



Introducción a la ciencia de datos

Capítulo 2: Tipos y estructuras de datos

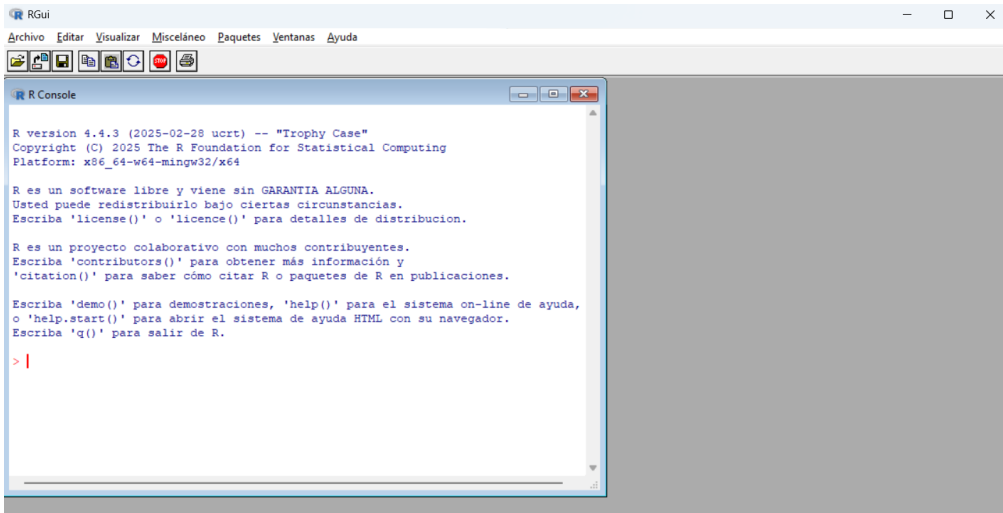
Mg. J. Eduardo Gamboa U.

2025-03-18

Primeros pasos en R y Rstudio

R

- ▶ Es un entorno y lenguaje de programación usado ampliamente para el análisis de datos.
- ▶ Es parte del sistema GNU y se distribuye bajo la licencia GNU GPL, es de código abierto.
- ▶ Al instalar viene con muchas funcionalidades incluidas, sin embargo algunas tendrán que adicionarse a través de paquetes.
- ▶ Link de descarga (última versión disponible 4.4.3):
<https://cran.r-project.org/bin/windows/base/> (Windows)
<https://cran.r-project.org/bin/> (Otros sistemas operativos)



Rstudio

- ▶ Es un entorno de desarrollo integrado (IDE) que permite manipular código en lenguaje R.
- ▶ Es necesario previamente haber instalado R.
- ▶ Permite una mejor gestión del espacio de trabajo y la edición de códigos
- ▶ Link de descarga:
<https://posit.co/download/rstudio-desktop/#download>
<https://docs.posit.co/previous-versions/> (si tu procesador es de 32 bits)

2025-1 - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Capítulo 1 - ICD.qmd Capítulo 2 - ICD.qmd Untitled1

Source on Save Run

1

Environment History Connections Tutorial

Import Dataset 38 MB

R Global Environment

Environment is empty

Console Terminal Background Jobs

R 4.4.1 - G:/Mi unidad/Cursos/Introduccion a la Ciencia de Datos/Clases/2025-1/

R es un software libre y viene sin GARANTIA ALGUNA.
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

>

Files Plots Packages Help Viewer Presentation

New Folder New Blank File Delete Rename More

G > Mi unidad > Cursos > Introduccion a la Ciencia de Datos > Clases > 2025-1

Name	Size	Modified
..		
.Rhistory	0 B	Mar 8, 2025, 2:11 PM
2025-1.Rproj	218 B	Mar 8, 2025, 2:10 PM
Capítulo 1 - ICD.qmd	9.6 KB	Mar 8, 2025, 2:17 PM
Capítulo 2 - ICD.qmd	9.6 KB	Mar 8, 2025, 2:17 PM
Capítulo-1---ICD.pdf	10.9 MB	Mar 8, 2025, 2:18 PM
images		
logo.png	23.2 KB	Feb 23, 2025, 10:14 PM
logop.jpg	14.6 KB	Feb 23, 2025, 10:25 PM
Capítulo-2---ICD.pdf	16 KB	Mar 8, 2025, 2:37 PM

Entorno de RStudio

The image shows the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu bar are several toolbars. The main workspace is divided into three panels, each highlighted with a colored border:

