

- RStudio
- Como ingresar los datos en R?
- Tipos de datos en R



Por que RStudio?

- Hace que R sea más fácil de usar:
- Organiza su trabajo de R en un solo conjunto de ventanas fácil de ver
- ¡Muchas otras características! (... Rmarkdown para hacer buenos informes, etc. Hablaremos de ellos más adelante)

Abran R Studio



Studio[®]



RStudio

Console ~/ Go to file/function Addins Project: (None)

R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions
.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications
.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

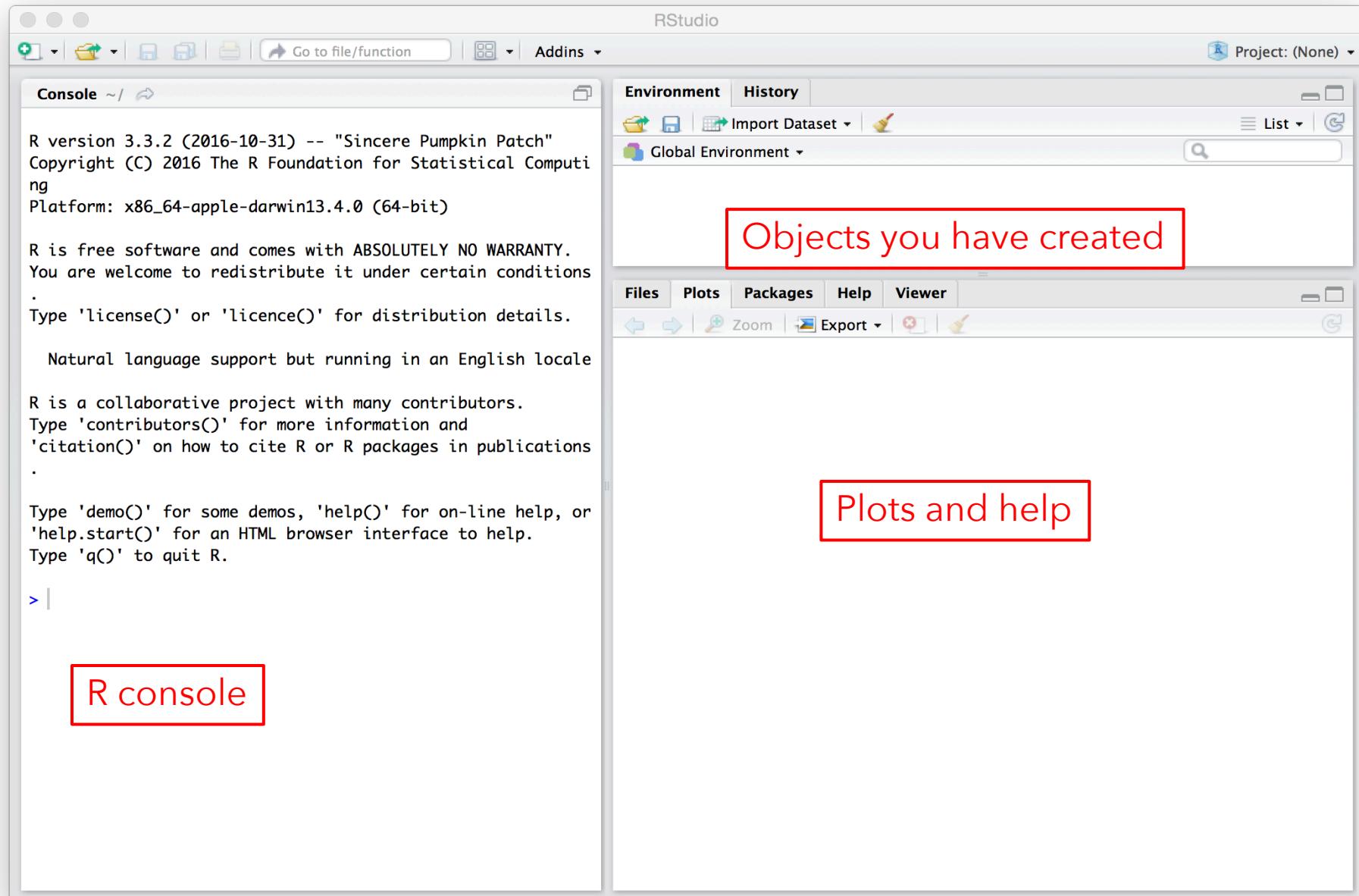
> |

Environment History Import Dataset Global Environment

Environment is empty

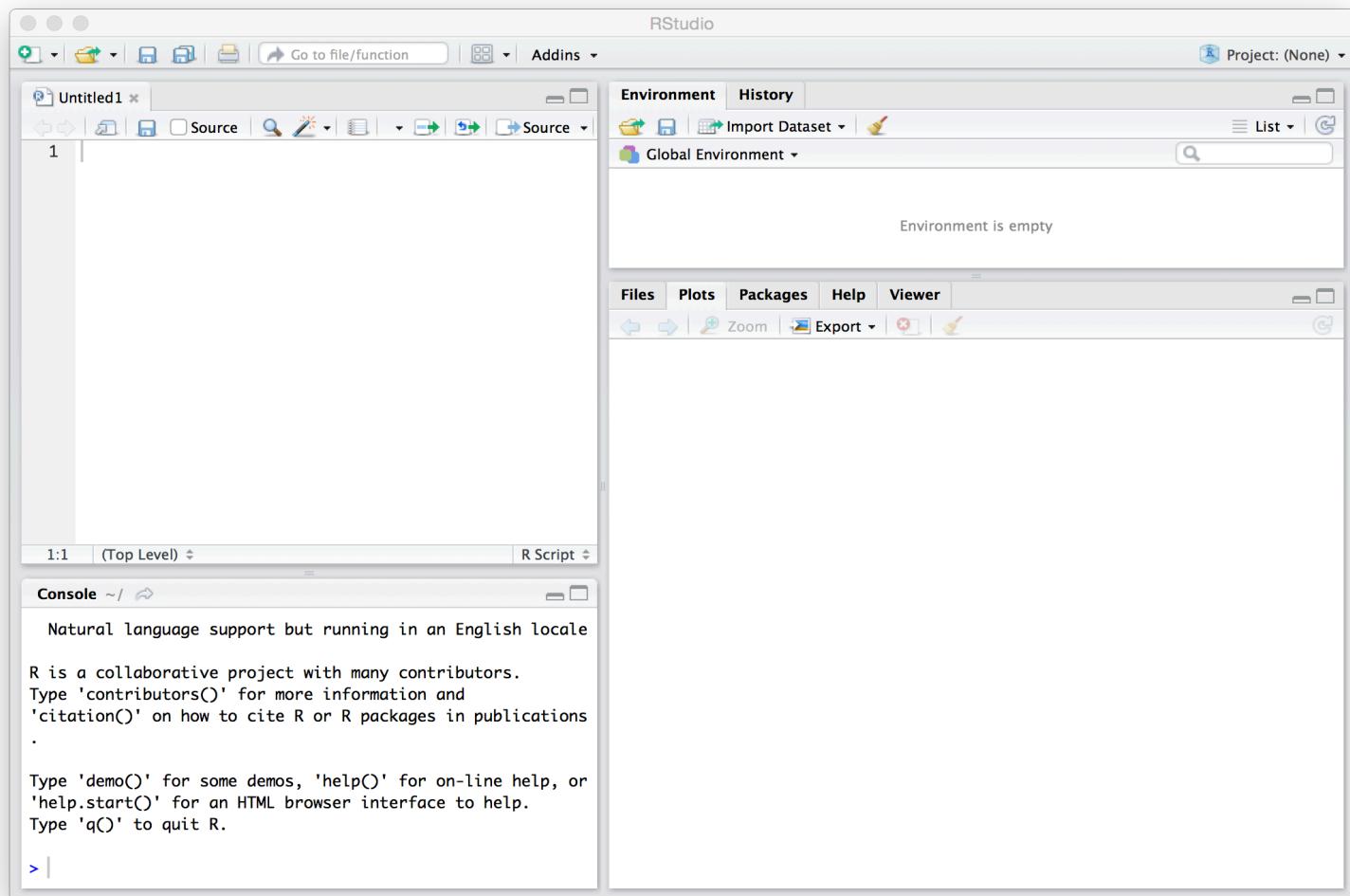
Files Plots Packages Help Viewer

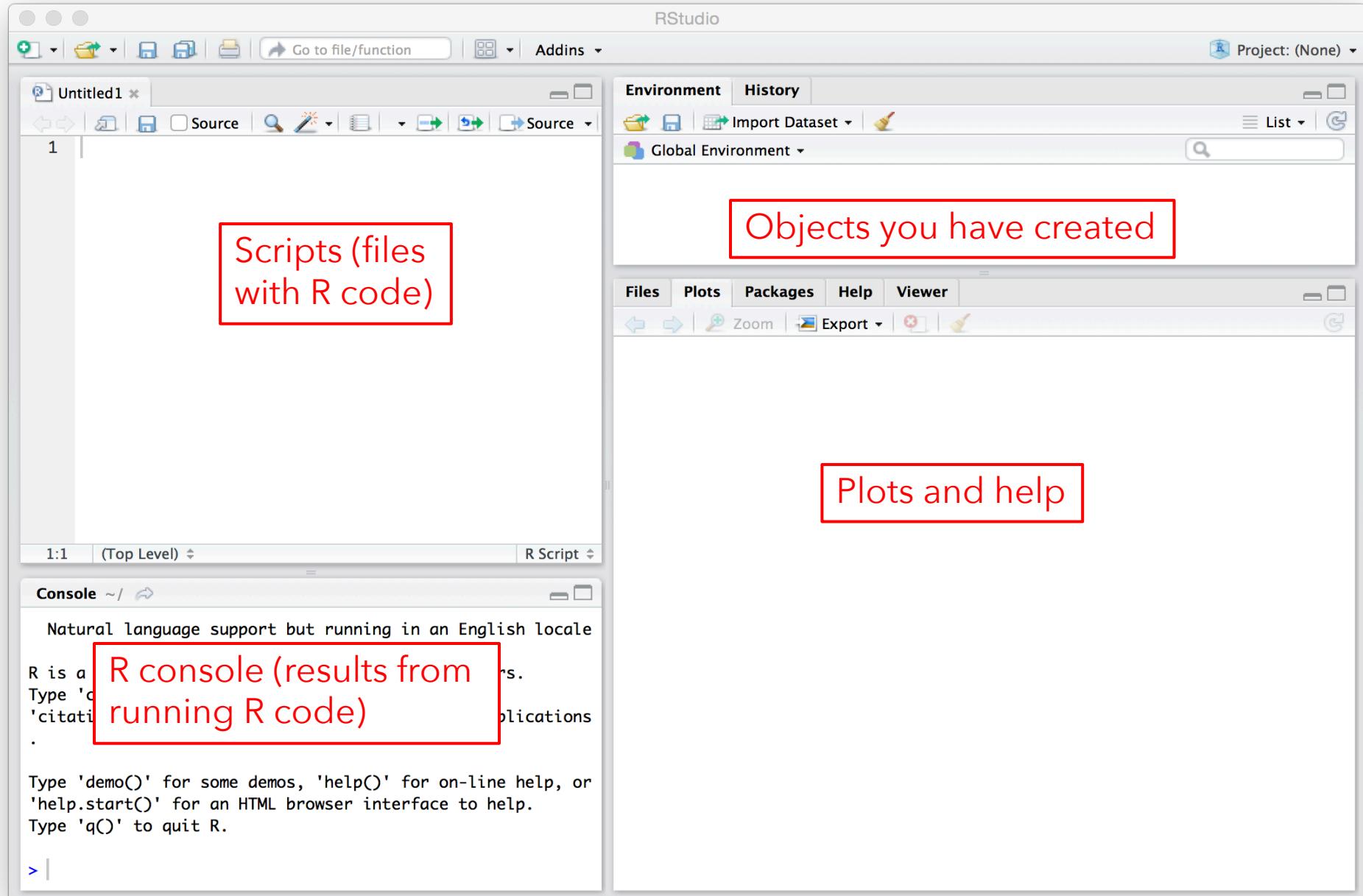
Zoom Export



Abrir un nuevo script...

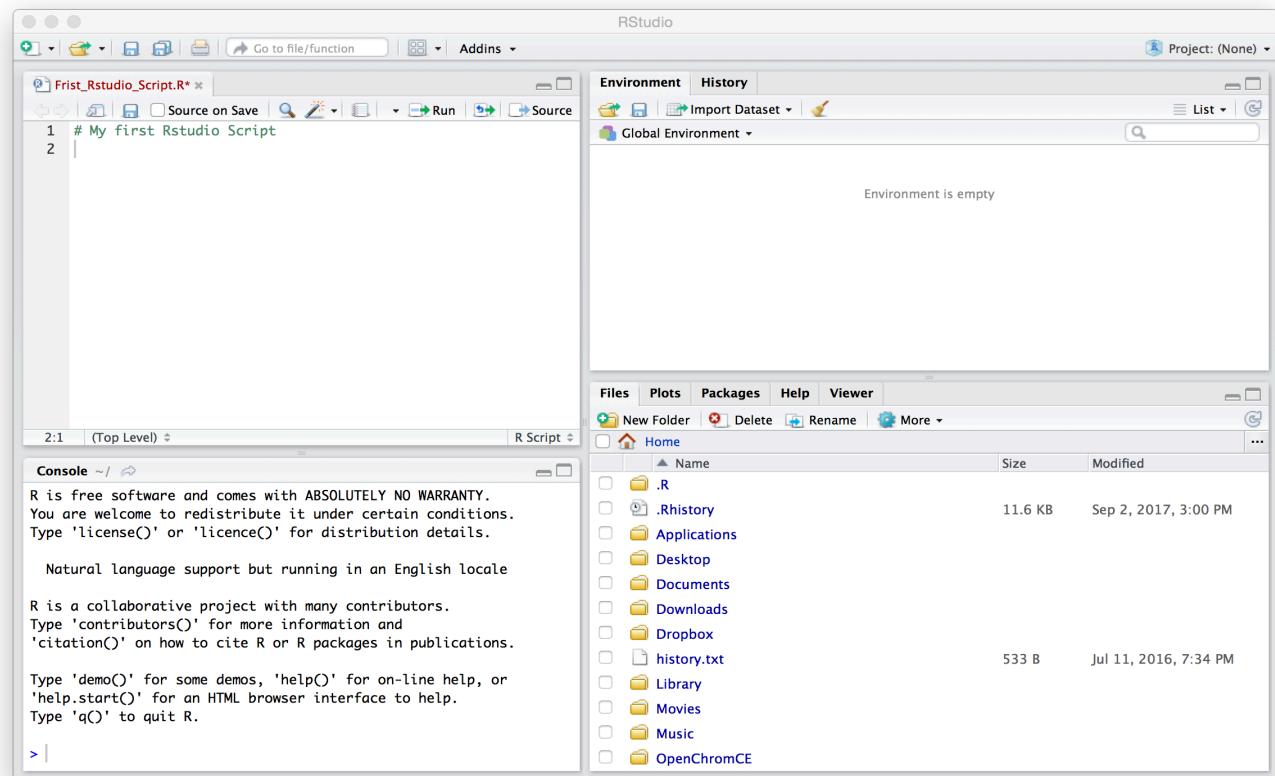
- File > New File > R Script



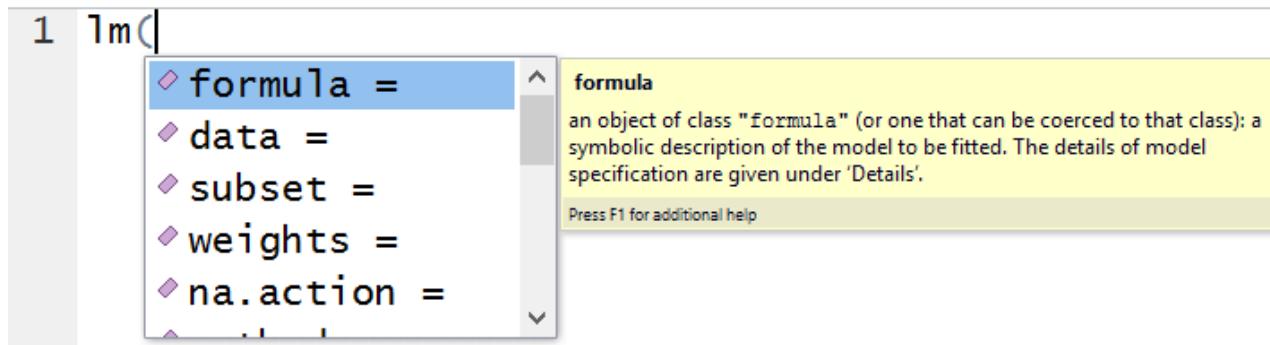
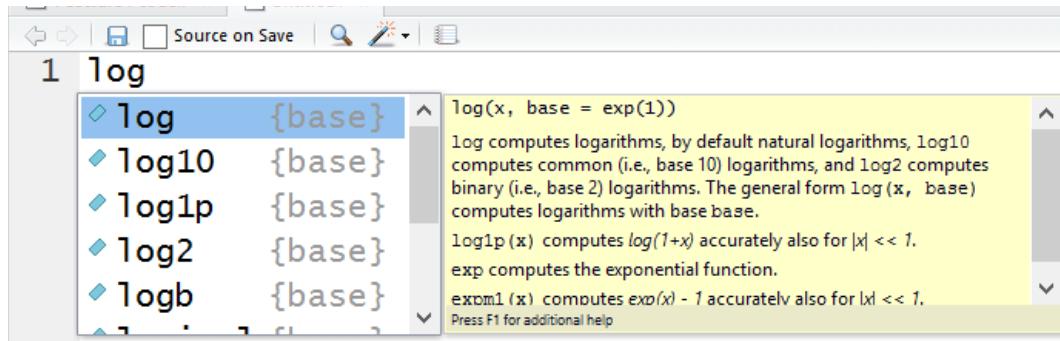


Guardar script

- File > Save
 - nombre.R”



