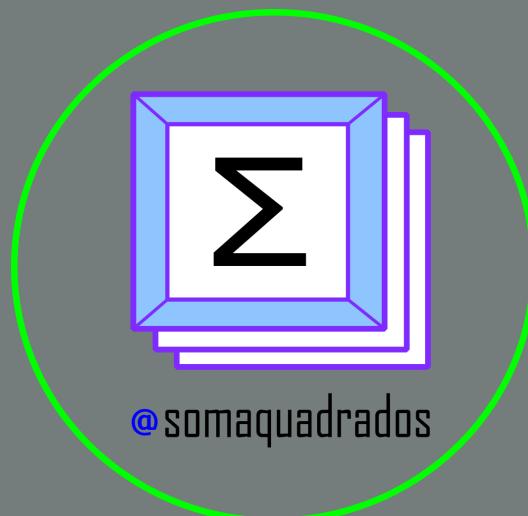


Programación con R

Clase 1



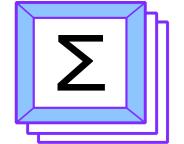
Marília Melo Favalesso

Script

- `clase_1.R`

Contenido de hoy

- Objetos y atribuciones
- Clases
- Tipos de objetos
- Estructuras de Control
- Funciones

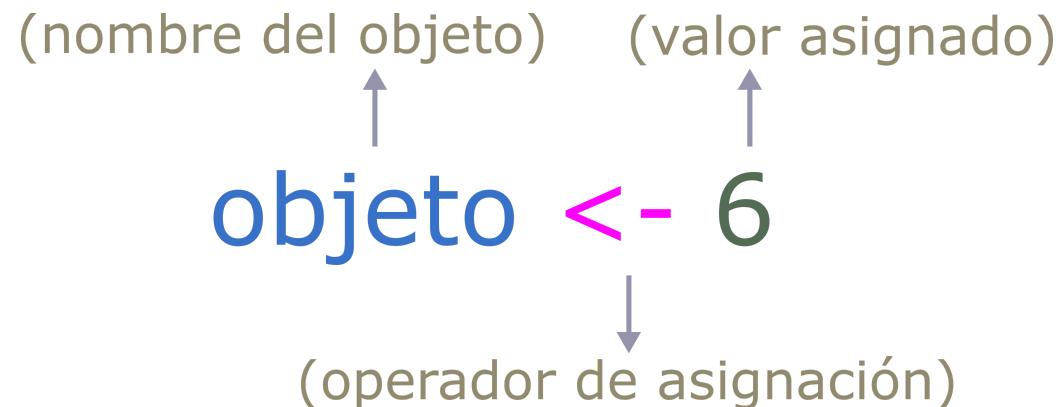


@somaquadrados

Objetos y atribuciones

Objetos y atribuciones

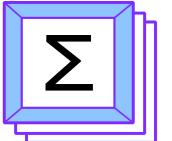
- Los **objetos** son variables capaces de almacenar cualquier valor o estructura de datos.



```
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.



@somaquadrados

Objetos y atribuciones

```
j <- 14
```

...está haciendo una **declaración**, es decir, declarando que la variable `j` ahora se convertirá en un objeto que contiene el número `14`. Las declaraciones se pueden hacer una en cada línea ...

```
j <- 14  
y <- 24
```

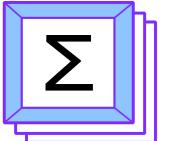
... o separados por ;

```
j <- 14; y <- 24
```

podemos usar el ; siguiendo la asignación para llamar a nuestro objeto y ver su contenido:

```
j <- 14; j
```

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



@somaquadrados

Objetos y atribuciones

Otros ejemplos de creación de objetos

```
b <- 24  
b
```

```
## [1] 24
```

```
c <- 69  
c
```

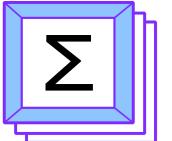
```
## [1] 69
```

```
d <- "e"  
d
```

```
## [1] "e"
```

```
e <- "d"  
e
```

```
## [1] "d"
```



@somaquadrados

Objetos y atribuciones

!!Tenga en cuenta que cada objeto solo puede almacenar una estructura a la vez (un número o una secuencia de valores)!!

