación)
- Almacena datos de una única clase.

	col1	col2	col3	col4	col5
linea 1					
linea 2					
linea 3					
linea 4					
linea 5					

columnas

lineas

# Tipos de objetos

## Matrix

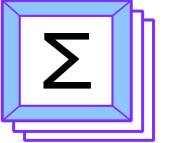
- Colección **bidimensional** de valores:
  - líneas (por ejemplo, unidades de muestreo)
  - columnas (variables cuantitativas o cualitativas, por ejemplo: horario, tubo de ensayo, ubicación)
- Almacena datos de una única clase.

	col1	col2	col3	col4	col5
linea 1					
linea 2					
linea 3					
linea 4					
linea 5					



	T1	T2	T3	T4	T5
Pacie. 1	3,00	2,98	2,68	2,60	2,57
Pacie. 2	4,06	4,00	3,80	3,52	3,00
Pacie. 3	4,12	3,71	3,57	3,49	3,00
Pacie. 4	2,77	2,75	2,71	2,52	2,60
Pacie. 5	2,46	2,68	2,50	2,51	2,4

Volume respiratório forzado (vez)



@somaquadrados

# Tipos de objetos

## Matrix

Para construir matrices en **R**:

Combinación de vectores:

```
# Creamos dos vectores con r
v1 <- c(1, 2, 3); v2 <- c(4, 5, 6)
```

- Combinar vectores por línea - **rbind()**.

```
# Combinamos los vectores verticalmente,
# uno debajo del otro
vr <- rbind(v1, v2)
vr
```

```
##      [,1] [,2] [,3]
## v1      1     2     3
## v2      4     5     6
```

- Combinar vectores por columna - **cbind()**.

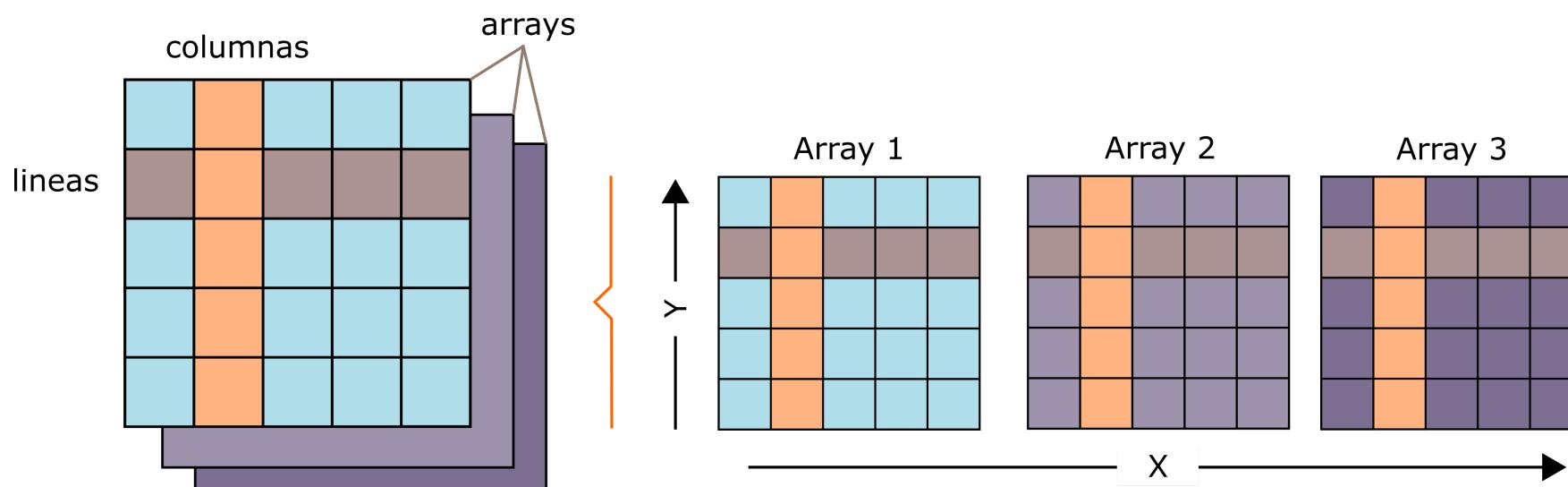
```
# Combinamos los vectores horizontalmente,
# uno al lado del otro.
vr <- cbind(v1, v2)
vr
```

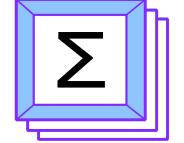
```
##          v1  v2
## [1,]    1  4
## [2,]    2  5
## [3,]    3  6
```

# Tipos de objetos

## Array

- Tiene **n dimensiones** - "varias matrices emparejadas".
- Tiene filas, columnas y dimensiones (**arrays**).
- Almacena datos de una única clase.