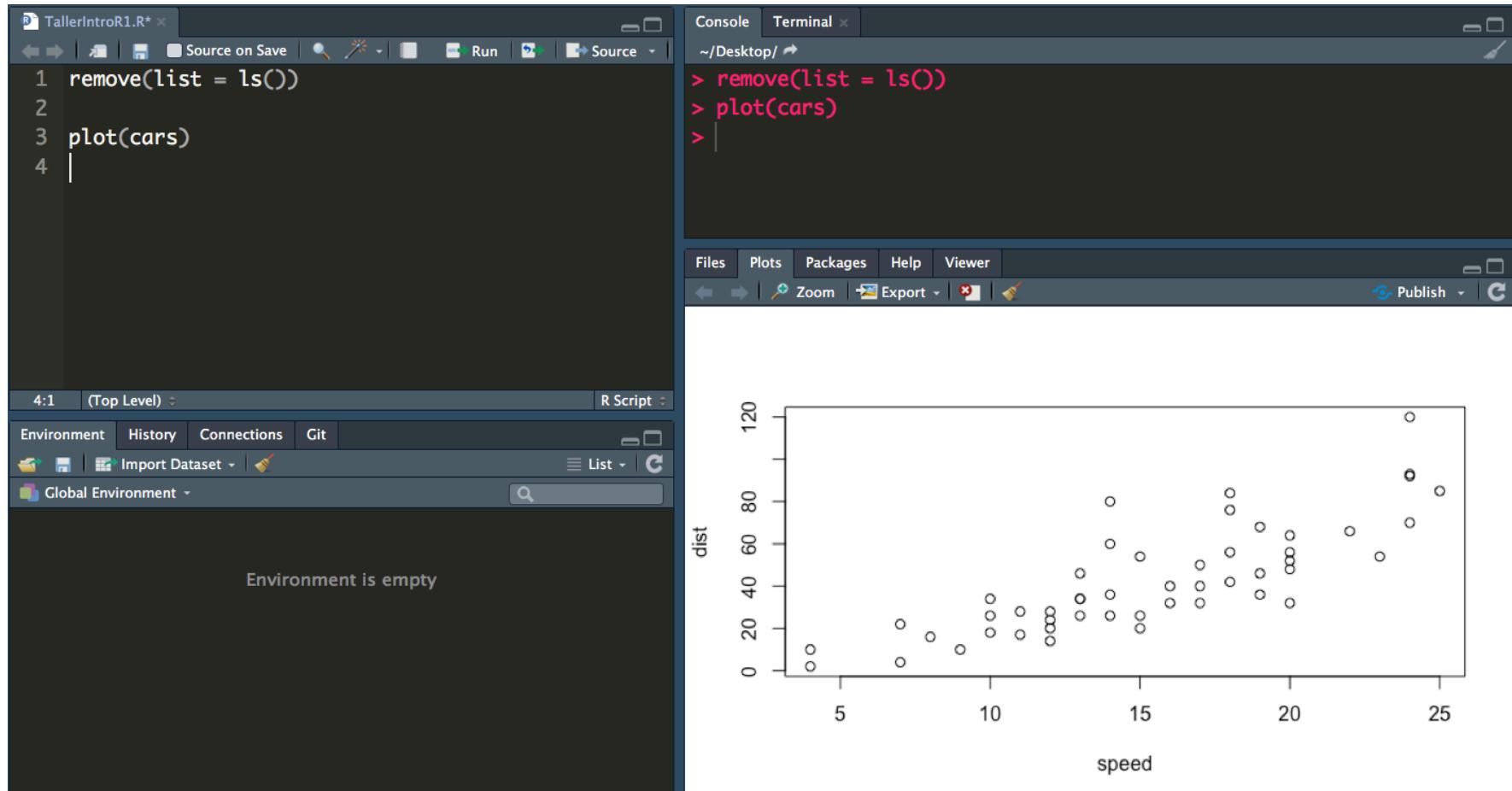
Ayuda rápida en RStudio



?lm

Graficar

plot(cars)



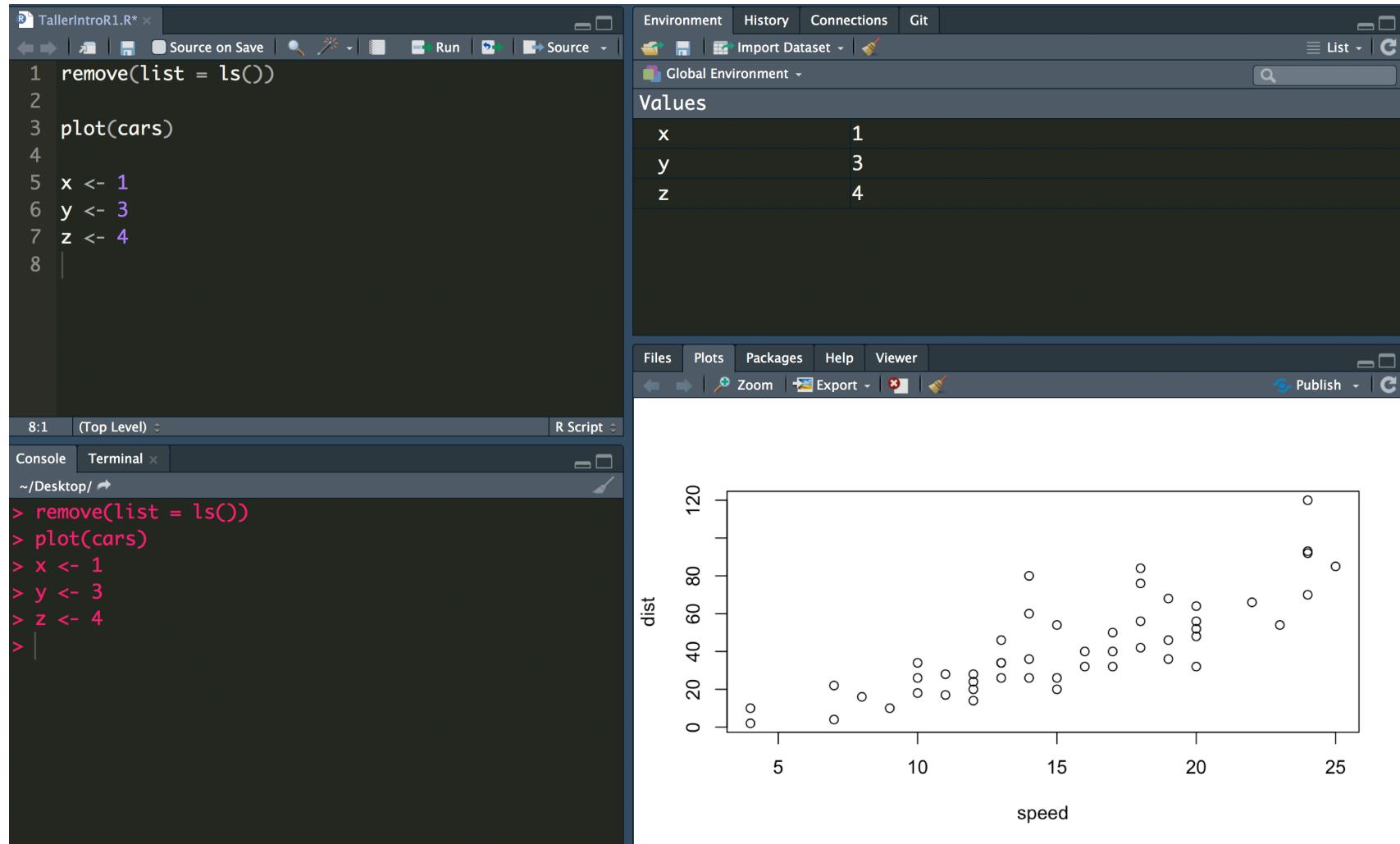
Asignar variables

```
x <- 1
```

```
y <- 3
```

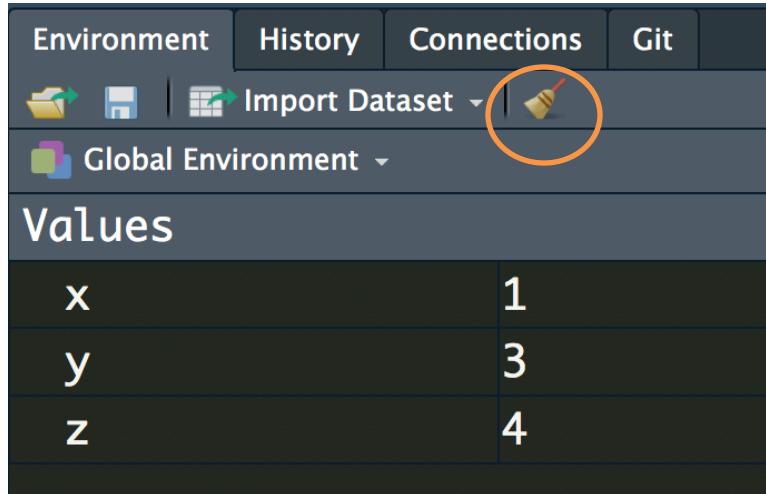
```
z <- 4
```

- Ahora deben estar en su global environment.



Limpiar el area de trabajo

- `rm(list=ls())`
- Buena practica limpiar todo!
- Otra forma:



The screenshot shows the RStudio interface with the 'Environment' tab selected. At the top, there are tabs for 'Environment', 'History', 'Connections', and 'Git'. Below the tabs, there are buttons for 'Import Dataset' and a 'Global Environment' dropdown. A brush icon, which is the clean workspace button, is highlighted with an orange circle. The main area displays a table titled 'Values' with three rows: x (1), y (3), and z (4).

Values	
x	1
y	3
z	4

Cerrar RStudio

- R pregunta si quiere guardar su “**Workspace Image**”
 - Significa que si quiere guardar los datos y variables
 - Generalmente no recomendado
 - Es mejor comenzar con un espacio de trabajo limpio y vacío para que los análisis anteriores no interfieran con los análisis actuales.
 - **Excepción:** trabajar con un conjunto de datos enorme

- RStudio
- **Cómo cargar los datos en R**
- **Tipos de datos en R**



Cargar los datos en R:

- paso 1: Guardar los datos en el formato correcto. 
- Paso 2: Dígale a R dónde buscar sus datos. 
- Paso 3: Cargar los datos en R. 