```
a <- 5  
a
```

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

```
a <- "bien"  
a
```

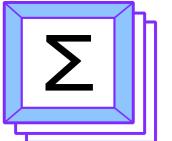
```
## [1] "bien"
```

```
a <- 36924612  
a
```

```
## [1] 36924612
```

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



@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
```

Objetos y atribuciones

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

```
k <- 10  
w <- k  
w
```

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

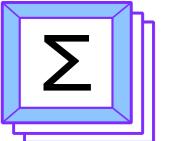
Podemos usar objetos para realizar operaciones matemáticas...

```
a + y / j
```

```
## [1] 36924614
```

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

```
k <- a + y / j
```



@somaquadrados

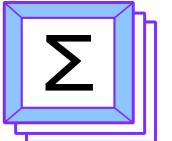
Gestionar el lugar de trabajo

Enumere los objetos creados con la función `ls()`:

```
ls()  
## [1] "a"      "b"      "c"      "d"      "e"      "i"      "j"      "k"      "objeto"  
## [10] "w"     "y"
```

Para eliminar solo un objeto con `rm()`:

```
rm(a) # elimina el objeto 'obj'  
ls() # ¿Qué objetos quedan?  
  
## [1] "b"      "c"      "d"      "e"      "i"      "j"      "k"      "objeto" "w"  
## [10] "y"
```



@somaquadrados

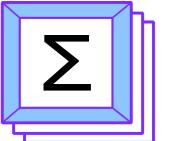
Gestionar el lugar de trabajo

Para eliminar > 1 objeto con `rm()`:

```
rm(c, j, k) # elimina los objetos 'c', 'j' y 'k'  
ls() # ¿Qué objetos quedan?  
  
## [1] "b"        "d"        "e"        "i"        "objeto"   "w"        "y"
```

Para eliminar todos los objetos:

```
rm(list = ls()) # elimina TODOS los objetos  
ls() # ¿Qué objetos quedan?  
  
## character(0)
```



@somaquadrados

Gestionar el lugar de trabajo

Observación:

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

The screenshot shows the RStudio environment. On the left, there is a script editor window containing the following R code:

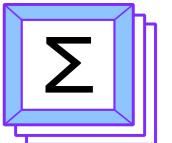
```
1
2
3 a <- 1
4
5 b <- 5
6
7 c <- a + b
8
```

Below the script editor is a console window showing the same code being run in an R session. The output in the console is:

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

On the right side of the interface, there is a large panel titled "Environment". This panel lists the objects created in the session, with a red box highlighting the "Values" table. The table contains the following data:

	Values
a	1
b	5
c	6



@somaquadrados

Objetos y atribuciones

Fuiste al campo y recogiste algunos mosquitos, que se enumeran en la siguiente tabla.

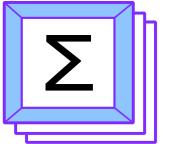
Objeto	n
Anopheles_sp	5
Lutzomyia_sp	35
Aedes_sp	4
Desconocido	16

1 - Cree objetos para cada especie de mosquito y almacene el número de individuos en cada objeto ([n](#) en la tabla).

2 - Cree objetos para cada especie de mosquito y almacene el número de individuos en cada objeto ([n](#) en la tabla).

3 - ¿Qué porcentaje de *Lutzomyia sp.* se muestreó?

4 - Si eliminamos los mosquitos desconocidos, ¿cuál es la cantidad total muestreada? ¿Y cuál es el porcentaje de *Aedes*?



@somaquadrados

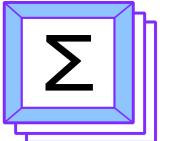
Clases

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 él 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 (enteros o decimales).

```
num <- 1.50
```

2 - **integer**: Números enteros.

```
num_int <- 1L
```

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

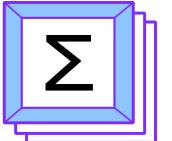
```
logtf <- TRUE
```

4 - **character**: una cadena de caracteres, comúnmente utilizada para representar palabras, frases o texto.

```
ca <- "holla!"
```

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

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



Clases de objetos

Importante!