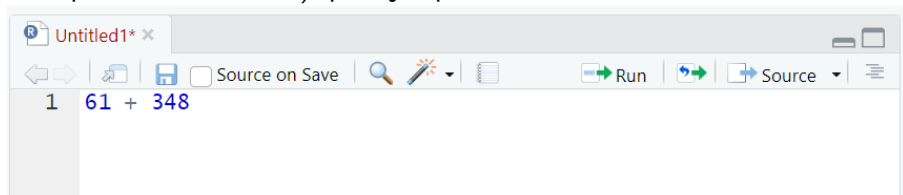
- Editor de código** (red border): The top-left panel where code is written. It contains a single line of code: `1`.
- Detalles del entorno de trabajo** (blue border): The top-right panel showing the environment. It displays "Environment is empty" and "Global Environment".
- Consola** (green border): The bottom-left panel showing the command line. It contains the following text:

```
usted puede redistribuirlo bajo ciertas circunstancias.  
Escriba 'license()' o 'licence()' para detalles de distribución.  
  
R es un proyecto colaborativo con muchos contribuyentes.  
Escriba 'contributors()' para obtener más información y  
'citation()' para saber cómo citar R o paquetes de R en publicaciones.  
  
Escriba 'demo()' para demostraciones. 'help()' para el sistema on-line de ayuda,  
o 'help.start()' para abrir el sitio web de ayuda en su navegador.  
Escriba 'q()' para salir de R.  
  
Registered S3 method overwritten by 'quantmod':  
  method      from  
as.zoo.data.frame zoo  
[workspace loaded from ~/.RData]
```

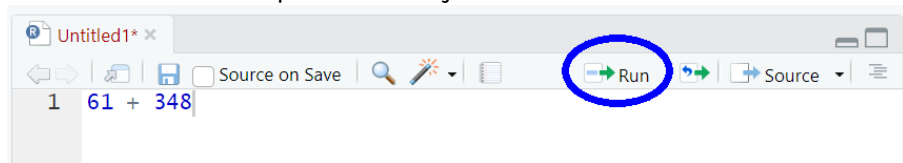
Otros: gráficas, ayuda, paquetes, visor, etc (yellow border): The bottom-right panel, which is currently empty, representing other features like plots, help, and package management.

Operaciones aritméticas

1. Escribe en el editor de código una operación aritmética (suma, resta, multiplicación o división), por ejemplo $61+348$:



2. Haz click en Run o usar Ctrl + Enter en el teclado, siempre y cuando el curso se encuentre en la línea que se desea ejecutar. No es necesario seleccionar el código.



3. Verás la respuesta en la Consola (panel inferior izquierdo)

R es un proyecto colaborativo con muchos contribuyentes.

Escriba 'contributors()' para obtener más información y

'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,

o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.

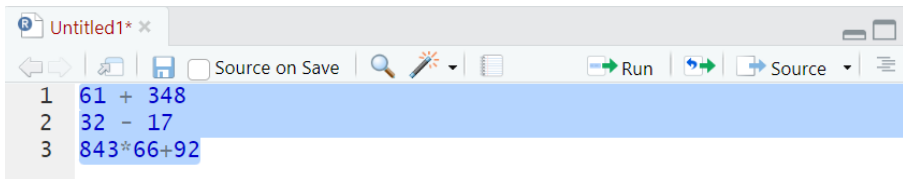
Escriba 'q()' para salir de R.

```
> 61 + 348
```

```
[1] 409
```

```
> |
```


Si quieres ejecutar más de una línea código sí será necesario seleccionar las líneas antes de hacer click en Run o Ctrl + Enter.



The screenshot shows the RStudio interface. At the top, there is a tab labeled 'Untitled1*' with a close button. Below the tab is a toolbar containing icons for navigation (back, forward), clipboard (copy, paste), save, a checkbox for 'Source on Save', search (magnifying glass), a lightning bolt icon, a list icon, a 'Run' button (green arrow), a 'Source' button (blue arrow), and a menu icon. The main editor area contains three lines of R code, each preceded by a line number (1, 2, 3). The code is: `61 + 348`, `32 - 17`, and `843*66+92`. All three lines of code are highlighted with a blue selection background.

```
1 61 + 348
2 32 - 17
3 843*66+92
```

R es un software libre y viene sin GARANTIA ALGUNA.
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

```
> 61 + 348
[1] 409
> 61 + 348
[1] 409
> 32 - 17
[1] 15
> 843*66+92
[1] 55730
```