@somaquadrados

# Tipos de objetos

## Array

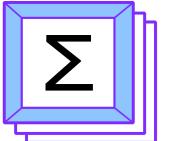
Construir un array en R: `array()`.

```
vc <- 1:8; vc # datos

## [1] 1 2 3 4 5 6 7 8

ar <- array(data = vc, dim = c(2, 2, 2)) # array
ar

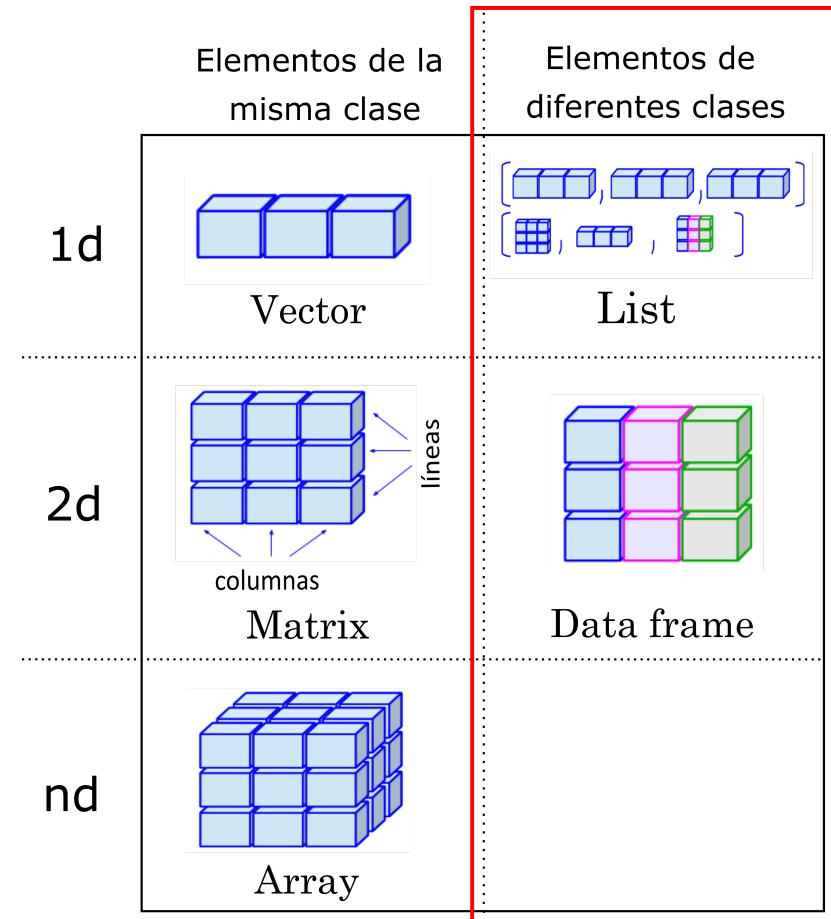
## , , 1
## 
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## , , 2
## 
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
```

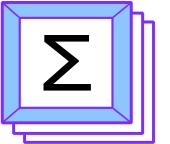


@somaquadrados

# Tipos de objetos

- El tipo del objeto está relacionado con la **clase** y la **estructura/organización**.
- Pueden estar formados por elementos de la misma clase o de clases diferentes.
- Pueden tener de una hasta n dimensiones.
- En **R** tenemos cinco estructuras:
  - **Vector**
  - **Matrix**
  - **Array**
  - **List**
  - **Data frame**





@somaquadrados

# Tipos de objetos

## Data frame

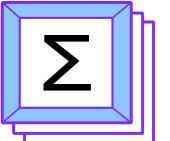
- Colección **bidimensional** de valores:
  - líneas (unidades de muestreo)
  - columnas (variables cuantitativas o cualitativas, por ejemplo: horario, tubo de ensayo, ubicación)
- Almacena datos de ≠ clases.

	Data frame				
	col1	col2	col3	col4	col5
linea 1					
linea 2					
linea 3					
linea 4					
linea 5					

columnas

líneas





# Tipos de objetos

## Data frame

Cómo construir un **data frame** en R: `data.frame()`.

area =   
mes =   
presencia =   
temperatura =



temp	pres	mes	area
20,1	T	1	urb
21,2	T	1	rur
22,2	F	2	urb
23,8	F	2	rur
22,9	T	3	urb
23,0	T	3	rur

# Tipos de objetos

## Data frame

Cómo construir un **data frame** en R: `data.frame()`.

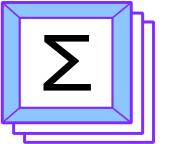
```
# Vamos a crear 4 vectores
area <- c("urb", "rur", "urb", "rur", "urb", "r
mes <- c(1, 1, 2, 2, 3, 3)
presencia <- c(T, T, F, F, T, T)
temperatura <- c(20.1, 21.2, 22.2, 23.8, 22.9,
                area; mes; presencia; temperatura

## [1] "urb" "rur" "urb" "rur" "urb" "rur"
## [1] 1 1 2 2 3 3
## [1] TRUE TRUE FALSE FALSE TRUE TRUE

## [1] 20.1 21.2 22.2 23.8 22.9 23.0
```

```
# Unamos los vetores en un dataframe.
# Observe que cada vector se convierte en una columna
dtf <- data.frame(area, mes, presencia, temperatura)
dtf
```

```
##   area mes presencia temperatura
## 1  urb  1      TRUE      20.1
## 2  rur  1      TRUE      21.2
## 3  urb  2     FALSE      22.2
## 4  rur  2     FALSE      23.8
## 5  urb  3      TRUE      22.9
## 6  rur  3      TRUE      23.0
```

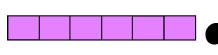
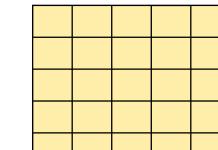
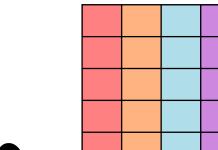
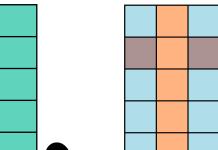
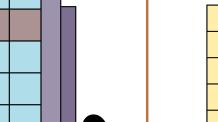
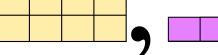


@somaquadrados

# Tipos de objetos

## List

- Colección **unidimensional** de objetos.
- Almacena datos de ≠ tipos (**vectors, arrays, data frame, lists**).
- Es un vector especial que acepta objetos como elementos.
  - Muchas funciones que usamos para analizar datos en R tienen listas como salida.

```
mi_lista = [  ,  ,  ,  ,  ,  ]
```

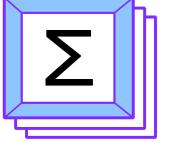
# Tipos de objetos

## List

crea una **lista** en r: `list()`.