1 - Use la función* `class()` en R para verificar si la clase de su objeto es correcta:

```
aa <- 1  
class(aa)
```

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

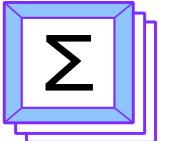
* La idea básica de una función es encapsular un código que se pueda invocar en cualquier momento en R. Su significado y uso son muy similares al de las funciones matemáticas, es decir, hay un nombre, una definición y posterior invocación de la función.

```
f <- function(x) {  
  y <- 6  
  x + y  
}  
  
f(5)
```

```
## [1] 11
```

```
5 + 6  
## [1] 11
```

** Hablaremos más sobre las funciones más adelante.



@somaquadrados

Clases de objetos

Importante!

2 - Las expresiones de tipo **character** deben aparecer entre comillas simples o dobles:

```
bb <- 'Esto es un character!'
bb
```

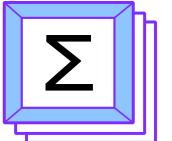
```
## [1] "Esto es un character!"
```

```
cc <- "Lo mismo (:"
cc
```

```
## [1] "Lo mismo (:"
```

```
dd <- "I'm gonna take my horse to the old town road 🎶"
dd
```

```
## [1] "I'm gonna take my horse to the old town road <U+266B>"
```



@somaquadrados

Clases de objetos

Importante!

3 - Los números en **R** generalmente se tratan como objetos **numeric** (números reales de doble precisión), incluso los que escribimos como enteros.

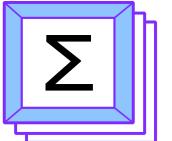
Para que un entero se trate como un objeto **integer**, debe usar la letra **L** después del número:

```
dd <- 1
class(dd) # NO interpretado como integer

## [1] "numeric"

ee <- 1L
class(ee) # interpretado como integer

## [1] "integer"
```



@somaquadrados

Clases de objetos

Importante!

4 - Los valores **logical** (o booleanos) son **TRUE** (verdaderos) o **FALSE** (falsos). También se aceptan **T** o **F**.

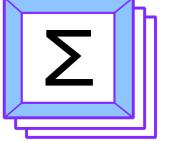
```
ff <- T  
gg <- TRUE  
  
ff == gg
```

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

```
hh <- F  
ii <- FALSE  
  
hh == ii
```

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

| == : operador matemático "exactamente igual a ..."

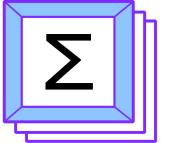


@somaquadrados

Clases de objetos

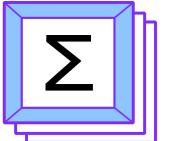
Ejercicios

Crea un objeto para las clases `numeric`, `integer`, `character` e `logical` y comprueba que  lo hizo correctamente.