Ejemplos

```
# Suma
```

```
32.2 + 64.5
```

```
[1] 96.7
```

```
# Resta
```

```
526.21 - 66.4882
```

```
[1] 459.7218
```

```
# Multiplicación
```

```
842 * 13.54
```

```
[1] 11400.68
```

Ejemplos

```
# División
```

```
50/7
```

```
[1] 7.142857
```

```
# División entera
```

```
50%/%7
```

```
[1] 7
```

```
# Residuo
```

```
50%%7
```

```
[1] 1
```

Ejemplos

```
# Potencia
```

```
3^2
```

```
[1] 9
```

```
3**2
```

```
[1] 9
```

```
# Potencia base e
```

```
exp(1.4)
```

```
[1] 4.0552
```

```
# Valor absoluto
```

```
abs(-4)
```

```
[1] 4
```

Ejemplos

```
# Raíz cuadrada  
sqrt(81)
```

```
[1] 9
```

```
# Raíz n, por ejemplo n = 3 para raíz cúbica  
125^(1/3)
```

```
[1] 5
```

Ejemplos

```
# Logaritmo base 10
```

```
log10(100)
```

```
[1] 2
```

```
log(100, 10)
```

```
[1] 2
```

```
# Logaritmo base e
```

```
log(100)
```

```
[1] 4.60517
```

```
log(100, exp(1))
```

```
[1] 4.60517
```

Ejemplo

```
# Logaritmo base 2
```

```
log2(8)
```

```
[1] 3
```

```
log(8, 2)
```

```
[1] 3
```

```
# Logaritmo base n, por ejemplo n = 6
```

```
log(36, 6)
```

```
[1] 2
```


Ejercicios

Resuelva las siguientes operaciones combinadas:

1. Residuo de dividir $(56+6)^2$ entre la raíz cuadrada de 25.
2. Parte entera de dividir 300 entre la raíz cúbica de 343.
3. Calcular el cociente entre el doble del valor de π y la raíz cuadrada de 16.
4. Calcular el valor absoluto de la diferencia entre la raíz cuarta de 81 y la raíz cúbica de 64.

Asignación de variables

Los valores que se ingresan a RStudio o que son el resultado de una operación pueden ser almacenados en una variable, por ejemplo:

```
# números
x = 3
x <-3
y = 36 - 9*2 + 5*log10(40)

# caracteres
z = 'hola'
m <- "universidad agraria"
```

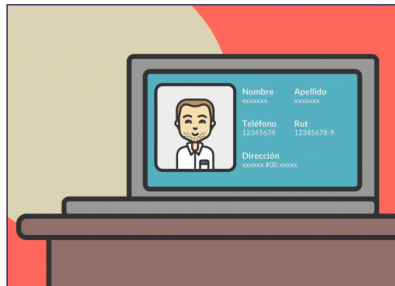
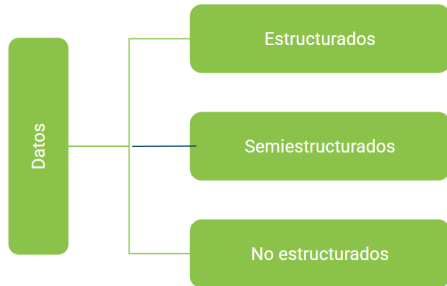
Proceso de guardado

En RStudio, se pueden guardar distintos tipos de archivos según su función:

1. Archivos de código R (.R): Guarda solo el código, no los resultados de la ejecución.
2. Archivos de datos (.RData): Almacena la memoria de trabajo (environment), incluyendo todas las variables y objetos creados, pero no los códigos. No recomendado para reproducir análisis debido a la posible pérdida de contexto.
3. Archivos de historial (.Rhistory): Guarda el historial de todos los comandos ejecutados en la consola.
4. Archivos de proyecto (.Rproj): Facilita la organización y gestión de archivos, datos y scripts dentro de un proyecto.

Datos

Un dato es el registro de un hecho. No es solamente un número o una palabra. Puede ser también un texto (oración, párrafo, etc), una imagen, un audio, un video, etc.