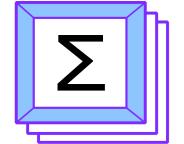
```
lis <- list(vr,
            temperatura,
            factor(c("a", "a", "b", "c")))
lis
```

```
## [[1]]
##      v1 v2
## [1,]  1  4
## [2,]  2  5
## [3,]  3  6
##
## [[2]]
## [1] 20.1 21.2 22.2 23.8 22.9 23.0
##
## [[3]]
## [1] a a b c
## Levels: a b c
```



@somaquadrados

# Manejo de datos

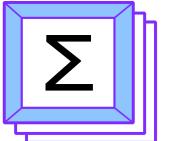


@somaquadrados

# Manejo de datos

- Los objetos son conjuntos *indexados* - Esto nos permite acceder a cada elemento de manera individual.
- Comprender la indexación es fundamental para manipular datos en **R**.
- Usamos corchetes (`[]`) para acceder a la posición de los elementos de un objeto:
  - **[L, C, D]**.

dimensiones	objetos	indexación
1d	 lineas (L)	vector[L]
2d	 lineas (L) vs. columnas (c)	tabla[L,C]
nd	 lineas (L) vs. columnas (c) vs. dimensión (D)	array[L,C,D]



@somaquadrados

# Manejo de datos

## Indexación

### Gestión de datos unidimensionales [L]

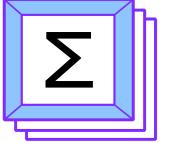
- Seleccionar elementos

ve =	11	12	13	14	15	16
	[1]	[2]	[3]	[4]	[5]	[6]

- No R:

```
ve <- c(11, 12, 13, 14, 15, 16)
ve
```

```
## [1] 11 12 13 14 15 16
```



@somaquadrados

# Manejo de datos

## Indexación

### Gestión de datos unidimensionales [L]

- Seleccionar elementos

ve =	11	12	13	14	15	16
	[1]	[2]	[3]	[4]	[5]	[6]

- No R:

```
ve <- c(11, 12, 13, 14, 15, 16)  
ve
```

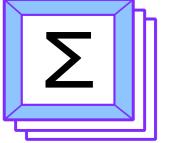
```
## [1] 11 12 13 14 15 16
```

```
ve[3]
```

```
## [1] 13
```

```
ve[2:4]
```

```
## [1] 12 13 14
```



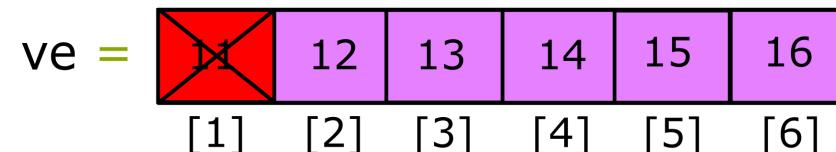
@somaquadrados

# Manejo de datos

## Indexación

### Gestión de datos unidimensionales [L]

- Quitar elementos



- No R:

```
ve <- c(11, 12, 13, 14, 15, 16)  
ve
```

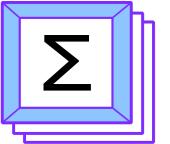
```
## [1] 11 12 13 14 15 16
```

```
ve[-1]
```

```
## [1] 12 13 14 15 16
```

```
ve[-c(2, 5)]
```

```
## [1] 11 13 14 16
```



# Manejo de datos

## Indexación

### Gestión de datos bidimensionales [L,C]

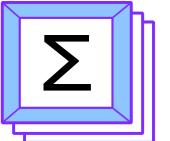
- Seleccionar elementos

		columnas (C)				
		col1	col2	col3	col4	col5
lineas (L)	linea 1	[1,1]	[1,2]	[1,3]	[1,4]	[1,5]
	linea 2	[2,1]	[2,2]	[2,3]	[2,4]	[2,5]
	linea 3	[3,1]	[3,2]	[3,3]	[3,4]	[3,5]
	linea 4	[4,1]	[4,2]	[4,3]	[4,4]	[4,5]
	linea 5	[5,1]	[5,2]	[5,3]	[5,4]	[5,5]

**tabla[L,C]**

		columnas (C)				
		col2	3	4	5	
lineas (L)	linea 1	1	2	3	4	5
	linea 2	6	7	8	9	10
	linea 3	11	12	13	14	15
	linea 4	16	17	18	19	20
	linea 5	21	22	23	24	25

**tabla[1,2]**



# Manejo de datos

## Indexación

### Gestión de datos bidimensionales [L,C]

- Seleccionar elementos

```
ma <- matrix(data = (c(1:25)), nrow = 5, ncol = 5, byrow = TRUE) # crear una matriz  
ma
```

```
##      [,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  
## [5,]    21   22   23   24   25
```

```
ma[1,2] # seleccionar el valor de la línea uno y la segunda columna.
```

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

# Manejo de datos

## Indexación

### Gestión de datos bidimensionales [L,C]

- Seleccionar elementos

```
ma[3:4, c(3, 5)] # seleccione las lineas 3 y 4 y las columnas 3 y 5
```

```
##      [,1] [,2]
## [1,]    13   15
## [2,]    18   20
```

```
ma[c(1,5), 3:5] # seleccione las lineas 1 y 5 y las columnas entre 3 - 5
```

```
##      [,1] [,2] [,3]
## [1,]    3    4    5
## [2,]   23   24   25
```

Es posible seleccionar más de una fila y columna al mismo tiempo.