@somaquadrados

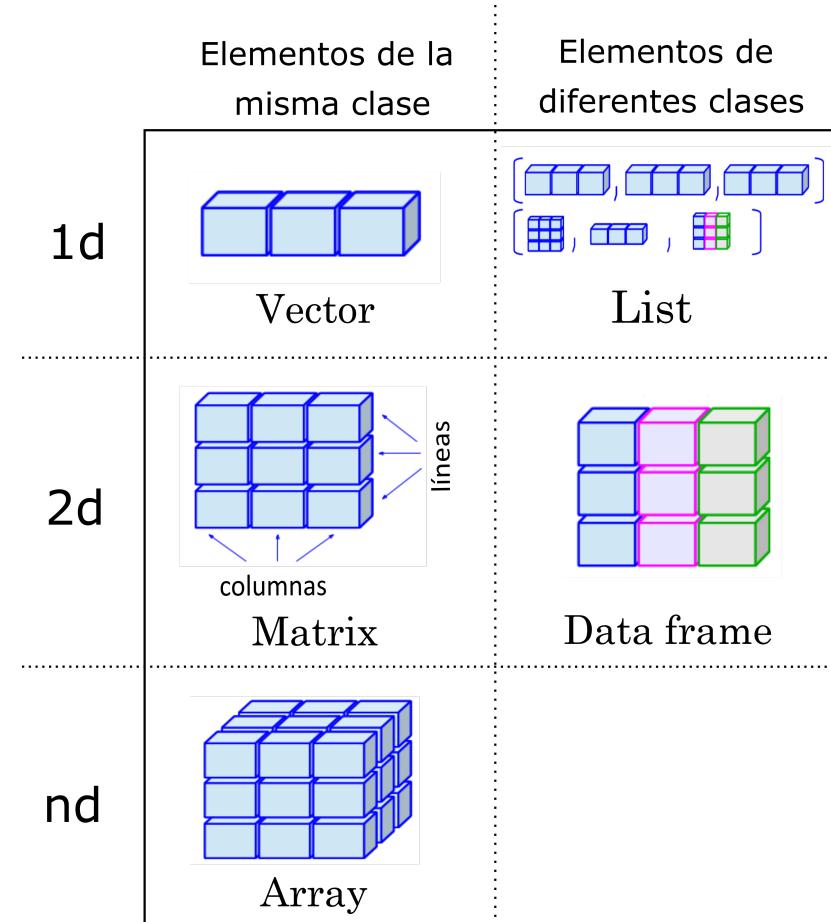
Tipos de objetos

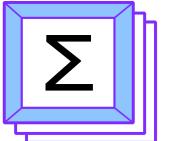


@somaquadrados

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**

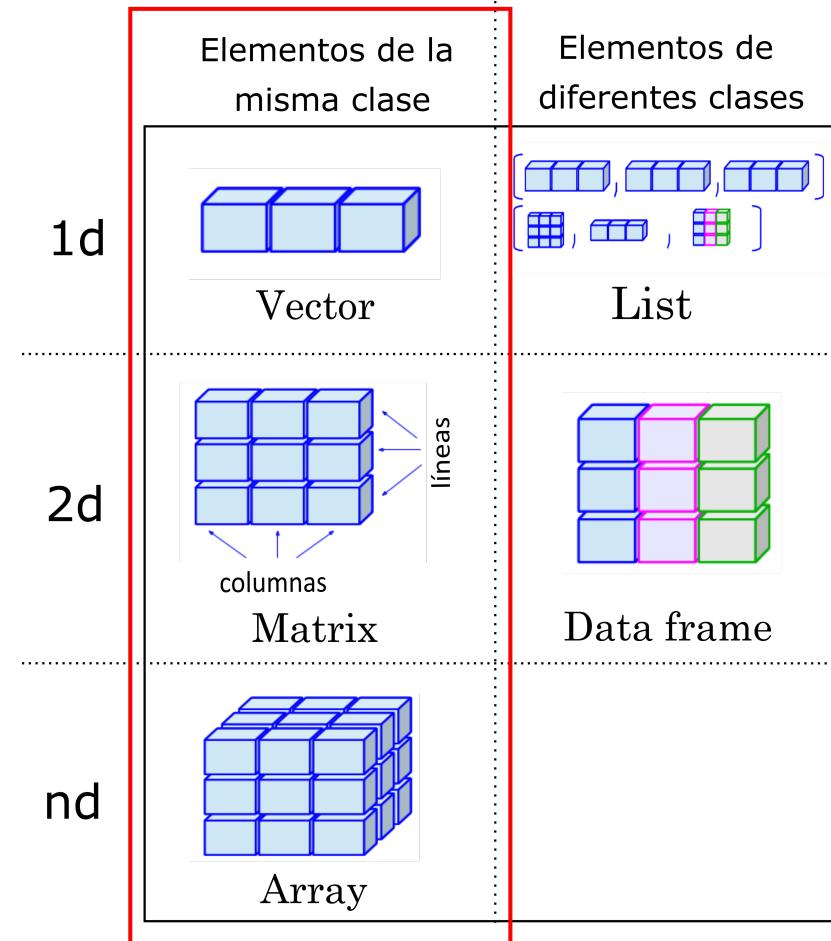


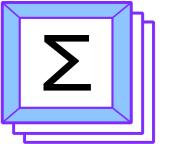


@somaquadrados

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**



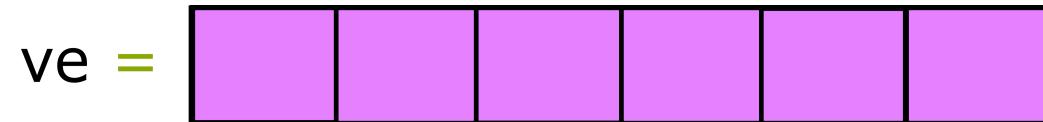


@somaquadrados

Tipos de objetos

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**.

