



R Base

Big Data e Inteligencia Territorial

Hoja de ruta

- ✓ Valores
- ✓ Vectores (Variables)
- ✓ Funciones
- ✓ Objetos
- ✓ Data frames (bases de datos)

VALORES

Valores


En R la unidad mínima para trabajar son los *valores*:


- `1` es un valor (numérico),
- `"uno"` es un valor (cáriter),
- `"1"` es un valor (cáriter) y
- `"Esto es un uno"` también es un valor (cáriter).

Valores

R permite realizar un sinfín de operaciones con estos valores, por ejemplo, operaciones matemáticas como *sumar*, *restar*, *multiplicar* o *dividir*:

Code

 Start Over

 Run Code

```
1  
2  
3
```

VECTORES

Vectores

- A un conjunto de valores lo llamaremos **vector** y R los interpretará bajo el comando `c()`.
- Los vectores, como los valores, también serán de un tipo determinado:

- Vector numérico (*numeric*)

```
c(1, 2, 3, 4, 5)
```

```
c(1:5)
```

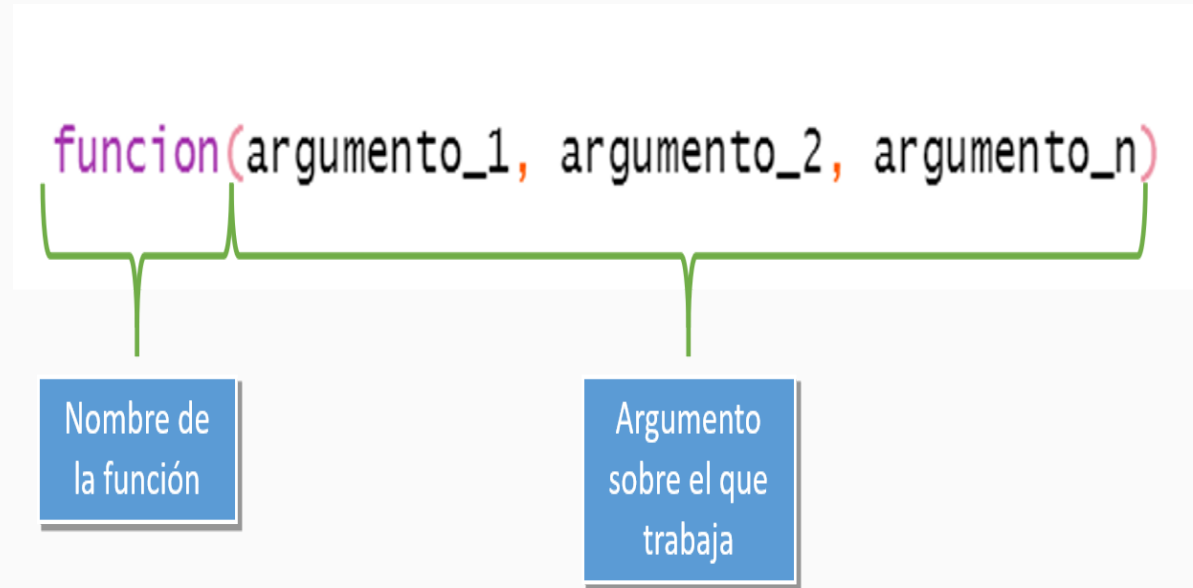
- Vector de texto (*character*)

```
c("uno", "dos", "tres", "cuatro", "cinco")
```

FUNCIONES

Funciones

- **La función** es una operación que nos ayuda a resolver un problema que de otra forma llevaría más pasos/tiempo.
- **Una función** tiene *nombre* y suele (intenta) ser intuitivo respecto a la acción que realiza (el problema que resuelve). También comienza y termina con paréntesis `()`, donde se ubica el o los argumentos.