# Manejo de datos

## Indexación

### Gestión de datos bidimensionales [L,C]

- Quitar elementos

#### Antes

```
ma  
##      [,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  
## [5,]   21   22   23   24   25
```

#### Después

```
ma[-2, -2] # menos la fila dos y la columna dos  
##      [,1] [,2] [,3] [,4]  
## [1,]    1    3    4    5  
## [2,]   11   13   14   15  
## [3,]   16   18   19   20  
## [4,]   21   23   24   25
```

# Manejo de datos

## Indexación

### Gestión de datos bidimensionales [L,C]

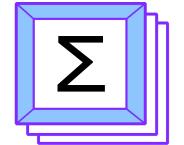
- Quitar elementos

#### Antes

```
ma  
  
##      [,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  
## [5,]    21   22   23   24   25
```

#### Después

```
ma[-2:-3, -c(1,5)]  
  
##      [,1] [,2] [,3]  
## [1,]     2    3    4  
## [2,]    17   18   19  
## [3,]    22   23   24
```



@somaquadrados

# Manejo de datos

## Indexación

### Gestión de datos bidimensionales [L,C]

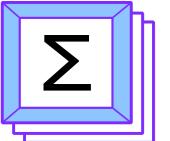
- También podemos usar el nombre de las filas y columnas para manejar los datos: [nombre\_linea, nombre\_columna].

```
rownames(ma) <- c("l1", "l2", "l3", "l4", "l5") # nombre en las lineas  
colnames(ma) <- c("c1", "c2", "c3", "c4", "c5") # nombre en las columnas  
ma # la tabla
```

```
##      c1 c2 c3 c4 c5  
## l1  1  2  3  4  5  
## l2  6  7  8  9 10  
## l3 11 12 13 14 15  
## l4 16 17 18 19 20  
## l5 21 22 23 24 25
```

```
ma["l1", "c2"] # selecciona linea 2 y columna 3
```

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



@somaquadrados

# Manejo de datos

## Indexación

### Gestión de *data frame* \$:

El operador `$` se utiliza para extraer elementos de una columna com nombre:

```
mb <- data.frame(ma)
class(mb)

## [1] "data.frame"

mb$c1 # columna 1

## [1] 1 6 11 16 21

mb$c2 # columna 2

## [1] 2 7 12 17 22
```

# Manejo de datos

## Indexación

Gestión de *data frame* con `$`:

- agregar una nueva columna:

```
# después
mb$ID <- 1:5
mb

##      c1  c2  c3  c4  c5 ID
##  l1   1   2   3   4   5   1
##  l2   6   7   8   9  10   2
##  l3  11  12  13  14  15   3
##  l4  16  17  18  19  20   4
##  l5  21  22  23  24  25   5
```

# Manejo de datos

## Seleccionar elementos por condición (**vector**)

¿Algún valor en el vector es igual a 11?

11	12	13	11	15	16
[1]	[2]	[3]	[4]	[5]	[6]

== 11?

resposta

T	F	F	T	F	F
---	---	---	---	---	---

{ T = verdadero  
F = falso

Operadores relacionales con salidas **booleanas** (VERDADERO o FALSO).

# Manejo de datos

## Seleccionar elementos por condición (vector)

A == B igual

A != B no igual

A < B menor que

A <= B menor o igual que

A > B mayor que

A >= B mayor o igual que

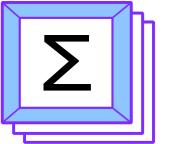
A | B o

A ! B no

A %in% B en el conjunto

Comparación de objetos: A con B.

Operadores relacionales con salidas **booleanas** (VERDADERO o FALSO).



@somaquadrados

# Manejo de datos

## Seleccionar elementos por condición (**vector**)

- Operadores relacionales con salidas booleanas (VERDADERO o FALSO).

```
A <- 6; B <- 28
```

```
A == B # A es igual a B?
```

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

```
A != B # A es distinto de B?
```

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

```
A > B # A es mayor que B?
```

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

```
A <= B # A menor o igual que?
```

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

```
A < B # A es menor que B?
```

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

```
A %in% B # A en B?
```

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

# Manejo de datos

## Seleccionar elementos por condición (**vector**)

- Operadores relacionales con salidas **booleanas** (VERDADERO o FALSO).

```
ve <- c(10, 15, 30, 32, 50, 68, 70)  
ve
```

```
## [1] 10 15 30 32 50 68 70
```

```
# ¿Qué elementos tienen el valor = 30?  
ve == 30
```

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

```
# ¿Qué elementos tienen un valor superior a 30?  
ve > 30
```

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

```
# ¿Qué elementos tienen el valor inferior a 50?  
ve < 50
```

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

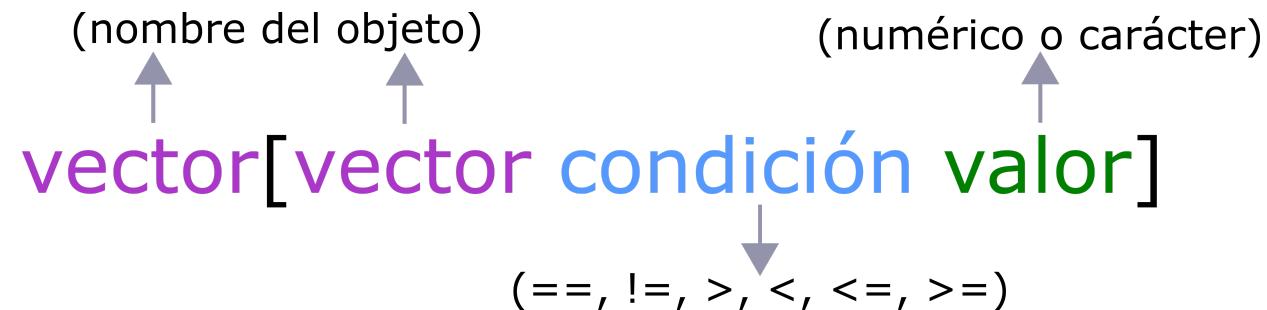
```
# ¿Qué elementos tienen valores mayores o igual  
ve >= 45
```