UNSTRUCTURED DATA



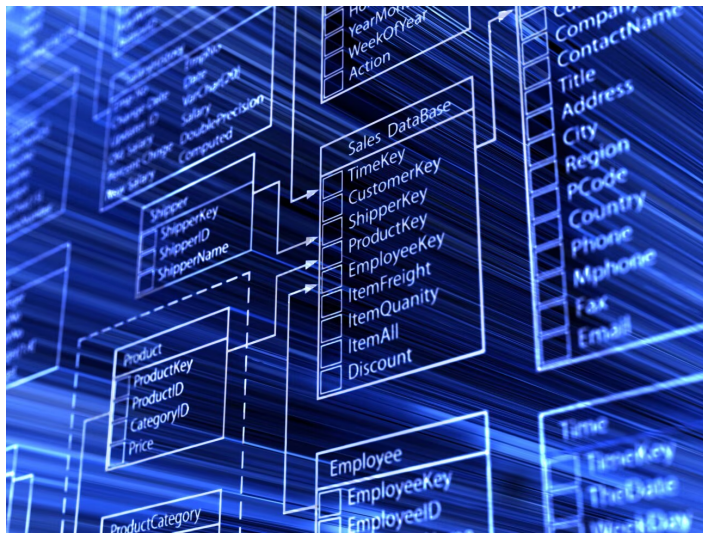
VS

STRUCTURED DATA



Datos estructurados

- ▶ Pueden ser dispuestos en un formato tabla (filas y columnas).
- ▶ Son los tipos de datos más sencillos, pero no los más comunes de encontrar.
- ▶ Se almacenan en bases de datos relacionales y se consultan a través de SQL (Structured Query Language).
- ▶ Formatos más comunes de almacenamiento: .csv, .txt, .xlsx, .xls



- ▶ Los datos son procesados mediante programas computacionales, desarrollados en distintos lenguajes de programación.
- ▶ En este curso, trabajamos con R, un lenguaje diseñado para análisis de datos y computación estadística.

Principales tipos de datos en R

- ▶ Integer → Números enteros
- ▶ Double → Números decimales (doble precisión)
- ▶ Complex → Números complejos
- ▶ Logical → Valores lógicos (TRUE / FALSE)
- ▶ Character → Texto o cadenas de caracteres
- ▶ Date → Fechas

A su vez, estos datos pueden agruparse en objetos como vectores, matrices, data.frames, etc.

Datos tipo integer

Permite representar números enteros

```
x = 4L  
typeof(x)
```

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

```
is.integer(x)
```

```
[1] TRUE
```

```
y = 4  
typeof(y)
```

```
[1] "double"
```

```
is.integer(y)
```

```
[1] FALSE
```

Datos tipo double

Permite representar números reales (permite decimales).

```
x = 36  
typeof(x)
```

```
[1] "double"
```

```
is.double(x)
```

```
[1] TRUE
```

```
y = 40.3  
typeof(y)
```

```
[1] "double"
```

```
is.double(y)
```

```
[1] TRUE
```

¿Qué sucederá al ejecutar estos códigos?

```
m = 65  
is.integer(m)
```

Datos tipo complex

Permite representar números complejos.

```
z = 1 + 2i
```

```
typeof(z)
```

```
[1] "complex"
```

```
is.complex(z)
```

```
[1] TRUE
```

```
u = 1i
```

```
typeof(u)
```

```
[1] "complex"
```

```
is.complex(u)
```

```
[1] TRUE
```

```
w = 10 + 0i  
typeof(w)
```

```
[1] "complex"
```

```
is.complex(w)
```

```
[1] TRUE
```

¿Qué sucederá al ejecutar estos códigos?

```
v = 10  
typeof(v)  
is.integer(v)  
is.double(v)  
is.complex(v)
```

Datos tipo logical

Permite representar datos lógicos o booleanos, los cuales admiten los valores verdadero (TRUE) o falso (FALSE), ¿qué sucede al ejecutar estos códigos?

```
a = TRUE  
typeof(a)
```

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

```
is.logical(a)
```

```
[1] TRUE
```

¿Qué sucederá al ejecutar estos códigos?

```
3 == 5.2
```

```
4.6 <= 8.1
```

```
12 > 3.5
```

```
5 != 8
```

```
b = (4 == 5)
```

Datos tipo character

Permite representar datos como cadenas no numéricas, es decir letras, símbolos y/o valores alfanuméricos. Estos datos ya se podrían considerar como no estructurados, ¿qué sucede al ejecutar estos códigos?

```
p = 'Universidad Agraria'  
typeof(p)
```