Funciones

- Por ejemplo, la función `class()` evalúa el tipo (numérico, carácter) de un elemento

Evaluemos de qué tipo es el valor `2`

```
class(2)
```

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

¿y el valor `"dos"`?

```
class("dos")
```

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

Funciones

- Otras funciones nos permiten incluir más de un argumento. En este caso, cada uno de ellos se separa con una coma (,).

Por ejemplo, la función `sum()` permite *sumar* varios valores numéricos:

```
sum(2, 5, 10)
```

```
[1] 17
```

La función `paste0()` permite *pegar* varios valores de tipo texto:

```
paste0("Esto", "es", "un texto", "con", "seis", "valores")
```

```
[1] "Esto es un texto con seis valores"
```

Funciones

- Ciertas funciones también incluyen *parámetros*, los cuales agregan *especificaciones* que hacemos a la operación que realiza la función.

Por ejemplo, queremos una función que pegue los valores anteriores pero que los separe por un espacio. En este caso, la función `paste()` (sin el `0` al final) contiene un parámetro llamado `sep =`, que permite definir un separador entre cada argumento.

```
paste("Esto", "es", "un texto", "con", "seis", "valores", sep = "_")
```

```
[1] "Esto_es_un texto_con_seis_valores"
```

PRÁCTICA

Práctica

1. Sumar dos o más valores sin utilizar una función
2. Crear un vector numérico que contenga 7 valores
3. Crear un vector de texto que contenga 3 valores
4. Verificar de qué tipo es el valor "67" (comprobar con comillas y sin comillas)
5. Pegar dos o más valores de tipo *character* (texto), cada uno separado por un espacio en blanco

05:00

OBJETOS

Objetos

- En R el elemento más importante es el **objeto**. Tanto valores como vectores (y prácticamente, cualquier elemento) pueden ser *asignados* a un objeto.
- Al objeto debemos definirlo por un nombre (elige tu propia aventura) y *asignarle* el contenido:

Operador para asignar

```
edad <- c(24, 56, 75, 42, 99)
```



Nombre del
objeto

Contenido del objeto

Objetos

- Al igual que *valores* y *vectores*, hay diferentes *tipos* de objetos:

- Objeto numérico (*numeric*)

```
edad ← c(24, 56, 75, 42, 99)
```

- Objeto de texto (*character*)

```
nombre ← c("D'rtanian", "Rigoberta", "Menganita", "Juancito", "Estanislao")
```

- (otro) objeto de texto (*character*)

```
nombre_y_apellido ← c("D'artanian estrujillo", "Rigoberta manchuria", "Menganita fulaique",  
                      "Juancito loquillo", "Estanislao leningrado")
```

Objetos

Objetos

- Para ver el contenido de un objeto, simplemente debemos escribir su nombre y ejecutar en el script o la consola

```
edad
```

```
[1] 24 56 75 42 99
```

```
nombre_y_apellido
```

```
[1] "D'artanian estrujillo" "Rigoberta manchuria"  "Menganita fulaique"  
[4] "Juancito loquillo"    "Estanislao leningrado"
```

Nombrando objetos

[Reglas](#)

[Sugerencias](#)

[Convenciones](#)