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

# Manejo de datos

## Seleccionar elementos por condición (**vector**)

- Elementos del vector
  - ¿Qué pasa si, en lugar de querer saber cuál valor coincide con la condición y cuál no, quisiera seleccionar los valores relacionados con esa condición?



```
# Antes:  
ve < 30
```

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

```
# Despues:  
ve[ve < 30]
```

```
## [1] 10 15
```

# Manejo de datos

## Seleccionar elementos por condición (**vector**)

- Elementos del vector

```
# ¿Qué valor/es es igual a 30?  
ve[ve == 30]
```

```
## [1] 30
```

```
# ¿Qué valor/es valen menos de 50?  
ve[ve < 50]
```

```
## [1] 10 15 30 32
```

```
# ¿ve esta inserido en el conjunto 32?  
ve[ve %in% 32]
```

```
## [1] 32
```

```
# ¿Qué valor/es es/son superior/es a 30?  
ve[ve > 30]
```

```
## [1] 32 50 68 70
```

```
# ¿Qué valor/es es/son mayores o iguales a 45?  
ve[ve >= 45]
```

```
## [1] 50 68 70
```

```
# ¿Qué valor/es es/son distintos de 10?  
ve[ve != 10]
```

```
## [1] 15 30 32 50 68 70
```

# Manejo de datos

## Seleccionar elementos por condición (matrix/data.frame)

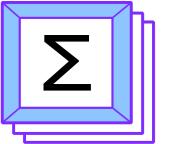
¿Algún valor en la matrix/data.frame es igual a 11?

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

`== 11?` — respuesta →

F	F	F	F	F
F	F	F	F	F
T	F	F	F	F
F	F	F	F	F
F	F	F	F	F

Operadores relacionales con salidas **booleanas** (VERDADERO o FALSO).



@somaquadrados

# Manejo de datos

## Seleccionar elementos por condición (matrix/data.frame)

- Operadores relacionales con salidas **booleanas** (VERDADERO o FALSO).

```
ma <- matrix(c(1:12), nrow = 3)
ma
```

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

```
# ¿Qué elementos tienen los valores
# = 5 o 6?
ma == c(5, 6)
```

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

```
# ¿Qué elementos tienen el valor
# inferior a 8?
ma < 8
```

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

# Manejo de datos

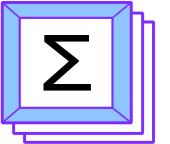


## Seleccionar elementos por condición (matrix/data.frame)

- Elementos de la **matrix/data.frame**.
    - ¿Qué pasa si, en lugar de querer saber cuál valor coincide con la condición y cuál no, quisiera seleccionar los valores relacionados con esa condición?

```
ma[ma == 8]
```

## [1] 8



# Manejo de datos

## Seleccionar elementos por condición (**matrix/data.frame**)

- Operadores relacionales con salidas **booleanas** (VERDADERO o FALSO).

```
# ¿La matriz contiene los valores 5, 7 y 15?  
ma[ma == c(5, 7, 15)]
```

```
## integer(0)
```

```
# ¿Está la matriz contenida en c(0, 2, 6, 18)?  
ma[ma %in% c(0, 2, 6, 18)]
```

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

```
# ¿La matriz es diferente de 5?  
ma[ma != 5]
```

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

# Manejo de datos



# Seleccionar elementos por condición (data.frame)

```
data.frame[data.frame$col condición valor,]  
          ↑  
          (nombre del objeto)  
          ↓  
          (seleccionar columna)  
          ↑  
          (==, !=, >, <, <=, >=)  
          ↓  
          (numérico o carácter)  
          ↑  
          (coma)
```

- Podemos usar `$` y `selección por condición` para seleccionar solo las filas de un `data.frame` que coinciden con una condición.

# Manejo de datos



# Seleccionar elementos por condición (data.frame)

```
edad <- c(20, 25, 30, 45)
sexo <- c("F", "F", "M", "M")
enfermedad <- c(T, T, F, F)

datos <- data.frame(edad, sexo, enfermedad)
datos
```

```

##     edad sexo enfermedad
## 1    20   F      TRUE
## 2    25   F      TRUE
## 3    30   M     FALSE
## 4    45   M     FALSE

```

```
data.frame[data.frame$col condición valor,]  
          ↑           ↑           ↑  
  (nombre del objeto)  (==, !=, >, <, <=, >=)  (coma)  
          ↓           ↓           ↓  
  (seleccionar columna)  (numérico o carácter)
```

```
datos[datos$edad <= 30, ]
```

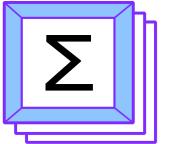
```
##     edad sexo enfermedad
## 1    20   F      TRUE
## 2    25   F      TRUE
## 3    30   M     FALSE
```

```
datos[datos$sexo == "F",]
```

```
##      edad sexo enfermedad
## 1    20   F     TRUE
## 2    25   F     TRUE
```

```
datos[datos$enfermedad == T,]
```

```
##      edad sexo enfermedad
## 1    20     F      TRUE
## 2    25     F      TRUE
```