Paso 1: Guardar los datos en el formato correcto

- La forma más sencilla de guardar datos es como un archivo .csv

En Excel:

Archivo> Guardar como> .csv



- Limpia tus datos para R:
¡espacios, símbolos, celdas en blanco!

Qué está mal?

Filas y columnas vacías

	A	B	C	D	E
1					
2		Species	height i(n feet)	smells bad? (yes/no)	
3		unicorn		5	no
4		bearwock		10	yes
5		yeti		11.5	yes
6					
7					
8					
9					
10					
11					

Paréntesis, ?, /

Columnas en la derecha deben estar vacías

Listo!

The screenshot shows a Microsoft Excel spreadsheet with the following data:

	A	B	C	D
1	Species	height_ft	smells_bad	
2	unicorn		5 no	
3	bearwock		10 yes	
4	yeti		11.5 yes	
5				
6				
7				
8				
9				
10				
11				
12				
13				

Los datos
empiezan en la
esquina izquierda

Nombres sin
espacios o símbolos

2. Dígale a R dónde buscar

- Su directorio de trabajo es la carpeta en su computadora donde R buscará (y guardará) sus archivos

```
getwd () # le dice su directorio de trabajo  
actual
```

- Para cambiar (o "configurar") su directorio de trabajo, use la función:

```
setwd ("ruta-al-archivo-aquí")
```

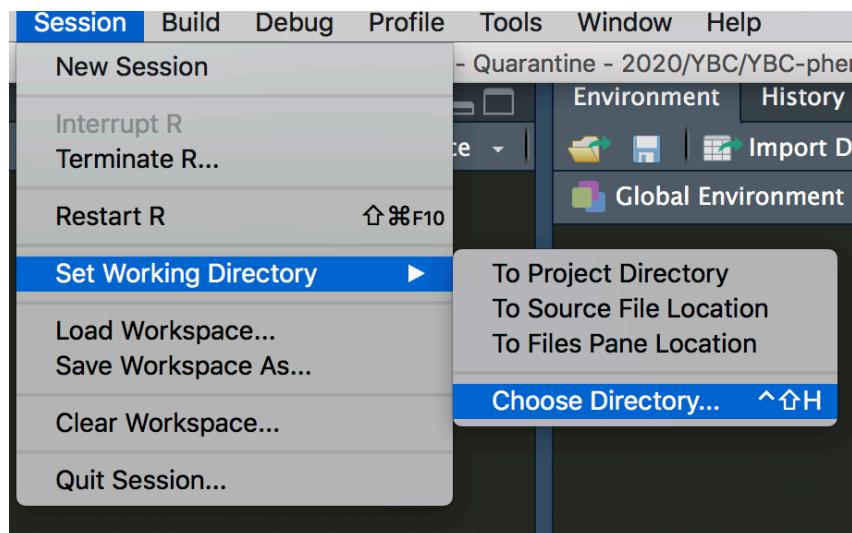
- Por ejemplo:

```
setwd (~ / Desktop) # establece su WD en su escritorio
```

Cómo encontrar la ruta ?

Para encontrar la ruta de su archivo, intente configurar interactivamente su directorio de trabajo en RStudio, luego copie el código.

Session > Set Working Directory > Choose Directory



A screenshot of the RStudio interface showing a script editor window titled 'TallerIntroR1.R' and a terminal window. The script editor contains the following R code:

```
1 #Paulo Izquierdo
2 #Enero 16, 2021
3 #Mi primer script
4 rm(list=ls())
5 getwd()
```

The terminal window shows the command: `> setwd("~/Desktop")`.

Paso 3: cargar los datos

La forma más fácil de cargar datos en R es usar la función **read.csv** y asignarla a un objeto:

```
x <- read.csv(file="FileName.csv")
```

```
# ver los datos
```

```
x
```

Diferencia entre un **WARNING** y un **ERROR**



- Un **WARNING** significa que R quiere advertir algo, pero el código aun funciona
- Un **ERROR** significa que el código no corre.

Tienes un ERROR o un WARNING y no sabe que es?



Opciones para leer archivos

- `header=T` la primera fila de datos son nombres de columna
- `sep = ","` como se separan las entradas en el archivo de texto
- `na.strings = NA` cuyos valores se tratan como NA
- `skip = 0` el número de líneas para saltar antes de leer los datos
- `nrows = -1` número de líneas de datos para leer (-1 significa todos)

Nombre de filas

Sin nombre

```
> read.csv("data1.csv")
```

	Species	height_ft	smells_bad
1	unicorn	5.0	no
2	bearwock	10.0	yes
3	yeti	11.5	yes

Con nombres!

```
> read.csv("data1.csv", row.names=1)
```

	height_ft	smells_bad
unicorn	5.0	no
bearwock	10.0	yes
yeti	11.5	yes

Muchas otras formas... Dependiendo de su tipo de datos

- Goooooogle es tu mejor amigo!

¿Cómo puedo introducir datos xxxx en R "

"Genómico"

"Filogenético"

"Rango"

"color"

"punto de referencia"

... ...



- Rstudio
- Cómo introducir sus datos en R
- **Tipos de datos en R**



Por qué nos preocupan los tipos de datos:

- R tiene diferentes formas de manejar diferentes tipos de datos
- Cuando carga sus datos en R, lo que puede HACER con esos datos depende del tipo de datos que R cree que le dio.
- En R, solo puede hacer ciertas cosas con ciertos tipos de datos
- Si intenta hacer algo con el tipo de datos incorrecto, ¡R no funciona!

```
> round("8.1111", 2)
```

- Error in round("8.1111", 2) : non-numeric argument to mathematical function

Arguments

x

a numeric vector

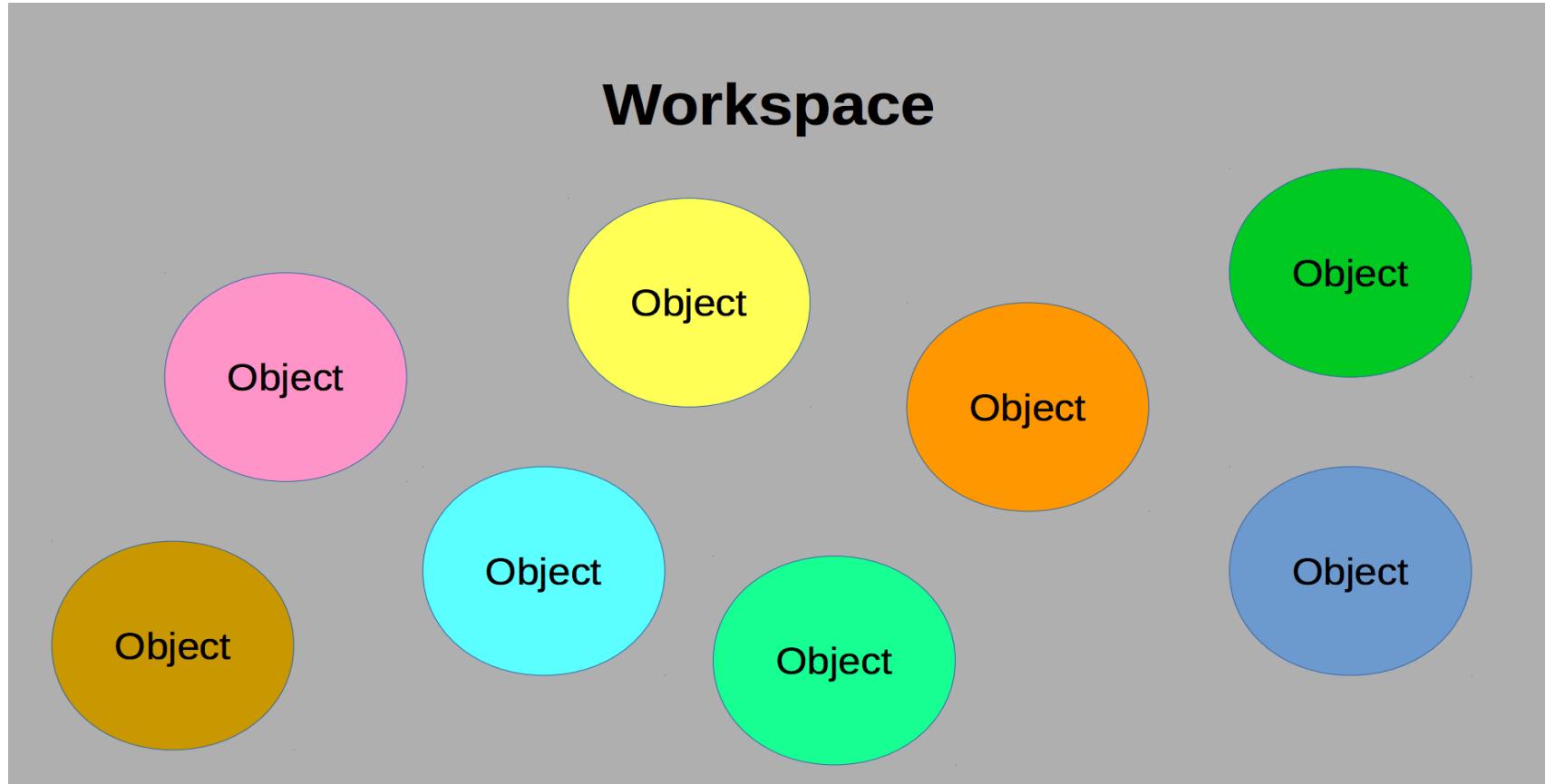
```
x <- "8.1111"  
class(x)  
[1] "character"
```

```
z <- 8.1111  
class(z)  
[1] "numeric"
```

¡La confusión de objetos de datos es una de las fuentes más comunes de frustración en R



Una vez que carga los datos en R, se guarda como un OBJETO



¿Cómo entender todos estos objetos?