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

```
is.numeric(p)
```

```
[1] FALSE
```

```
is.complex(p)
```

```
[1] FALSE
```

```
is.logical(p)
```

```
[1] FALSE
```

```
is.character(p)
```

```
[1] TRUE
```

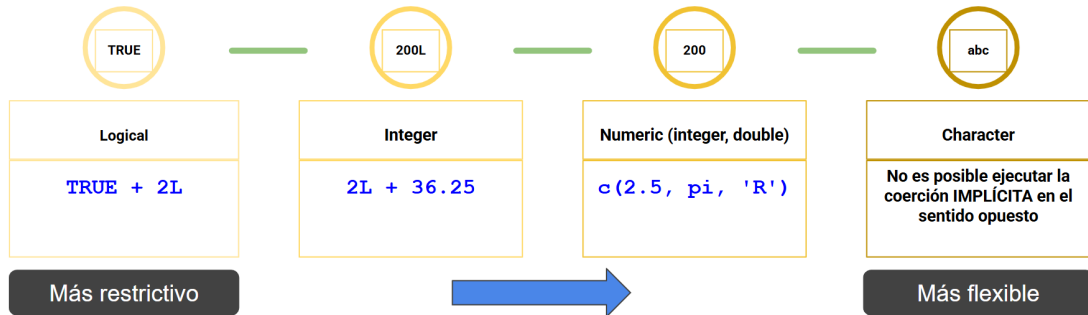
Verificar qué tipo de dato corresponde a q y s

```
q = '70096321'
```

```
s = 'R is a free software environment for statistical computing  
and graphics. To download R, please choose your preferred CRAN  
mirror.'
```


Coerción de datos

Podemos forzar **implícitamente** el cambio de tipo de un dato, siguiendo este orden:



También es posible realizar la coerción explícita a través de las funciones:

```
as.logical(...)  
as.integer(...)  
as.double(...)  
as.complex(...)  
as.numeric(...)  
as.character(...)
```

Recuerde que NA = Not Available (R no logró realizar la coerción).

Colecciones o estructuras de datos en R

Los datos se agrupan en estructuras definidas según su dimensión y homogeneidad.

Así, tenemos:

- ▶ Atomic vector (vector)
- ▶ Matrix (matriz)
- ▶ Array (arreglo)
- ▶ List (lista)
- ▶ Data frame

single type

multiple types

1D

Vector

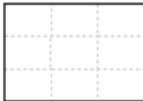


List

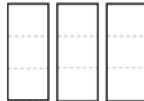


2D

Matrix

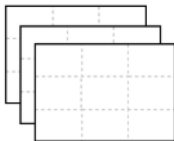


Data frame



nD

Array



Vector (atómico)

Se trata de una colección de n datos del mismo tipo (Integer, Double, Logical, Character, Complex). De ser necesario, aplica coerción implícita. Por ejemplo:

```
x1 = c(3, 7, 5, 3.4, pi, log10(33))
```

```
x1
```

```
[1] 3.000000 7.000000 5.000000 3.400000 3.141593 1.518514
```

```
typeof(x1)
```

```
[1] "double"
```

```
str(x1)
```

```
num [1:6] 3 7 5 3.4 3.14 ...
```

```
is.double(x1)
```

```
[1] TRUE
```

```
is.atomic(x1)
```

```
[1] TRUE
```

(cont.)

```
x1 = c(3, 7, 5, 3.4, pi, log10(33))
```

```
x1[1]
```

```
[1] 3
```

```
x1[1:3]
```

```
[1] 3 7 5
```

```
x1[c(2,3,6,7)]
```

```
[1] 7.000000 5.000000 1.518514      NA
```

¿Qué resultado se obtiene al ejecutar estos códigos?

```
x1 = c(3, 4, 5, 12, 3.4, pi, log10(33))  
is.complex(x1)  
x1[exp(2)]  
x1[-2]  
x1[-c(1:3)]
```

```
y1 = c(FALSE, 5L, 6.3, 'abc123')  
typeof(y1)  
str(y1)  
is.double(y1)  
is.atomic(y1)  
y1[3]
```

Lista

Se trata de una colección de n datos que pueden ser de distinto tipo. Puede incluir vectores atómicos (u otras estructuras) dentro. Por ejemplo:

```
x2 = list(1L, 'a', TRUE)
```

```
x2
```

```
[[1]]
```

```
[1] 1
```

```
[[2]]
```

```
[1] "a"
```

```
[[3]]
```

```
[1] TRUE
```