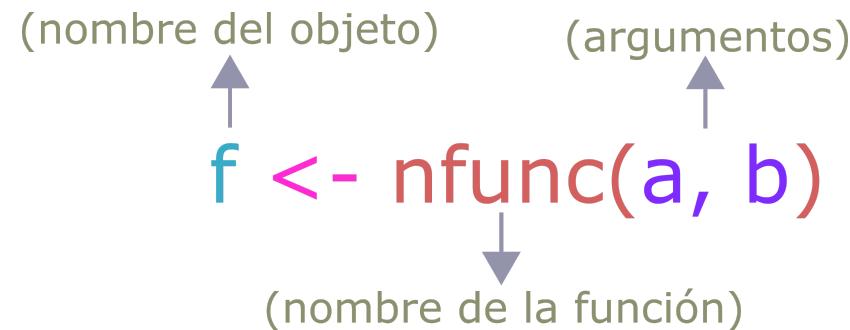


@somaquadrados

# Funciones

# Funciones

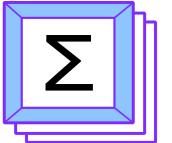
- Mientras que los **objetos** son *nombres que contienen valores*, las **funciones** son *nombres que contienen un código R*.



- La idea básica de una función es encapsular un código que se pueda invocar en cualquier momento en R.

`nfunc(a,b)` → {code} → resultado  
(ejecutar código)

Usamos algunas funciones hasta ahora: `c()`, `rep()`, `data.frame()`, `class()`, otros.



@somaquadrados

# Funciones

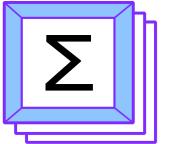
## Argumentos

- Las funciones toman **argumentos**.
- Los argumentos son los valores u objetos que ponemos entre paréntesis y que las funciones necesitan para funcional (calculando un resultado).
- Por ejemplo, la función `class()` necesita recibir un objeto para investigar la clase y devolverlo:

```
a <- 3  
class(a)
```

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

| En este caso, "a" es el argumento que incluimos en la función `class()`.



@somaquadrados

# Funciones

## Argumentos

- Para las funciones que toman más de un argumento, tenemos que separar los argumentos con comas.
- Por ejemplo, cuando usamos la concatenación (`c()`) para crear un **vector**.

```
ve <- c(1, 2, 3, 4)
```

### Importante:

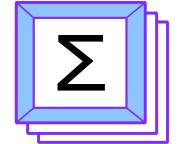
!! Observe cómo debe ser la entrada de valores para que funcione la función.

```
class(1, 2, 3, 4)
```

```
# simpleError in class(1, 2, 3, 4): 4 argumentos passados para 'class', que requer 1
```

```
class(ve)
```

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



@somaquadrados

# Funciones

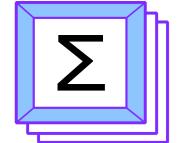
## Argumentos

Los argumentos de las funciones también tienen **nombre**, que pueden o no ser usando en la función. Por ejemplo a función `rep()`.

```
rep(x = c(1, 3),  
     times = 2)
```

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

- **x**: valores que se repetirán.
- **times**: Un vector de valor entero que da el número (no negativo) de veces que se repite cada elemento si tiene una longitud (x), o que se repite todo el vector si tiene una longitud 1.

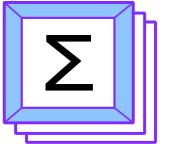


@somaquadrados

# Funciones

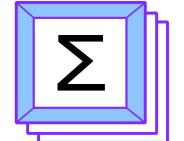
- ¿Existe una función lista para mi problema?
- ¿Cómo averiguar el nombre de esta función?





@somaquadrados

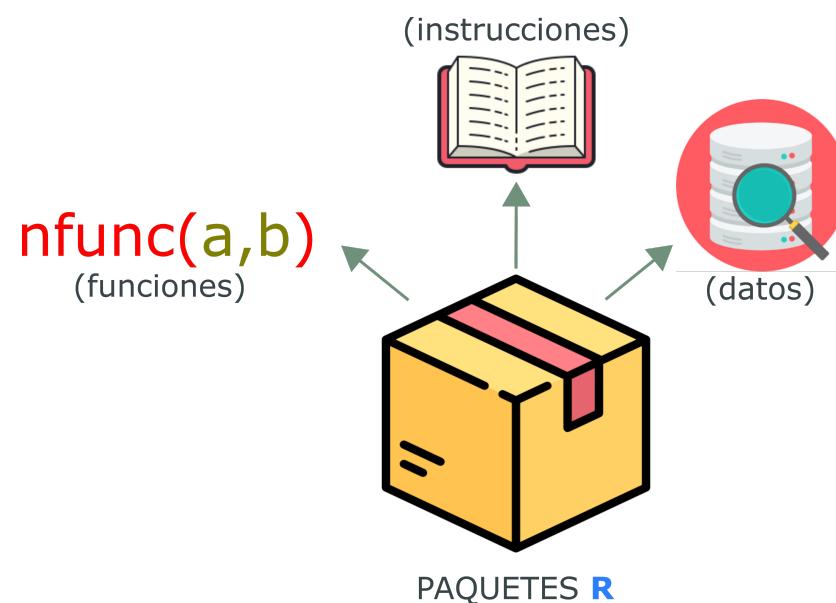
# Paquetes



@somaquadrados

# Paquetes

- Las funciones provienen de dos fuentes:
  1. paquetes **R** estándar que se cargan siempre que trabajamos con el lenguaje
  2. paquetes que instalamos y cargamos por comandos.
- Básicamente, un paquete es una convención para organizar y estandarizar la distribución de funciones **R**.