Tipos de datos

Que clase son?

*Number? Integer?
Character?...*

Estructura

Como estan
agrupados?

*Vector? Matrix?
Array?...*

Tipos de datos

- **Numeric:** e.g., 15.5, 3.228
 - **Integers:** e.g., 1, 5
 - **Characters:** e.g., “hello”, “honey badger”
 - **Logical:** e.g., TRUE, FALSE
 - **Factors:** e.g., control, treatment (categorical)
-
- El tipo de datos no siempre es obvio en R, ¡pero saber cuál es puede ser importante!

```
> x <- c(1,1,1,1) #que tipo de datos es?
```

Logical object

TRUE or FALSE

- Utilice los operadores de comparación para crear datos lógicos

5 < 2

[1] FALSE

<	Menos que
>	Mayor que
==	Igual a
!=	No es igual

Characters vs Factors

- Un **factor** tiene valores discretos (categóricos):
rana, sapo (2 niveles)
Demócrata, Republicano, No Afiliado (3 niveles)
- Un carácter contiene cadenas de texto
"A", "hola", "Vitis vinifera"

Generemos algunos objetos

```
> my_numeric <- 42.5  
  
> my_character <- "universe"  
  
> my_logical <- TRUE  
  
> my_factor <- factor(my_character)
```

*Podemos convertir una variable en un factor categórico con el comando `factor ()`

Esquina superior derecha del espacio de trabajo de Rstudio:

Values

my_character

"universe"

my_factor

Factor w/ 1 level "universe": 1

my_logical

TRUE

my_numeric

42.5

Que tipo de dato es?

R tiene funciones para explorar características de los objetos :

`str()` es una función genérica que te dice que tipo de datos y estructura tiene un objeto

```
> str(my_numeric)
```

```
> str(my_character)
```

```
> str(my_logical)
```

```
> str(my_factor)
```

También pueden usar `class()`

Cambiar los tipos de datos

- Puede usar “as.function” para coaccionar datos de un tipo a otro tipo de datos.

```
> my_numeric
```

```
[1] 42.5
```

```
> as.character(my_numeric)
```

```
[1] "42.5"
```

```
> as.factor(my_character)
```

```
[1] universe
```

```
Levels: universe
```

Tipos de datos

Que clase son?

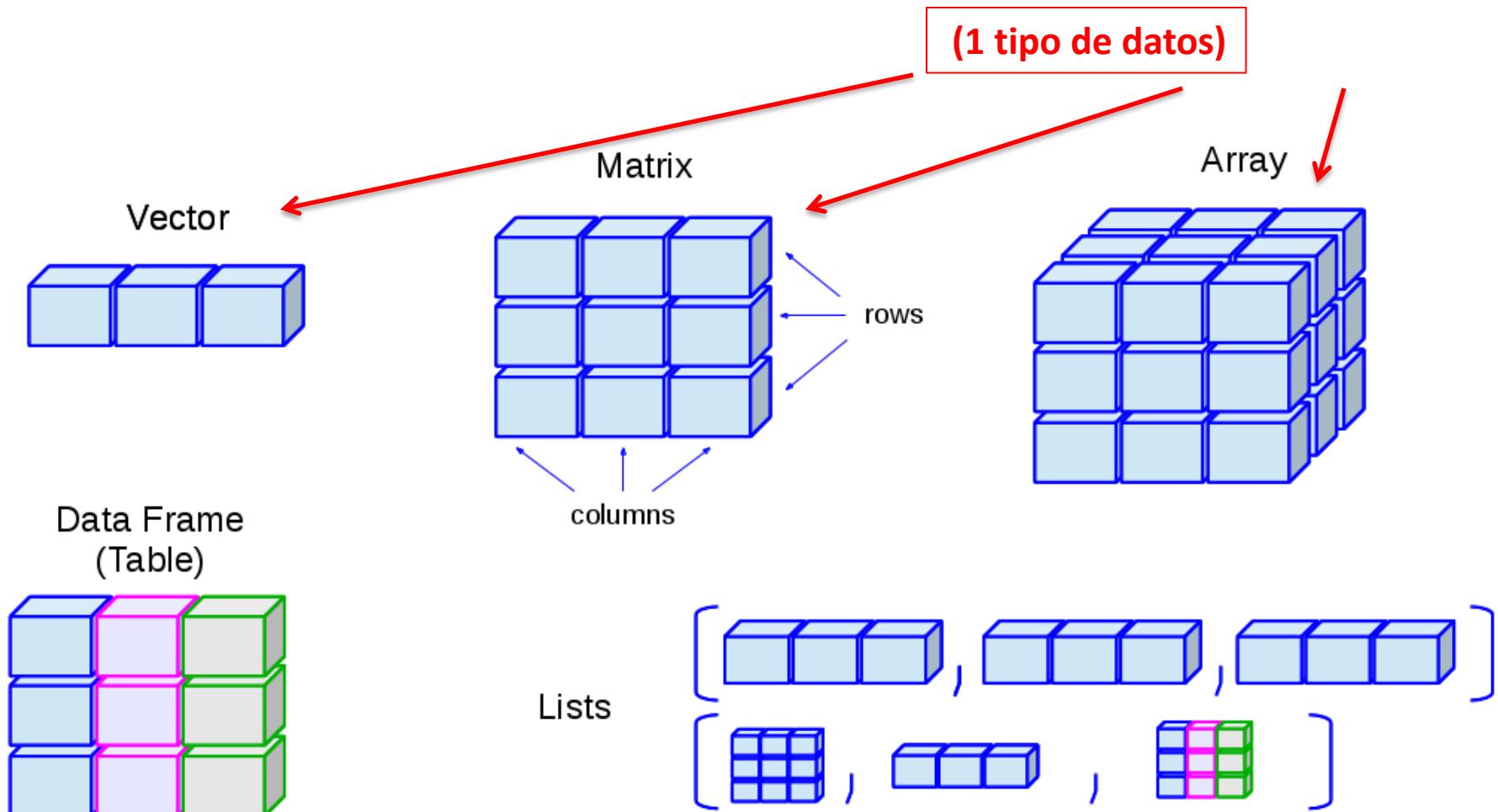
*Number? Integer?
Character?...*

Estructura

Como estan
agrupados?

