
# Sesión 1

## Introducción a R como lenguaje de programación

Campamento de invierno EGOB | UC | 26 de julio, 2023

**José D. Conejeros** | jdconejeros@uc.cl

# Guía

1. Introducción a R
2. Lógica de programación
3. Estructura de datos en R
4. Operadores
5. Errores frecuentes
6. Manipulación de objetos

# ¿De qué se trata este taller?

El objetivo de este taller es aplicar de manera práctica a las principales herramientas que ofrece R para la lectura, procesamiento y visualización de datos cuantitativos. Con esto se espera tener una base troncal para la exploración e implementación de técnicas de análisis y modelamiento.

Una de las herramientas más útiles para realizar estas tareas es R, un lenguaje y entorno de programación abierto para el procesamiento y análisis de datos en un sinnúmero de disciplinas. La gran ventaja de esta herramienta, además de ser gratuito, es su flexibilidad y versatilidad, ya que con 19000 librerías disponibles se puede realizar una amplia gama de tareas con datos de distinta naturaleza https://cran.r-project.org/.

# Contenidos

1. Introducción a R como lenguaje de programación estadístico y RStudio como interfaz gráfica
2. Lectura, exploración y procesamiento de tablas de datos
3. Manipulación de tablas y objetos
4. Manipulación de variables: strings, dates, numeric y factors
5. Programación funcional: funciones, iteraciones y procesos
6. Visualización de datos con ggplot

# ¿Qué se espera al final de este taller?

Al terminar el taller los estudiantes sabrán:

a. Trabajar con vectores, matrices y listas

b. Importar y explorar tablas de datos en diferentes formatos

c. Manipular tablas de datos y variables

d. Implementar funciones y realizar procesos iterativos

e. Visualizar datos cuantitativos

# ¿Cómo se evalua este curso?

El curso tendrá una evaluación que consistirá en una tarea a realizar dentro del plazo de dos semanas. El objetivo de la tarea es poner en práctica todos los contenidos vistos en el taller de R. La evaluación será publicada el lunes 31/07/2023 y se deberá enviar por correo a más tardar el viernes 11/08/2023.

# Literatura del curso

El curso tendrá lecturas obligatorias y complementarias clase a clase.

Healy, K. (2018). Data visualization: a practical introduction. Princeton University Press. Versión en inglés: https://socviz.co/lookatdata.html

Wickham, H., & Grolemund, G. (2016). R for data science: import, tidy, transform, visualize, and model data. " O'Reilly Media, Inc.". Versión en inglés: https://r4ds.hadley.nz/

Wickham, H. (2019). Advanced r. CRC press. Versión en inglés: https://adv-r.hadley.nz/

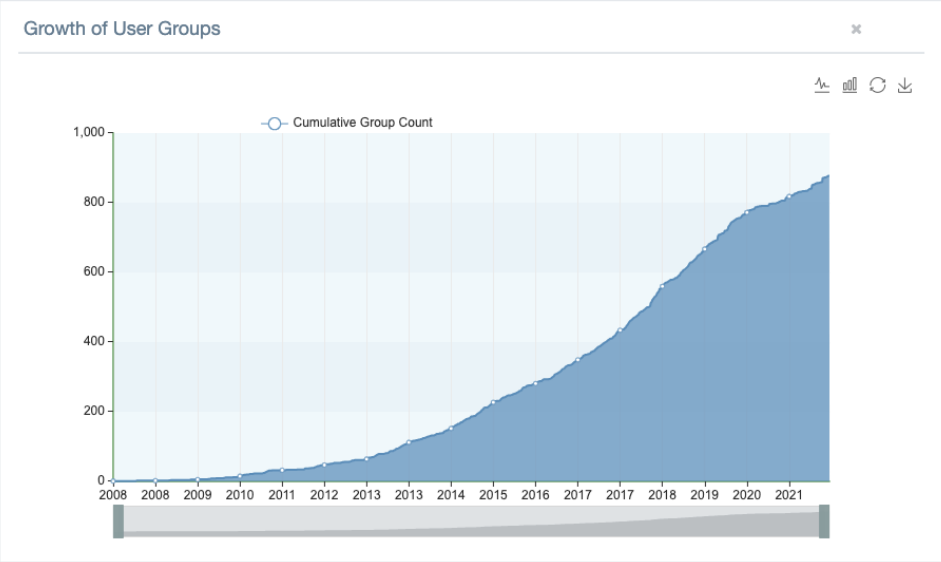
Whickham, H., Navarro, D., & Pedersen, T. L. ggplot2: Elegant Graphics for Data Analysis [Internet]. [cited 2021 Mar 17]. Versión en inglés: https://ggplot2-book.org/