# Paquetes

- La principal motivación de crear un paquete **R** es de organizar y compartir funciones de nuevos métodos y/o implementaciones creadas y que son útiles para otras personas.
- En general, descargamos paquetes de dos fuentes: **CRAN** y **GitHub**.



The Comprehensive R  
Archive Network

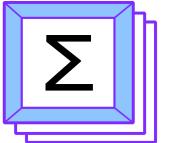
(paquetes listos  
para usar)

# GitHub



(paquetes en  
desarrollo)

# Paquetes



@somaquadrados

## Instalación

- Para instalar paquetes desde **CRAN** usamos el comando `install.packages("nombre_paquete")`.

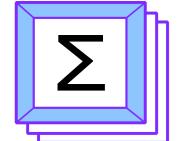
```
install.packages("ggplot2") # Para instalar el paquete "ggplot2"
```

\* Tenga en cuenta que el nombre del paquete siempre debe ir entre comillas para la instalación.

- Compruebe si el paquete se ha instalado:

```
library()
```

abre una nueva pestaña en R escrita  
"Paquetes R disponibles".



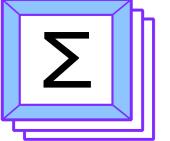
@somaquadrados

# Paquetes

## Instalación

- Para instalar paquetes de **Github**, usamos el paquete `devtools`:  
`install_github("direccion/nombre_paquete")`.
- Para hacer esto, necesitaremos la dirección y el nombre del paquete de un repositorio de GitHub (<https://github.com/tidyverse/dplyr>)

```
# Instalar el paquete 'devtools'  
install.packages("devtools")  
  
# Cargar el paquete para su uso  
library(devtools)  
  
# Incluir la dirección de descarga  
# del paquete do github en install_github()  
install_github("tidyverse/dplyr")
```



@somaquadrados

# Paquetes

## Instalación

- Para instalar paquetes de **Github**, usamos el paquete **devtools**:

tidyverse/ggplot2: An implement x +

github.com/tidyverse/ggplot2

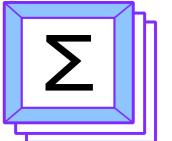
Library Genesis Localiza&ccedil;o Ionomia Banco de dados Epidemiologia casamento Modelagem de nicho Outline - Read & a... GLMM\_Ionomia

☰ README.md

ggplot2 is a system for declaratively creating graphics, based on [The Grammar of Graphics](#). You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

## Installation

```
# The easiest way to get ggplot2 is to install the whole tidyverse:  
install.packages("tidyverse")  
  
# Alternatively, install just ggplot2:  
install.packages("ggplot2")  
  
# Or the development version from GitHub:  
# install.packages("devtools")  
devtools::install_github("tidyverse/ggplot2")
```



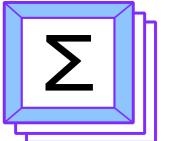
@somaquadrados

# Paquetes

## Instalación

- Solo instalamos los paquetes una vez.
- Los paquetes se descargan a través de la internet.
- El nombre del paquete debe estar entre comillas ("paquete\_nombre"), independientemente de si lo vamos a descargar de [CRAN](#) o [GitHub](#).
- Para cargar paquetes en R usamos la función `library(paquete_nombre)`.
  - En este caso no es necesario incluir comillas.
  - Cargamos paquetes para usar sus funciones.

```
library(ggplot2)
library(dplyr)
```



@somaquadrados

# Paquetes

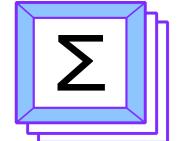
## Actualización

- Los paquetes no se actualizan solos.
- Es necesario actualizarlos de vez en cuando.
- ¡Es un proceso que lleva tiempo!

```
update.packages(ask = FALSE)
```

## Dirección en mi compu

- ¿Dónde están los paquetes?
  - Windows: `C:/Users/nombre_del_compu/Documentación/R/win-library/versión_r`
  - Unix (Linux o MacOS): `/home/nombre_del_compu/R/tipo_compu/versión_r`



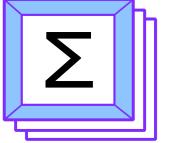
@somaquadrados

# Paquetes

## Paquetes útil para empezar! (clase mañana)

- **tidyverse**
  - es una colección obstinada de paquetes R diseñados para la ciencia de datos.