⊘ No se aceptan espacios

Objeto llamado ~~nombre y apellido~~

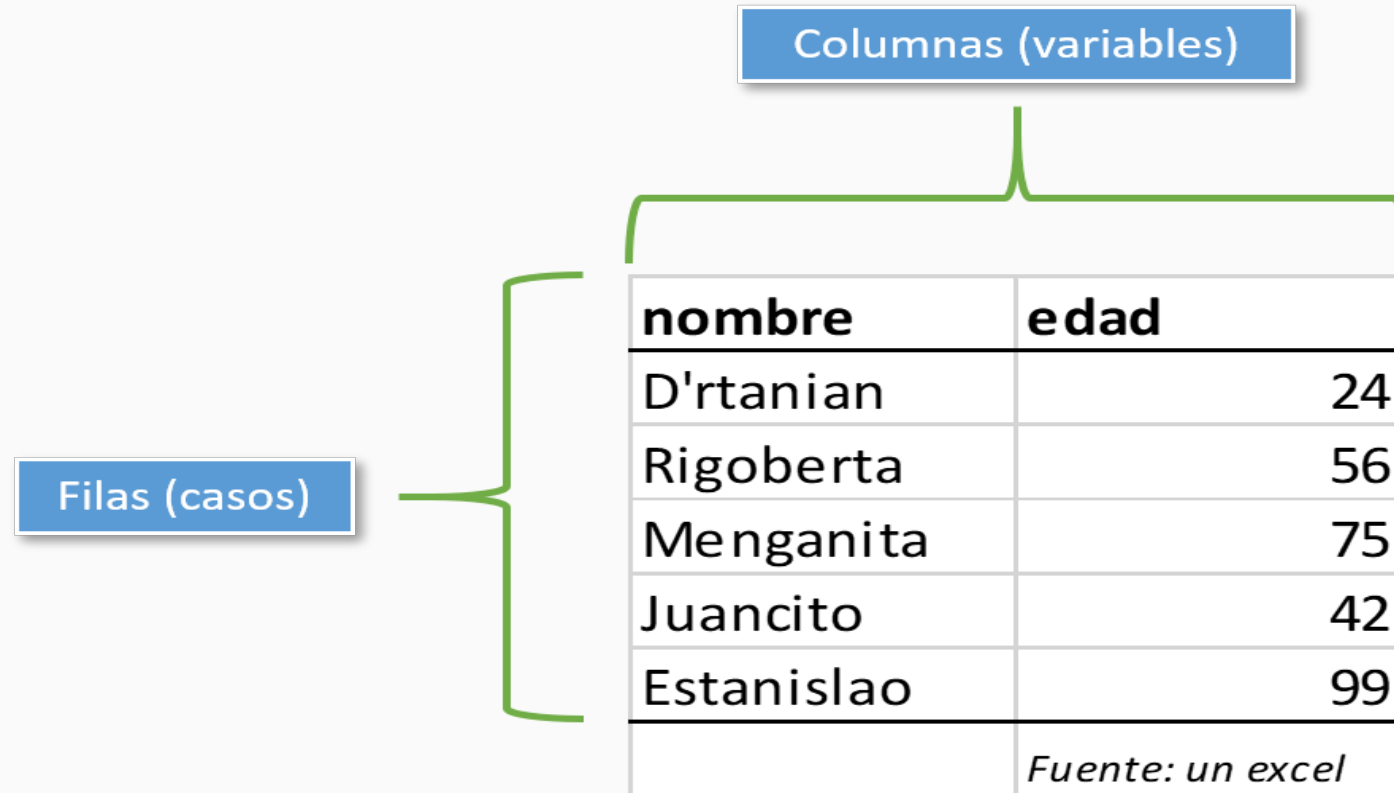
⊘ No se puede empezar con un número

Objeto llamado ~~1_objeto~~

DATA FRAMES (bases de datos)

Data frames

El elemento `data.frame` es lo que conocemos como una *base de datos*: Filas (casos) y columnas (variables) relacionadas entre sí:



The diagram illustrates a data frame structure. A blue box labeled "Columnas (variables)" is positioned above the table, with a green bracket indicating the columns. Another blue box labeled "Filas (casos)" is positioned to the left of the table, with a green bracket indicating the rows. The table itself has two columns: "nombre" and "edad". The rows contain the following data:

nombre	edad
D'rtanian	24
Rigoberta	56
Menganita	75
Juancito	42
Estanislao	99
Fuente: un excel	

Data frames

- La función `data.frame()` nos permite crear una base de datos vinculando vectores:

```
data.frame(nombre_y_apellido, edad)
```

	nombre_y_apellido	edad
1	D'artanian estrujillo	24
2	Rigoberta manchuria	56
3	Menganita fulaique	75
4	Juancito loquillo	42
5	Estanislao leningrado	99