(cont.)

```
x2 = list(1L, 'a', TRUE)
typeof(x2)
```

```
[1] "list"
```

```
str(x2)
```

```
List of 3
 $ : int 1
 $ : chr "a"
 $ : logi TRUE
```

```
is.atomic(x2)
```

```
[1] FALSE
```

```
is.list(x2)
```

```
[1] TRUE
```

(cont.)

```
x2 = list(1L, 'a', TRUE)
x2[2]
```

```
[[1]]
[1] "a"
```

```
x2[[2]]
```

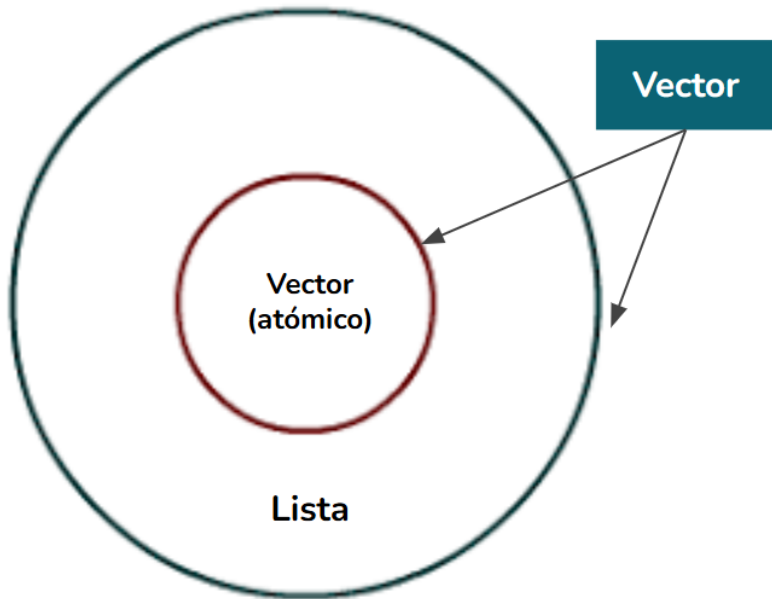
```
[1] "a"
```

¿Qué resultado se obtiene al ejecutar estos códigos?

```
x3 = list(1L, 2L, 4L)
is.integer(x3)
is.character(x2)
```

¿Qué resultado se obtiene al ejecutar estos códigos?

```
y2 = list(m = c(1,5,12), n = c(17,44,65))  
is.atomic(y2)  
is.list(y2)  
y2[[2]]  
y2[[2]][3]  
y2$n[3]
```



¿Qué resultado se obtiene al ejecutar estos códigos?

```
is.atomic(x1); is.list(x1); is.vector(x1)  
is.atomic(x2); is.list(x2); is.vector(x2)  
is.atomic(x3); is.list(x3); is.vector(x3)  
is.atomic(y1); is.list(y1); is.vector(y1)  
is.atomic(y2); is.list(y2); is.vector(y2)
```

Matriz

Se trata de una colección de $m \times n$ datos del mismo tipo (Integer, Double, Logical, Character, Complex).

De ser necesario, aplica coerción implícita. Por ejemplo:

```
x3 = matrix(c(7,3,5,8,6,6,-3,5,6,9),ncol=2)
```

```
x3
```

	[,1]	[,2]
[1,]	7	6
[2,]	3	-3
[3,]	5	5
[4,]	8	6
[5,]	6	9

```
typeof(x3)
```

```
[1] "double"
```

```
str(x3)
```

```
num [1:5, 1:2] 7 3 5 8 6 6 -3 5 6 9
```

```
x3[1,2]
```

```
[1] 6
```

```
x3[2,]
```

```
[1] 3 -3
```

```
x3[,2]
```

```
[1] 6 -3 5 6 9
```

```
x3[c(1,3),2]
```

```
[1] 6 5
```

¿Qué resultado se obtiene al ejecutar estos códigos?

```
y3 = matrix(c(7,TRUE,5,8,1,2),nrow=3)
typeof(y3)
str(y3)
is.double(y3)
is.vector(y3)
is.list(y3)
is.matrix(y3)
y3[3,1]
y3[2,]
y3[,2]
```


Data frame

Se trata de una colección de $m \times n$ datos de distinto tipo (en cada columna).