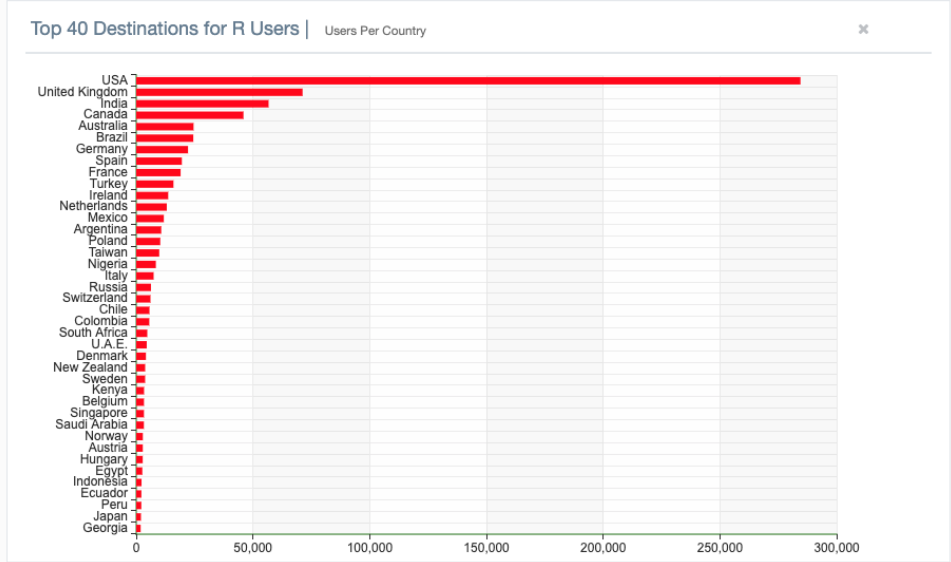
# ¿Qué es R?

R es un Ambiente y lenguaje de programación libre con un enfoque estadístico para el desarrollo de herramientas, métodos, cálculos y gráficos. Utiliza un lenguaje específico, anidado principalmente en las palabras comunes de técnicas estadísticas en ingles.

En R, un análisis estadístico se realiza en una serie de pasos, con resultados intermedios que se van almacenando para ser observados o analizados posteriormente.

# ¿Qué es R?




Fuente: https://benubah.github.io/r-community-explorer/rugs.html

# ¿Qué es R?

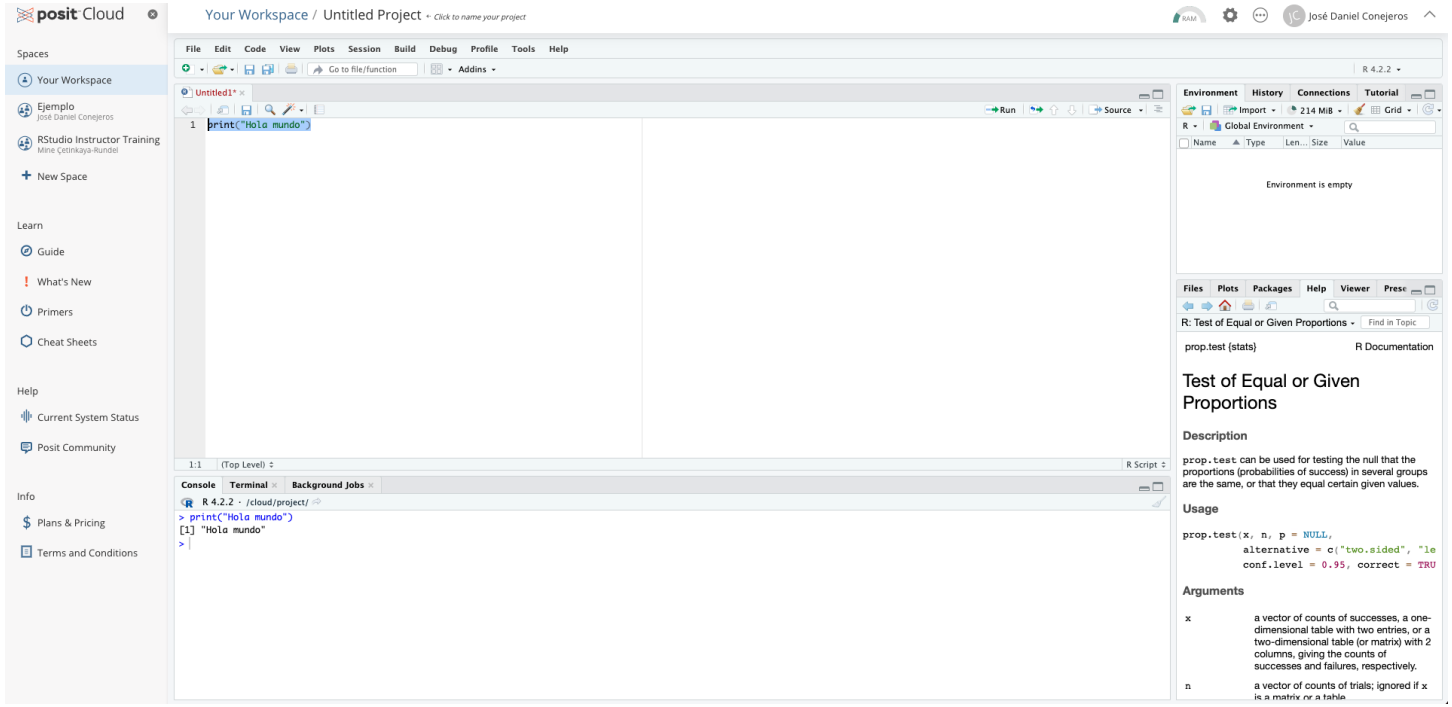
- Almacenamiento y manipulación efectiva de datos.
- Operadores para cálculo sobre variables indexadas, en particular, objetos.
- Una amplia, coherente e integrada colección de herramientas para análisis de datos.
- Ofrece un flujo completo e integrado para el análisis de datos.
- Posibilidades gráfcas para análisis de datos, que funcionan directamente sobre pantalla o para exportar.
- Lenguaje de programación de código abierto bien desarrollado, simple y efectivo.
- Amplia comunidad de desarrolladores.
- Interacción con otros lenguajes o softwares estadísticos.