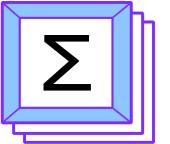
```
aa <- c(1, 2, 3, 4, "a")
class(aa)
```

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

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

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

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



@somaquadrados

Tipos de objetos

Vetores

Conversión

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

```
a <- 1
```

```
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"
```

Tipos de objetos

Vetores

Hay algunas formas prácticas de crear vectores...

- Secuencia de unidad: `x1:xn`.

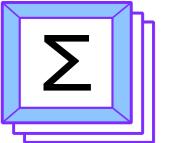
```
anos <- 2001:2021  
anos
```

```
## [1] 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017  
## [18] 2018 2019 2020 2021
```

- Secuencia con espaciado diferente: `seq()`

```
edad <- seq(from = 0, to = 80, by = 20)  
edad
```

```
## [1] 0 20 40 60 80
```



@somaquadrados

Tipos de objetos

Vetores

- Repetición: `rep()`.

```
area <- rep(x = c("urb", "rur"), times = 3)
area
```

```
## [1] "urb" "rur" "urb" "rur" "urb" "rur"
```

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

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

Tipos de objetos

Vetores

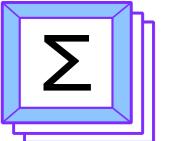
- Nombres con secuencia numérica: `paste()`.

```
# Sin una separación definida
muestras <- paste("muestra", 1:10)
muestras
```

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

```
# Con una separación definida
muestras <- paste("muestra", 1:10, sep = "_")
muestras
```

```
## [1] "muestra_1"  "muestra_2"  "muestra_3"  "muestra_4"  "muestra_5"  "muestra_6"
## [7] "muestra_7"  "muestra_8"  "muestra_9"  "muestra_10"
```



@somaquadrados

Tipos de objetos

Vetores

- Muestreo aleatorio de valores: `sample()`.

```
# sorteo sin reemplazo
```

```
sorteo1 <- sample(1:100, 20, replace = F)  
sorteo1
```

```
## [1] 45 81 66 7 30 8 41 67 11 79 54 36 48 70 50 96 2 10 51 17
```

```
# sorteo con reemplazo
```

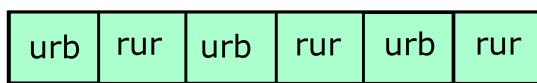
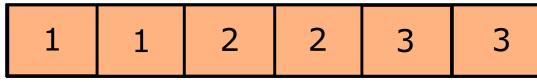
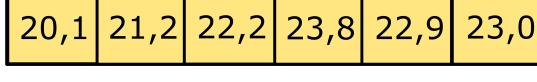
```
sorteo2 <- sample(1:100, 20, replace = T)  
sorteo2
```

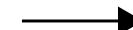
```
## [1] 96 30 13 15 77 47 56 27 99 60 6 85 100 63 81 25 26 98 87 92
```

Tipos de objetos

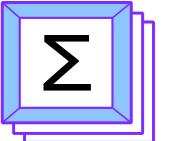
Vetores

- ¡Muy importante para el análisis de datos!
- Un `data frame` se compone de diferentes vectores.

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



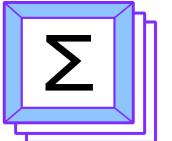
@somaquadrados

Tipos de objetos

Vetores

Pertenece (`%in%`) es un operador muy útil cuando necesitamos verificar si un cierto valor está dentro de nuestro conjunto de valores (vector):

```
# ¿recuerdas el objeto "área" que creamos?  
area  
  
## [1] "urb" "rur" "urb" "rur" "urb" "rur"  
  
# ¿Hay 'valores' llamados 'urb' en él?  
"urb" %in% area  
  
## [1] TRUE  
  
# ¿Hay 'valores' llamados 'for'?  
"for" %in% area  
  
## [1] FALSE
```



@somaquadrados

Tipos de objetos

Vetores

Ejercicios

1. Comenzará un estudio con flebótomos en la ciudad de Puerto Iguazú y deberá seleccionar al azar 3 de 10 vecindarios para el muestreo. Haga un sorteo con  y almacena los resultados en un objeto.
2. Crear un nuevo objeto mediante la repetición de los barrios donde recogerá los flebótomos 4 veces (una por cada estación de barrio). Por ejemplo, si va a muestrear los vecindarios "1", "3" y "6", el resultado debería ser  (1, 3, 6, 1, 3, 6, 1, 3, 6, 1, 3, 6).
3. ¿El barrio "1" forma parte de su muestra? ¿Y el barrio "10"? ¿Y el "7"?