Es la estructura comúnmente utilizada para trabajar con conjuntos de datos reales. Por ejemplo:

```
var1 = c(4,5,3,5,3)
var2 = c('1','3','66','12','15')
x4 = data.frame(var1, var2)
x4
```

	var1	var2
1	4	1
2	5	3
3	3	66
4	5	12
5	3	15

¿Qué resultado se obtiene al ejecutar estos códigos?

```
typeof(x4)
str(x4)
is.double(x4)
is.vector(x4)
is.list(x4)
is.matrix(x4)
is.data.frame(x4)
x4[1,2]
x4[2,]
x4[,2]
x4$var1
```

Array

Es una estructura de dimensión no definida cuyos elementos son del mismo tipo.

```
(ar1 = array(c(1,3,4,10,10,20,2,1), dim = 8)); str(ar1)
```

```
[1]  1  3  4 10 10 20  2  1  
 num [1:8(1d)] 1 3 4 10 10 20 2 1
```

```
(ar2 = array(c(1,3,4,10,10,20,2,1), dim = c(8,1))); str(ar2)
```

```
      [,1]  
[1,]    1  
[2,]    3  
[3,]    4  
[4,]   10  
[5,]   10  
[6,]   20  
[7,]    2  
[8,]    1  
 num [1:8, 1] 1 3 4 10 10 20 2 1
```

(cont.)

```
(ar3 = array(c(1,3,4,10,10,20,2,1), dim = c(2,4))); str(ar3)
```

	[,1]	[,2]	[,3]	[,4]
[1,]	1	4	10	2
[2,]	3	10	20	1

num [1:2, 1:4] 1 3 4 10 10 20 2 1

(cont.)

```
(ar4 = array(c(1,3,4,10,10,20,2,1), dim = c(2,2,2)));str(ar4)
```

, , 1

	[,1]	[,2]
[1,]	1	4
[2,]	3	10

, , 2

	[,1]	[,2]
[1,]	10	2
[2,]	20	1

num [1:2, 1:2, 1:2] 1 3 4 10 10 20 2 1

(cont.)

```
(G1 = array(c(1,3,4,10,10,20,2,'1'),  
            dim = c(2,2,2),  
            dimnames = list(c("Fila1","Fila2"),  
                             c("Col 1","Col 2"),  
                             c("Capa1","Capa2"))))
```

, , Capa1

	Col 1	Col 2
Fila1	"1"	"4"
Fila2	"3"	"10"

, , Capa2

	Col 1	Col 2
Fila1	"10"	"2"
Fila2	"20"	"1"

Tensores

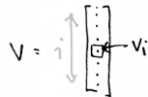
a number



a vector



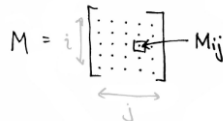
v_i



a matrix



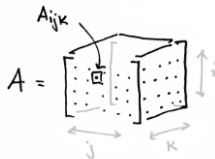
M_{ij}



a 3-tensor



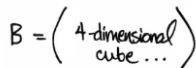
A_{ijk}



a 4-tensor



B_{ijkl}



\vdots

\vdots

\vdots

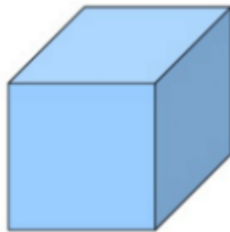
\vdots



1d-tensor



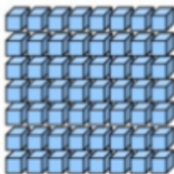
2d-tensor



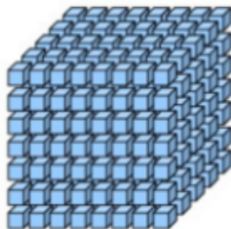
3d-tensor



4d-tensor



5d-tensor



6d-tensor

Funciones importantes para el manejo de vectores atómicos

```
(a = c(9,4,2,3,4))
```

```
[1] 9 4 2 3 4
```

```
(a1 = rep(4,3))
```

```
[1] 4 4 4
```

```
(a2 = seq(1,6))
```

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

```
(a3 = seq(1,6,0.5))
```

```
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0
```

```
(a4 = rev(a))
```

```
[1] 4 3 2 4 9
```

(cont.)

```
(a = c(9,4,2,3,4))
```

```
[1] 9 4 2 3 4
```

```
(a5 = sort(a))
```

```
[1] 2 3 4 4 9
```

```
(a6 = unique(a))
```

```
[1] 9 4 2 3
```

```
(a7 = letters)
```

```
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r"  
[20] "t" "u" "v" "w" "x" "y" "z"
```

```
which(letters=="c")
```

```
[1] 3
```