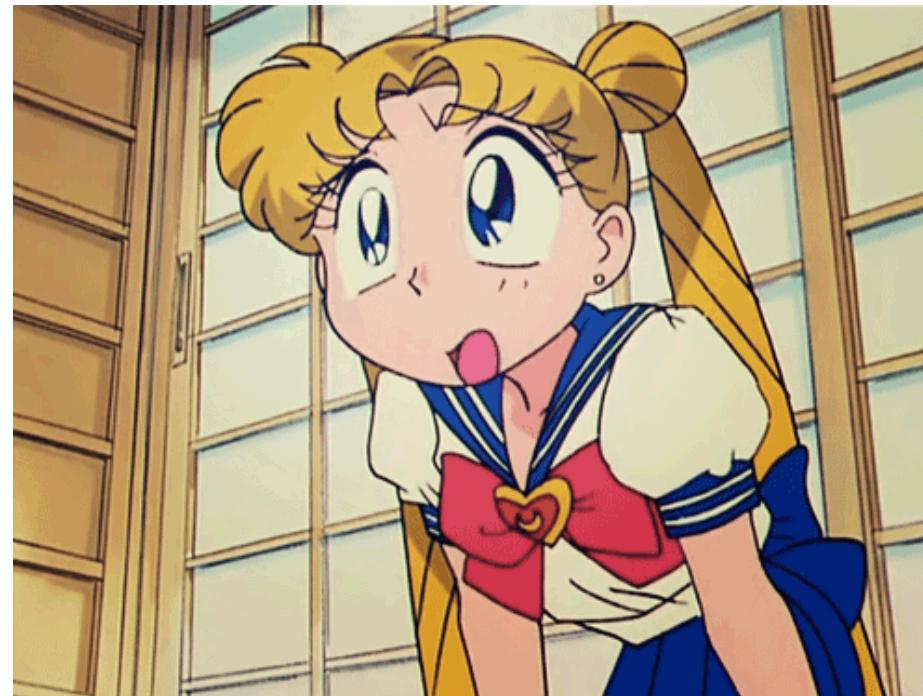


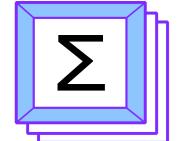
@somaquadrados

# Paquetes

## Cantidad de paquetes disponibles

```
nrow(available.packages(repos = "http://cran.r-project.org"))  
## [1] 18290
```





@somaquadrados

# Paquetes

## Help! (ayuda)

- El "help" de R es muy útil cuando necesitamos ayuda para comprender una función.

```
help("sum")
# es necesario encerrar el nombre de
# la función entre comillas.
```

The screenshot shows the RStudio interface. In the top-left, there's a code editor window with the following R code:

```
1
2
3 help("sum")
4
```

The third line, `help("sum")`, is highlighted with a red box. In the bottom-left, the R Console shows the command being run:

```
> help("sum")
```

In the top-right, the Help Viewer is open. It has tabs for File, Plots, Packages, Help, and Viewer. The Help tab is selected, and the title is "R: Sum of Vector Elements". A red box highlights the entire content area of the Help Viewer, which contains the function's documentation:

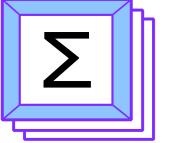
**Sum of Vector Elements**

**Description**  
sum returns the sum of all the values present in its arguments.

**Usage**  
sum(..., na.rm = FALSE)

**Arguments**  
... numeric or complex or logical vectors.  
na.rm logical. Should missing values (including NaN) be removed?

**Details**  
This is a generic function: methods can be defined for it directly or via the `Summary` group generic. For this to work properly, the arguments ... should be unnamed, and dispatch is on the first argument.  
If `na.rm` is `TRUE` an `NA` or `NaN` value in any of the arguments will cause a value of `NA` or `NaN` to be returned, otherwise `NA` and `NaN` values are ignored.  
Logical true values are regarded as one, false values as zero. For historical reasons, `NULL` is accepted and treated as if it were `integer(0)`.  
Loss of accuracy can occur when summing values of different signs: this can even occur for sufficiently long integer inputs if the partial sums would cause integer overflow. Where possible extended-precision accumulators are used, typically well-supported with C99 and newer but



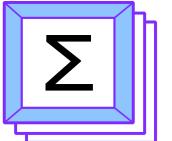
@somaquadrados

# Paquetes

## ¿Cómo citar la R?

```
citation()

##
## To cite R in publications use:
##
##   R Core Team (2021). R: A language and environment for statistical computing. R
##   Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {R: A Language and Environment for Statistical Computing},
##     author = {{R Core Team}},
##     organization = {R Foundation for Statistical Computing},
##     address = {Vienna, Austria},
##     year = {2021},
##     url = {https://www.R-project.org/},
##   }
##
## We have invested a lot of time and effort in creating R, please cite it when using it for
```



@somaquadrados

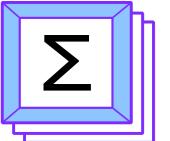
# Paquetes

## ¿Cómo citar un paquete?

```
citation("ggplot2")

##
## To cite ggplot2 in publications, please use:
##
##   H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York,
##   2016.

##
## A BibTeX entry for LaTeX users is
##
##   @Book{,
##     author = {Hadley Wickham},
##     title = {ggplot2: Elegant Graphics for Data Analysis},
##     publisher = {Springer-Verlag New York},
##     year = {2016},
##     isbn = {978-3-319-24277-4},
##     url = {https://ggplot2.tidyverse.org},
##   }
```

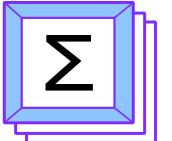


@somaquadrados

# ¡Es todo por hoy!

## ¡Gracias por tu presencia!





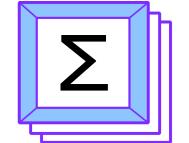
@somaquadrados

# Si tenemos tiempo



## Resuelve los siguientes ejercicios

1. Cree un vector que contenga tres especies de mamíferos y guárdelo en un objeto.
2. Cree una matriz de datos con valores aleatorios entre 0 y 100 (tip: `sample()`). La matriz debe contener 3 filas y 4 columnas.
3. Nombra las líneas de tu matriz con el vector que creaste en el [ejercicio 1](#). Las columnas deben contener el nombre de las estaciones.
  1. Transforme su matriz en un data frame.
  2. ¿Qué especie fue más abundante en primavera? ¿Y cuál fue el menos abundante en invierno?



@somaquadrados

# Soma dos quadrados

- [Soma-Dos-Quadrados/introductioR](https://github.com/Soma-Dos-Quadrados/introductioR)
- [/somaquadrados](https://www.youtube.com/@somaquadrados)
- [/somadosquadrados](https://discord.com/invite/somadosquadrados)
- [@somadosquadrados](https://twitter.com/somadosquadrados)

# Marília Melo Favalesso

- [mariliabioufpr@gmail.com](mailto:mariliabioufpr@gmail.com)
- [www.mmfava.com](http://www.mmfava.com)
- [/mmfava](https://github.com/mmfava)