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*.

```
temporada <- factor(x = c("verano", "verano", "primavera", "primavera", "primavera", "otono", "invierno", "invierno"),  
levels = c("verano", "primavera", "otono", "invierno"))
```

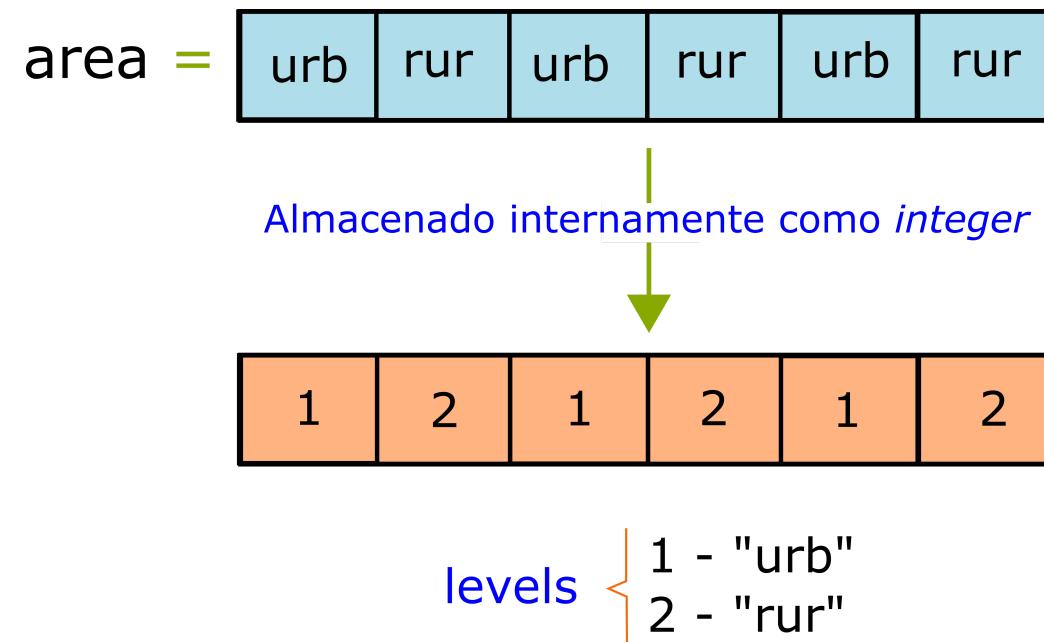
```
temporada
```

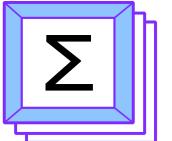
```
## [1] verano     verano     primavera primavera primavera otono      invierno  invierno  
## Levels: verano primavera otono invierno
```

Tipos de objetos

Factor

Internamente, R almacena los factores como *integer*.





@somaquadrados

Tipos de objetos

Factor

Tipos

- *Factor nominal:* variables nominales.

```
genero <- factor(x = c("Lonomia", "Megalopyge", "Automeris", "Hylesia", "Megalopyge", "Automeris", "Hylesia"),
                  levels = c("Lonomia", "Megalopyge", "Hylesia", "Automeris"))
genero

## [1] Lonomia     Megalopyge Automeris   Hylesia     Megalopyge Automeris   Hylesia     Lonomia
## [9] Hylesia     Megalopyge
## Levels: Lonomia Megalopyge Hylesia Automeris

levels(genero)

## [1] "Lonomia"    "Megalopyge"  "Hylesia"    "Automeris"
```

Tipos de objetos

Factor

Tipos

- *Factor ordinal*: variable ordinal.

```
mes <- factor(x = c("Janeiro", "Janeiro", "Fevereiro", "Fevereiro", "Março", "Março"),
               levels = c("Janeiro", "Fevereiro", "Março"), ordered = TRUE)
mes
```

```
## [1] Janeiro Janeiro Fevereiro Fevereiro Março      Março
## Levels: Janeiro < Fevereiro < Março
```

```
levels(mes)
```

```
## [1] "Janeiro"   "Fevereiro"  "Março"
```