Funciones importantes para manejo de listas

```
b1 = list(x = 4L, y = TRUE, 5-3i)
names(b1)
```

```
[1] "x" "y" ""
```

```
unlist(b1)
```

```
      x      y
4+0i 1+0i 5-3i
```

```
sapply(b1,"as.numeric")
```

```
x y
4 1 5
```

(cont.)

```
b1 = list(x = 4L, y = TRUE, 5-3i)
lapply(b1,"as.numeric")
```

\$x

[1] 4

\$y

[1] 1

[[3]]

[1] 5

¿Qué resultado se obtiene al ejecutar estos códigos?

```
b2 = list(x = c("r","a","f"), y = TRUE, z = list(0,1))  
names(b2)  
unlist(b2)  
lapply(b2, "as.character")
```

Funciones importantes para manejo de matrices

```
c1 = matrix(c(5,6,3,4,3,3,1,1),  
            ncol = 4,  
            byrow = TRUE,  
            dimnames = list(c("f1","f2"),c("co1","co2","co3","co4")))
```

```
c1
```

	co1	co2	co3	co4
f1	5	6	3	4
f2	3	3	1	1

```
dim(c1)
```

```
[1] 2 4
```

```
rownames(c1)
```

```
[1] "f1" "f2"
```

```
colnames(c1)
```

```
[1] "co1" "co2" "co3" "co4"
```

```
t(c1)
```

	f1	f2
co1	5	3
co2	6	3
co3	3	1
co4	4	1

¿Qué resultado se obtiene al ejecutar estos códigos?

```
c1 = matrix(c(5,6,3,4,3,3,1,1),ncol = 4, byrow = TRUE)
c2 = matrix(c(3,4,3,4),nrow = 2)
cbind(c1,c2)
c3 = matrix(c(3,4,3,4),ncol = 4)
rbind(c1,c3)
```


Datos semiestructurados

- ▶ No son almacenados en tablas de bases de datos relacionales pero tiene etiquetas que permiten identificar los datos y sus características.
- ▶ Los formatos más comunes son XML y JSON



```
{
  "firstName": "Jonathan",
  "lastName": "Freeman",
  "loginCount": 4,
  "isWriter": true,
  "worksWith": ["Spantree Technology Group", "InfoWorld"],
  "pets": [
    {
      "name": "Lilly",
      "type": "Raccoon"
    }
  ]
}
```

```
{
  "name": "City Zoo",
  "animal":
    [
      {"type": "Reptile", "species": "crocodile"},
      {"type": "Bird", "species": "crowned pigeon"},
      {"type": "Mammal", "species": "giraffe"}
    ]
}

{
  "name": "Myrtle Edwards",
  "city": "Seattle",
  "phone":
    [
      {"type": "Home", "number": "508-555-0123"},
      {"type": "Work", "number": "617-555-0112"},
      {"type": "Cell", "number": "774-555-0121"}
    ]
}
```

Datos no estructurados

- ▶ Sin estructura interna clara
- ▶ Predominantes en la actualidad, pero difíciles de analizar
- ▶ No procesables directamente por algoritmos tradicionales
- ▶ Requieren estructuración previa para su análisis
- ▶ Comunes en redes sociales y plataformas digitales



Datos textuales

- ▶ Documentos (artículos, libros, revistas, informes)
- ▶ Comentarios en redes sociales (Facebook, Twitter, Reddit)
- ▶ Reseñas de productos (Amazon, TripAdvisor)
- ▶ Transcripciones de entrevistas y discursos

Imágenes

- ▶ Fotografías personales (perfiles, selfies)
- ▶ Imágenes satelitales y médicas (radiografías, tomografías)
- ▶ Señales de tránsito y carteles publicitarios
- ▶ Memes y contenido visual en redes sociales

Audio

- ▶ Llamadas telefónicas y mensajes de voz
- ▶ Podcasts y audiolibros
- ▶ Música en plataformas (Spotify, Apple Music)
- ▶ Reconocimiento de voz (Siri, Alexa, Google Assistant)

Video

- ▶ Streaming (YouTube, Netflix, Twitch)
- ▶ Videovigilancia (cámaras de seguridad, drones)
- ▶ Videos de redes sociales (TikTok, Instagram Reels)
- ▶ Clases virtuales y conferencias (Zoom, Teams, Meet)