- Podemos guardarlo en un objeto:

```
base_personas ← data.frame(nombre_y_apellido, edad)
```

Data frames

- Algunas funciones para trabajar con los data frames:

```
dim(base_personas)
```

```
[1] 5 2
```

```
summary(base_personas)
```

nombre_y_apellido	edad
Length:5	Min. :24.0
Class :character	1st Qu.:42.0
Mode :character	Median :56.0
	Mean :59.2
	3rd Qu.:75.0
	Max. :99.0

Navengando por objetos: vectores

R base nos permite *navegar* entre los valores de un vector o data frame, y lo hace a través de `[]`

- Supongamos que queremos extraer el 2do valor del objeto edad:

```
edad ← c(24, 56, 75, 42, 99)
```

```
edad[2]
```

```
[1] 56
```

- Podemos guardar en un objeto dicho valor

```
valor_2do ← edad[2]
```

```
valor_2do
```

```
[1] 56
```

Navengando por objetos: vectores

Así como podemos consultarle a R por un valor en particular de un vector, también podemos usar el comando `[]` para *editar* ese valor:

```
edad[2]
```

```
[1] 56
```

```
edad[2] ← 76
```

Chequeo el contenido de mi objeto `edad`

```
edad
```

```
[1] 24 76 75 42 99
```

Navengando por objetos: base de datos

Supongamos que tenemos la siguiente base de datos:

```
edad ← c(24, 56, 75, 42, 99)
nombre_y_apellido ← c("D'artanian estrujillo", "Rigoberta manchuria", "Menganita fulaique", "J
base_personas ← data.frame(nombre_y_apellido, edad)

base_personas
```

	nombre_y_apellido	edad
1	D'artanian estrujillo	24
2	Rigoberta manchuria	56
3	Menganita fulaique	75
4	Juancito loquillo	42
5	Estanislao leningrado	99

Navengando por objetos: base de datos

Queremos extraer la **edad** de *Menganita fulaique*:

```
base_personas[3,2]
```

```
[1] 75
```

```
base_personas[3,2]
```

nombre_y_apellido	edad
D'artanian estrujillo	24
Rigoberta manchuria	56
Menganita fulaique	75
Juancito loquillo	42
Estanislao leningrado	99
Fuente: un excel	

Navengando por objetos: base de datos

También podemos consultarle a R por los valores de una columna entera de nuestra base con el símbolo `$`:

```
base_personas$nombre_y_apellido
```

```
[1] "D'artanian estrujillo" "Rigoberta manchuria"  "Menganita fulaique"  
[4] "Juancito loquillo"    "Estanislao leningrado"
```

```
base_personas$edad
```

```
[1] 24 56 75 42 99
```

Navengando por objetos: base de datos

Y aplicar una función sobre esa columna

```
mean(base_personas$edad)
```

```
[1] 59.2
```

```
summary(base_personas$edad)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
24.0	42.0	56.0	59.2	75.0	99.0

PRÁCTICA

1. Crear un vector llamado *nombre* que contenga 6 valores (6 nombres)
 2. Extraer el segundo valor del vector creado y asignarlo a un nuevo objeto
-

- Dados los siguientes vectores:

```
localidad ← c("Jujuy", "Jujuy", "La Pampa", "Córdoba", "Jujuy", "Chubut")  
tipo_alojamiento ← c("Casa", "Casa", "Depto", "Depto", "Depto", "Casa")
```

1. Crear un objeto de tipo data.frame (base de datos) que contenga el vector creado (*nombre*) más los dos propuestos (*localidad* y *tipo_alojamiento*)
2. Extraer del data.frame el valor de la tercer fila y segunda columna.
3. Consultar del data.frame sobre los valores de la columna *tipo_alojamiento*

05:00