Tipos de objetos

Factor

Convertir un vector para un factor: `as.factor()`.

```
# Vector de caracteres.  
letras <- c("a", "c", "b", "d", "c", "a", "b", "d", "c")  
letras  
  
## [1] "a" "c" "b" "d" "c" "a" "b" "d" "c"  
  
# Convierta el objeto en factor.  
letras2 <- as.factor(letras)  
letras2  
  
## [1] a c b d c a b d c  
## Levels: a b c d
```

Tipos de objetos

Factor

Ejercicios

Muestrará en tres áreas diferentes de Puerto Iguazú (Misiones, Argentina):

1. El Parque Nacional Iguazú (PNI)
2. un área antropogénica
3. un área rural

De esta manera, cree un vector que repita 12 veces el nombre de cada ubicación de recolección (= 1 año de muestreo).

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

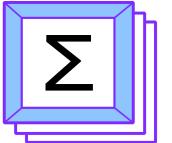
Matrix

- 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



Tipos de objetos

Matrix

Puede construir matrices en **R** de dos formas:

1 - Disposición de elementos de un vector: **matrix()**.

```
ma <- 1:12  
ma
```

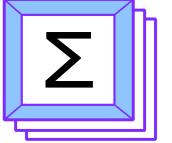
```
## [1] 1 2 3 4 5 6 7 8 9 10 11 12
```

```
m <- matrix(data = ma, nrow = 4, ncol = 3, byrc  
m
```

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

```
m <- matrix(data = ma, nrow = 4, ncol = 3, byrc  
m
```

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



@somaquadrados

Tipos de objetos

Matrix

Puede construir matrices en **R** de dos formas:

2 - 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

Matrix

Para cambiar el nombre de las filas y columnas de una `matrix`, utilice las funciones `rownames()` y `colnames()` respectivamente.

- Antes

```
ma <- 1:12
m <- matrix(data = ma, nrow = 4, ncol = 3, byrc
m

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

- Despues

```
colnames(m) <- c("A", "B", "C") # cambia colum
rownames(m) <- c("LA", "LB", "LC", "LD") # cam
m

##      A  B  C
## LA  1  2  3
## LB  4  5  6
## LC  7  8  9
## LD 10 11 12
```

Tipos de objetos

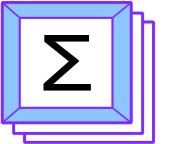
Matrix

Ejercicios

Está desarrollando un medicamento para la fiebre y desea analizar el efecto a lo largo del tiempo. Tiene 3 pacientes y midió su fiebre después de la medicación en los tiempos: 5 m, 10 m, 15 m, 20 m y 25 m. Los resultados son:

- Paciente 1: 38, 37.9, 37.3, 37.2, 36.9
- Paciente 2: 37.9, 37.6, 37.1, 36.8, 36
- Paciente 3: 38.2, 38, 37.8, 37.2, 36.8

Configure una matriz de datos con pacientes en filas y tiempos en columnas.

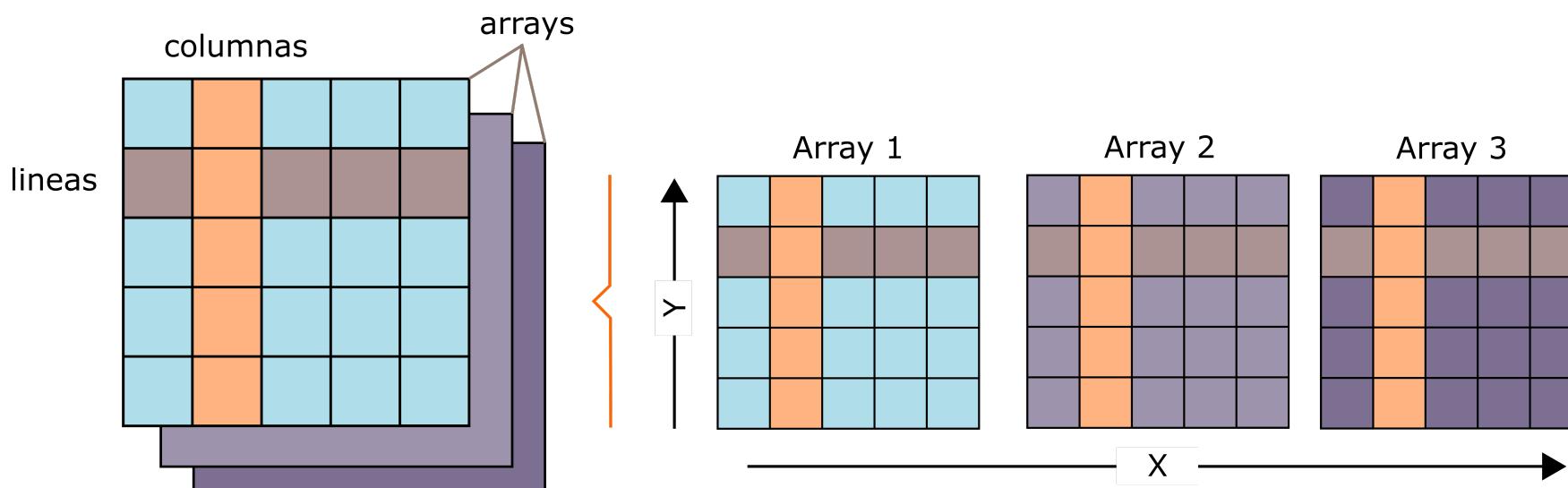


@somaquadrados

Tipos de objetos

Array

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



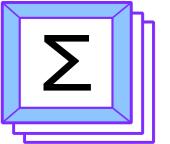
Tipos de objetos

Array

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