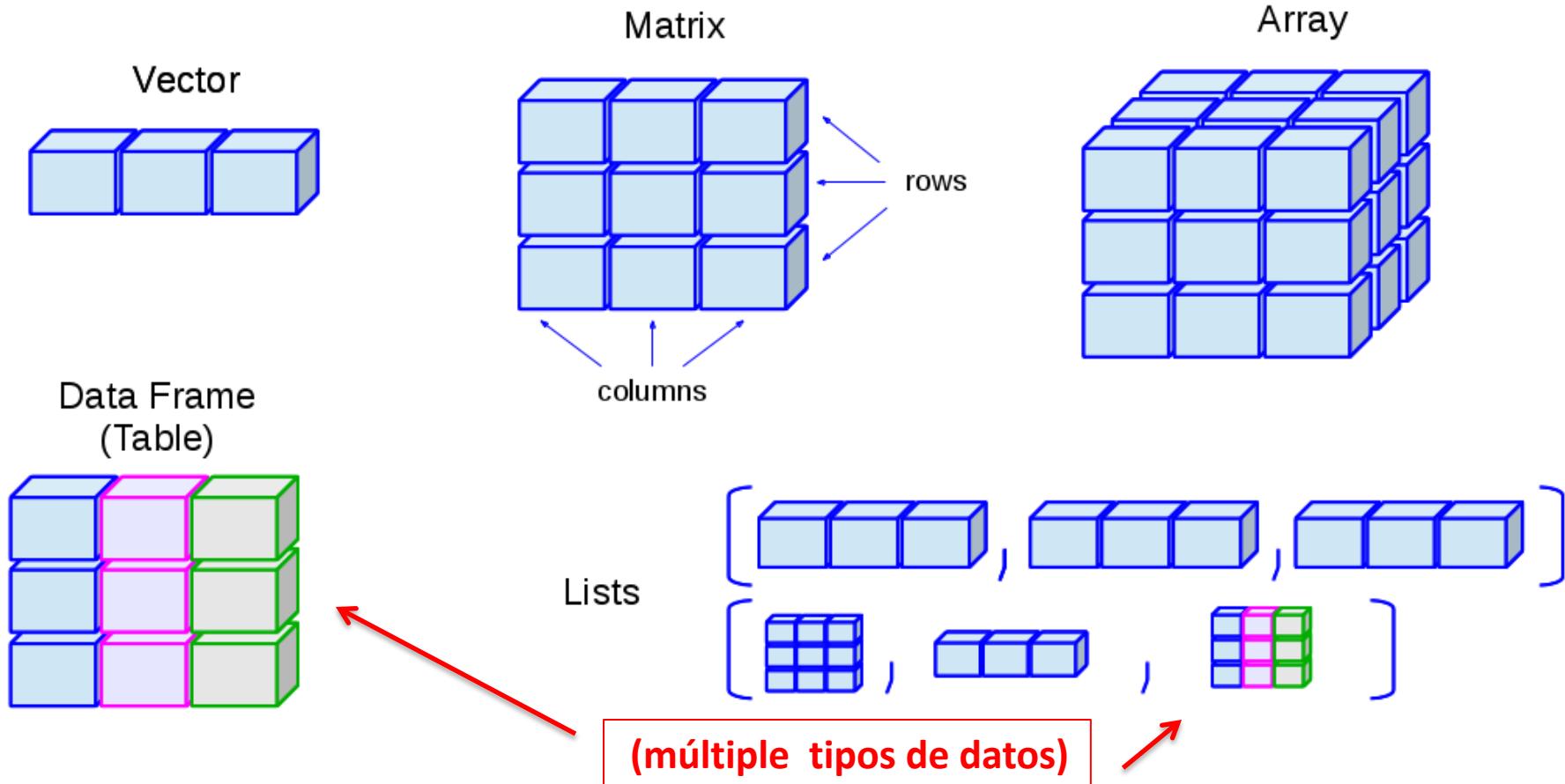
*Vector? Matrix?
Array?...*

Puede juntar datos en diferentes estructuras de datos



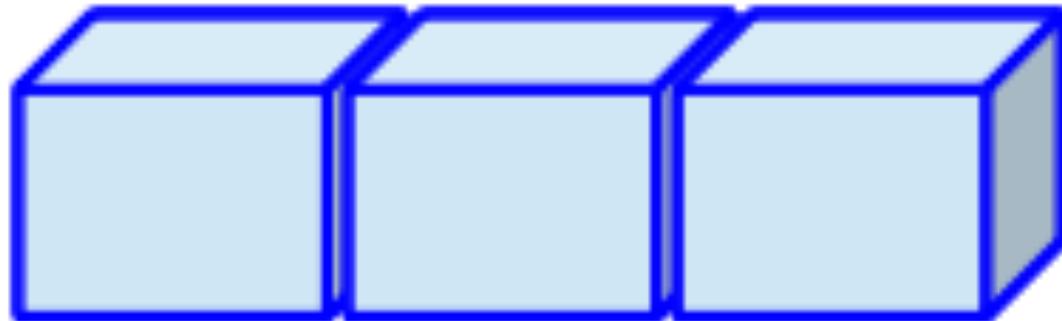
You can put data together into different data structures

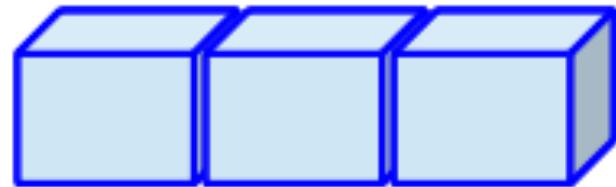
- Common data structures in R :



Vectors

- Una sola columna o fila de datos
- 1 tipo de datos



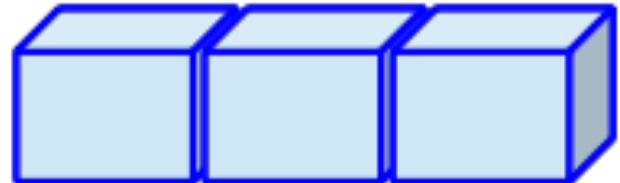


x <- 4.5

a <- c(1, 2, 5.3, 6, -2, 4)

c () concatenar

Ejemplos



```
# numeric
```

```
a <- c(1,2,5.3,6,-2,4)
```

```
# character
```

```
b <- c("one", "two", "three")
```

```
# factor
```

```
c <- factor(c("one", "two", "three"))
```

```
# logic
```

```
d <- c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)
```

Funciones para generar vectores numéricos

Números consecutivos

```
1:10
```

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

```
seq(from=1, to=10, by=2)
```

```
seq(1,10,2)
```

```
1 3 5 7 9
```

```
seq(from=1, to=10, length.out=5)
```

```
1.00 3.25 5.50 7.75 10.00
```

```
> rep(3, times=10)
[1] 3 3 3 3 3 3 3 3 3 3
```

```
> y <- 1:3
[1] 1 2 3
```

```
> rep(y, times=4)
[1] 1 2 3 1 2 3 1 2 3 1 2 3
```

```
> rep(y, length=10)
[1] 1 2 3 1 2 3 1 2 3 1
```

Logical vectors

```
a <- c(1,2,3,4,5)
```

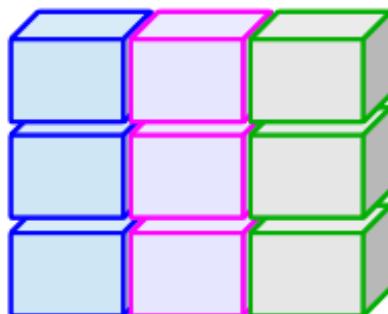
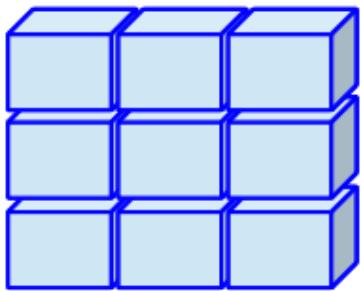
```
[1] 1 2 3 4 5
```

```
a>3
```

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

```
a>=3
```

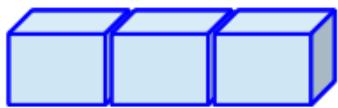
```
[1] FALSE FALSE TRUE TRUE TRUE
```



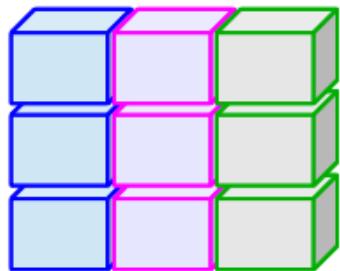
Los vectores son
geniales, pero es
conveniente
almacenar datos
como una
colección de
variables

Data frames and matrices!

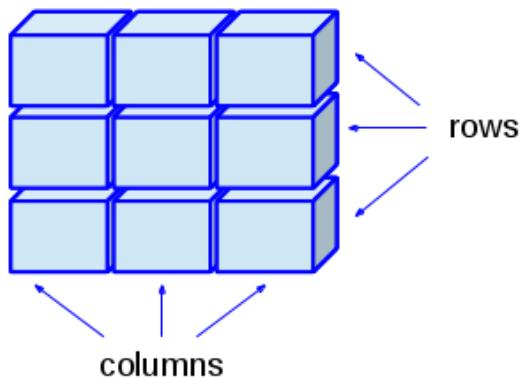
Vector



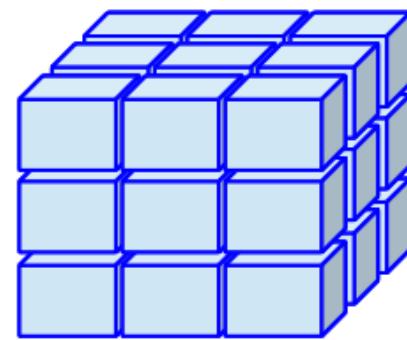
Data Frame
(Table)



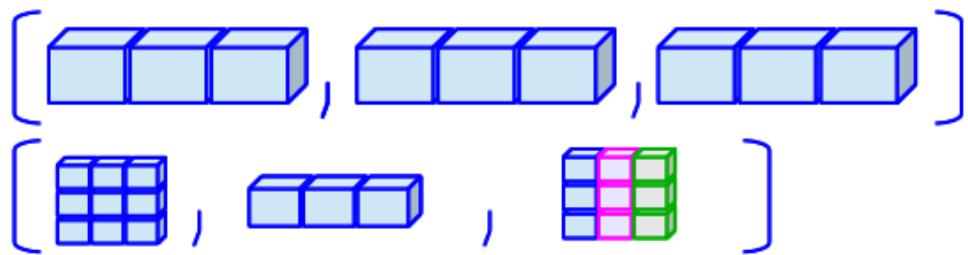
Matrix



Array

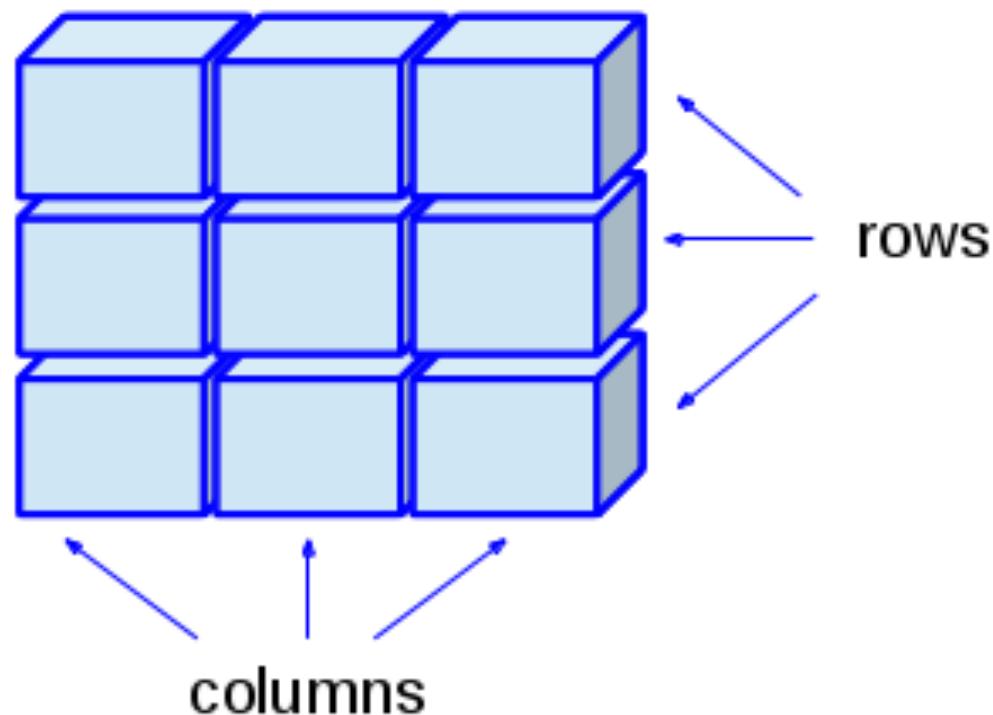


Lists



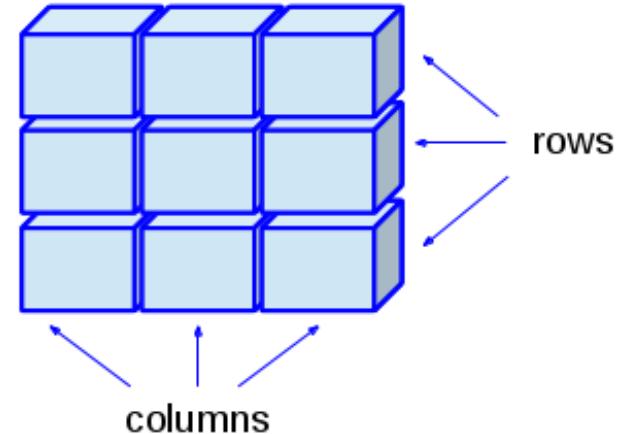
Matrix

- Varias columnas y / o filas de datos
- Un solo tipo de datos.



Matrix

Construir una matriz con:
`matrix()`



```
> my_matrix <- matrix(data=c(1:6), Numbers to put into matrix  
                      nrow=3, ncol=2,      Number of rows, number of columns  
                      byrow=FALSE)      Fill numbers in by column  
  
> my_matrix  
     [,1] [,2]  
[1,]    1    4  
[2,]    2    5  
[3,]    3    6
```

Enlazar vectores en matrices usando enlace de columna (cbind) y enlace de fila (rbind)

```
> x<- c(TRUE, FALSE, TRUE)  
> y<- c("a", "b", "c")  
> z<- c(4, 5, 7)
```

```
> cbind(x, y, z)  
      x     y   z  
[1,] "TRUE" "a"  "4"  
[2,] "FALSE" "b"  "5"  
[3,] "TRUE"  "c"  "7"
```

```
> rbind(x, y, z)  
 [,1] [,2] [,3]  
x "TRUE" "FALSE" "TRUE"  
y "a"    "b"    "c"  
z "4"    "5"    "7"
```

Data

DF_example 3 obs. of 3 variables
x: logi TRUE FALSE TRUE
y: Factor w/ 3 levels "a", "b", "c": 1 2 3
z: num 4 5 7

```
> str(p)
```

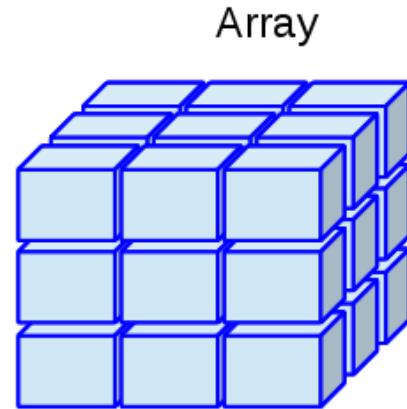
chr [1:3, 1:3] "TRUE" "a" "4" "FALSE" "b" ...
- attr(*, "dimnames")=List of 2
..\$: chr [1:3] "x" "y" "z"
..\$: NULL

Arrays

Los arrays son los objetos de datos R que pueden almacenar datos en más de dos dimensiones.

```
vector1 <- c(5,9,3)
vector2 <- c(10,11,12,13,14,15)

>my_array <-
array(c(vector1, vector2),
      dim = c(3,3,3))
```



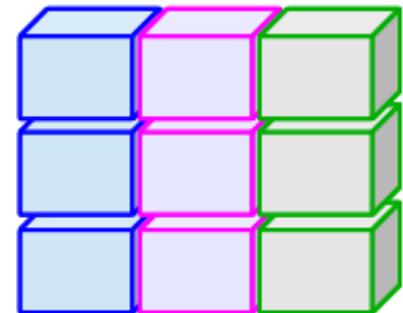
```
,, 1
[,1] [,2] [,3]
[1,] 5 10 13
[2,] 9 11 14
[3,] 3 12 15
```

```
,, 2
[,1] [,2] [,3]
[1,] 5 10 13
[2,] 9 11 14
[3,] 3 12 15
```

```
,, 3
[,1] [,2] [,3]
[1,] 5 10 13
[2,] 9 11 14
[3,] 3 12 15
```

Data frames

Data Frame
(Table)



- 2 dimensiones
- Múltiples tipos de datos

```
> x<- c(TRUE, FALSE, TRUE)  
> y<- c("a", "b", "c")  
> z<- c(4, 5, 7)  
  
> my_dataframe<- data.frame(x, y, z)
```

	x	y	z
1	TRUE	a	4
2	FALSE	b	5
3	TRUE	c	7

```
> str(my_dataframe)
```

```
'data.frame':      3 obs. of  3 variables:  
 $ x: logi  TRUE FALSE TRUE  
 $ y: chr  "a" "b" "c"  
 $ z: num  4 5 7
```

Data frames

```
> x<- c(TRUE, FALSE, TRUE) # 3
```

```
> y<- c('a', 'b', 'c') #3
```

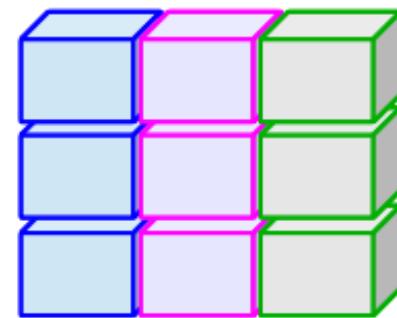
```
> z<- c(4, 5) #2
```

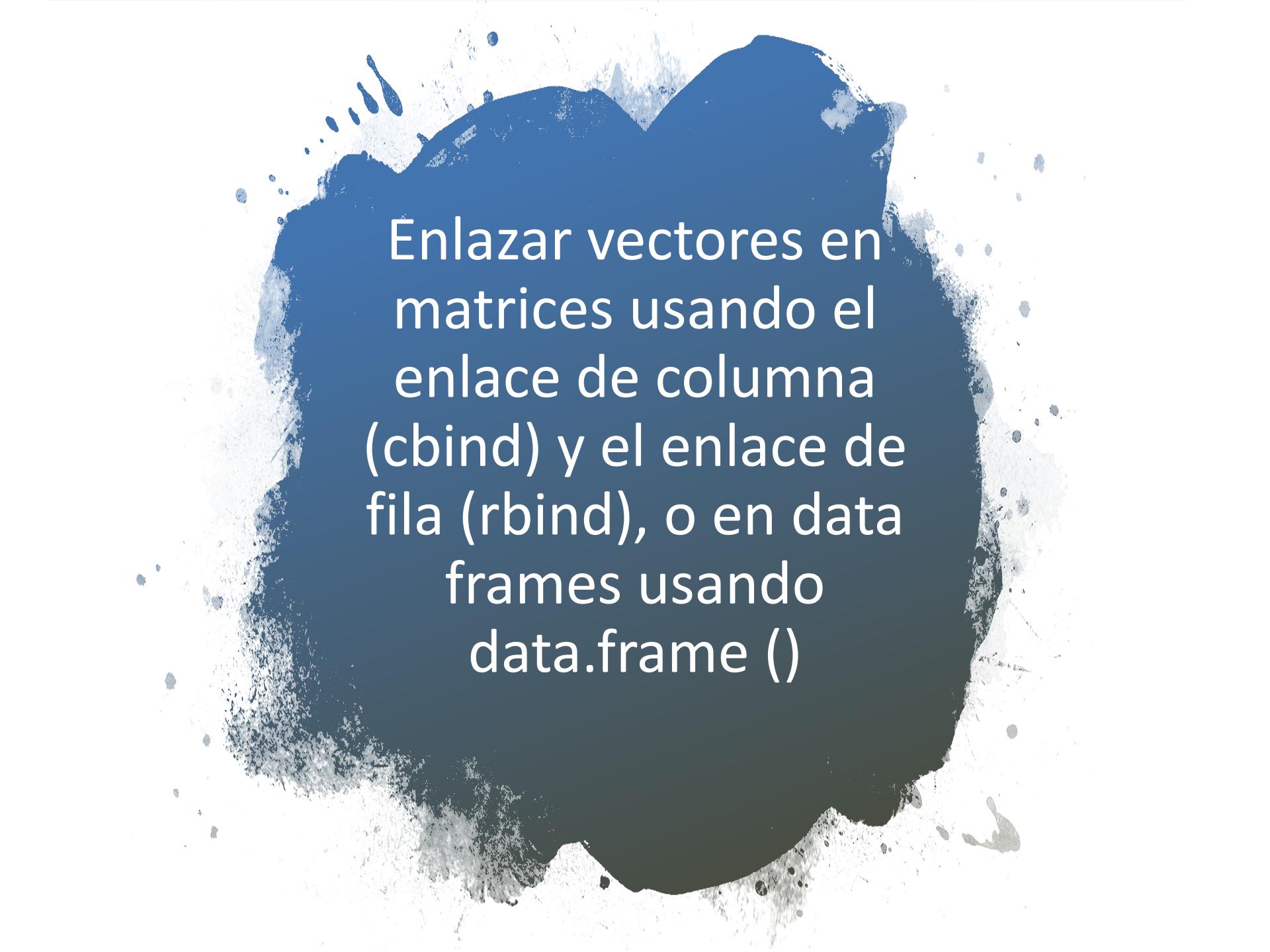
```
> my_dataframe<- data.frame(x, y, z)
```

Error in data.frame(x, y, z) :

arguments imply differing number of rows: 3, 2

Data Frame
(Table)





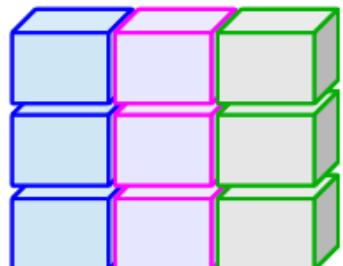
Enlazar vectores en
matrices usando el
enlace de columna
(cbind) y el enlace de
fila (rbind), o en data
frames usando
data.frame ()

Ahora otra estructura!