```
vc <- 1:8 # datos
ar <- array(data = vc, dim = c(2, 2, 2)) # array
ar

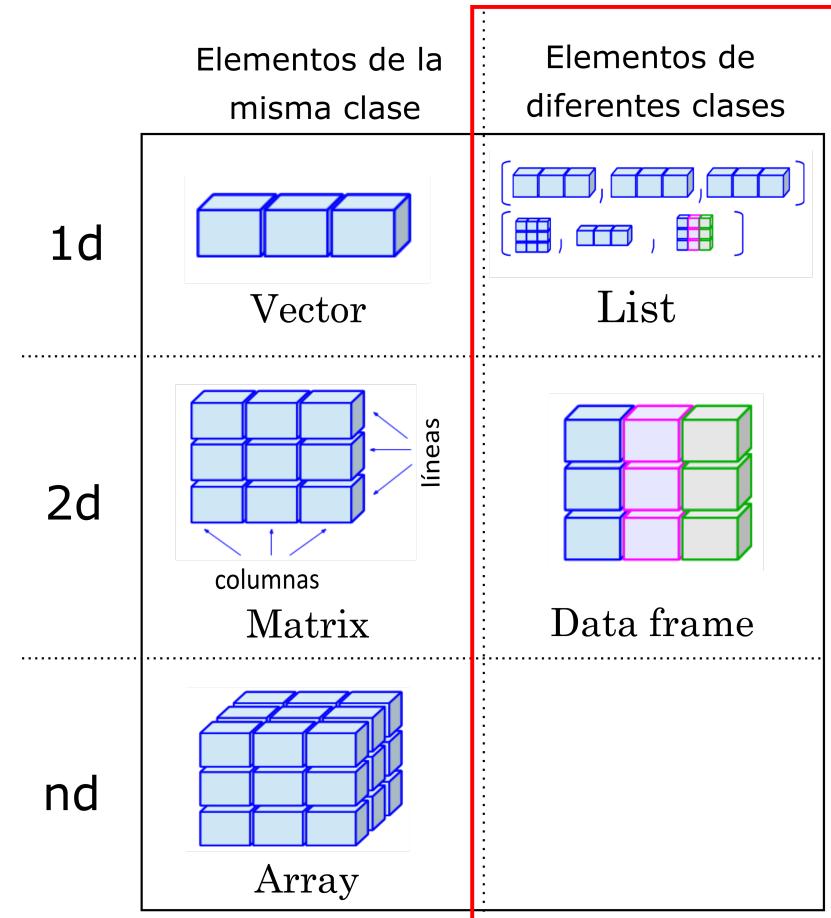
## , , 1
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## , , 2
##
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
```

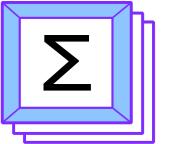


@somaquadrados

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.

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

lineas

columns