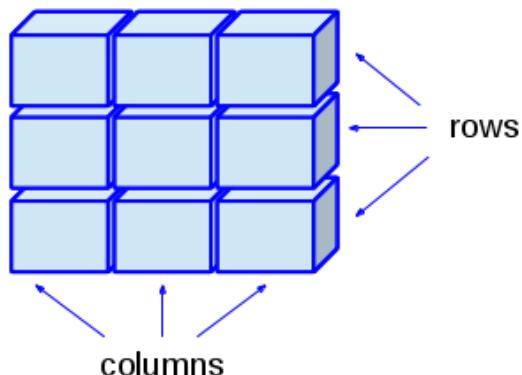
Vector



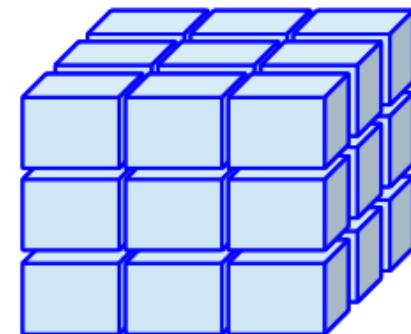
Data Frame
(Table)



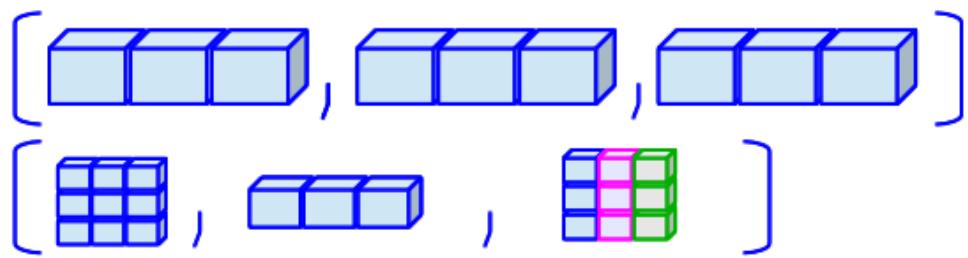
Matrix



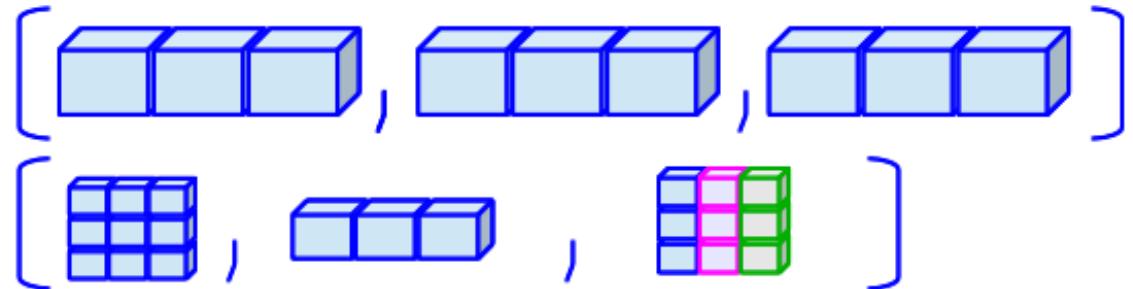
Array



Lists



List



- Una lista es una colección de objetos, donde los elementos pueden ser diferentes tipos de objetos.
- Un objeto de objetos....



Definir una lista usando: list()

```
> my_list<- list(  
  my_dataframe,  
  my_character,  
  my_matrix)
```

```
> my_list
```

```
[[1]] my_dataframe  
      x y z  
1 TRUE a 4  
2 FALSE b 5  
3 TRUE c 7
```

```
[[2]] my_character  
[1] "universe"
```

```
[[3]] my_matrix  
      [,1] [,2]  
[1,]    1    4  
[2,]    2    5  
[3,]    3    6
```

```
> str(my_list)
```

my_list

List of 3

: 'data.frame': 3 obs. of 3 variables:

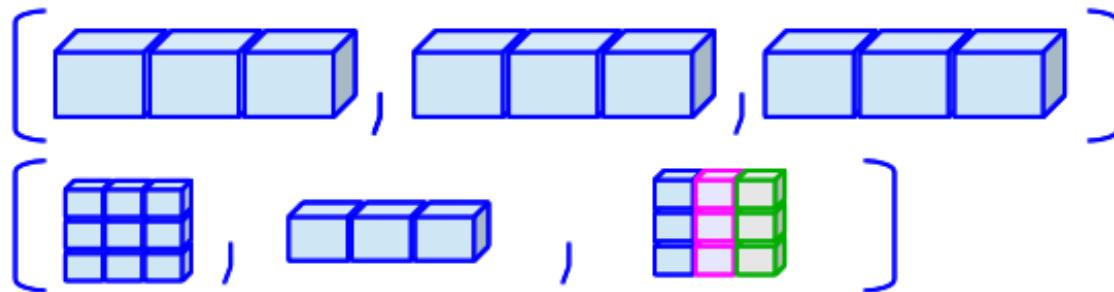
..\$ x: logi [1:3] TRUE FALSE TRUE

..\$ y: Factor w/ 3 levels "a","b","c": 1 2 3

..\$ z: num [1:3] 4 5 7

: chr "universe"

: int [1:3, 1:2] 1 2 3 4 5 6



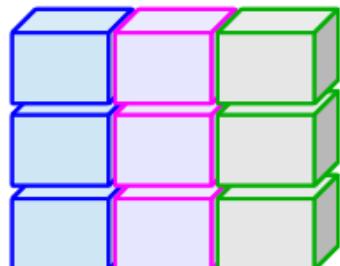
Estructuras!



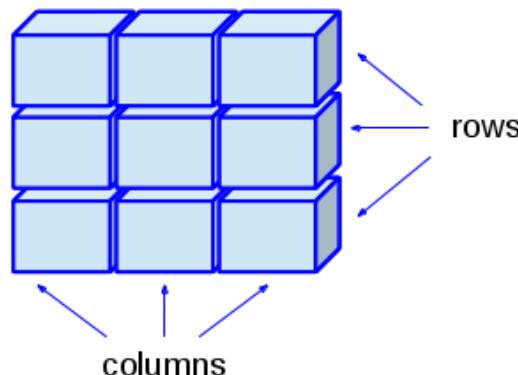
Vector



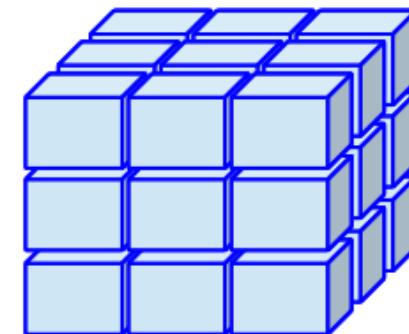
Data Frame
(Table)



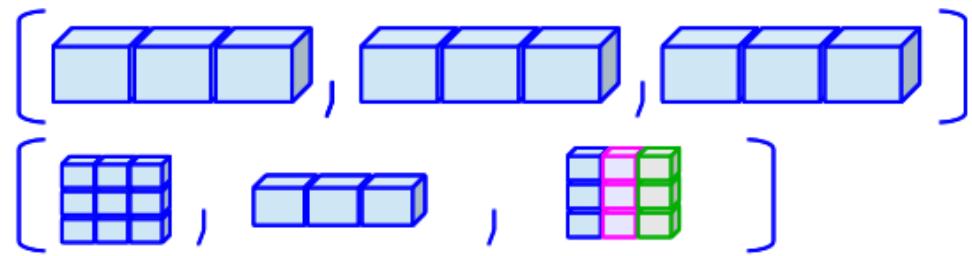
Matrix



Array



Lists



Confusión en la estructura de datos: una fuente de errores muy común

Si es la estructura incorrecta, ¡conviértala en la estructura correcta!

```
> as.data.frame()  
> as.matrix()
```



Qué aprendimos?

- Rstudio!
- Cómo cargar datos en R
- ¿Cuáles son las estructuras de datos comunes en R?
- Cómo crear vectores
- Cómo verificar y cambiar el tipo de datos
- Cómo unir vectores en matrices, dataframes, lists