# ¿Qué es RStudio?


Es un entorno de desarrollo integrado (IDE) para R.
Incluye varios elementos que facilitan las tareas durante el flujo de trabajo.

# RStudio Cloud

Posit Cloud es un servicio web que ofrece una experiencia basada en navegador similar a RStudio, el IDE estándar para usuarios y desarrolladores de R.




***Nota:*** Se debe crear una cuenta de usuario para acceder.

Pueden ver una guía con un click aquí

# Objetos en R

```
saludo ← "Hola mundo"
print(saludo)
```

```
[1] "Hola mundo"
```

```
# Esto es un vector
nombre  ← c("José", "Constanza")

edad    ← c(29, 24)

altura  ← c(1.67, 1.65)

matriz  ← cbind(nombre, edad, altura)
matriz
```

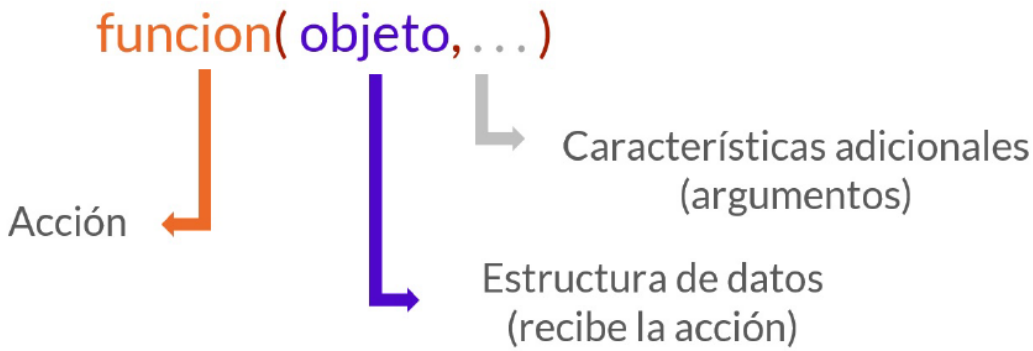
```
     nombre      edad altura
[1,] "José"      "29" "1.67"
[2,] "Constanza" "24" "1.65"
```

```
lista   ← c(matriz, edad, nombre); lista
```

```
 [1] "José"      "Constanza" "29"        "24"        "1.67"      "1.65"
```

# Lógica de ejecución

Ejecutan una acción sobre nuestros datos. Algunas requieren `inputs` (argumentos) que van dentro del paréntesis.



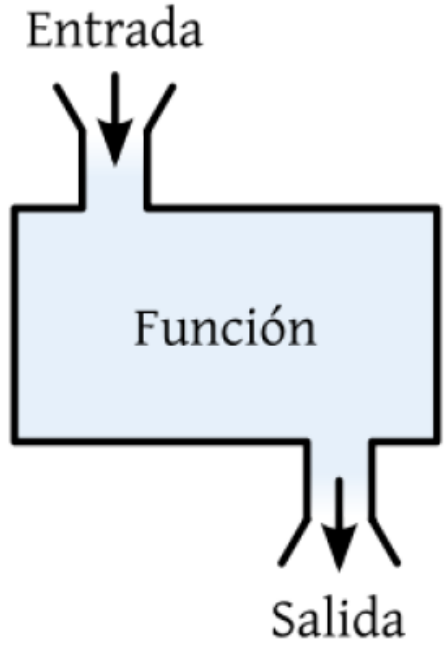

---

¿Qué hacer?
**Función**

¿A quién?
**Objeto**

# Funciones y argumentos

***Hacer que las cosas sucedan***




Algunas requieren inputs y otras no, los inputs (argumentos) van dentro del paréntesis

# Funciones y argumentos

```
?sum

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

Arguments
 ...          numeric or complex or logical vectors.

na.rm        logical. Should missing values (including NaN) be removed?
```

```
edad
```

```
[1] 29 24
```

```
sum(edad)
```

```
[1] 53
```

**Objetos** $\longrightarrow$ **Valores** $\longrightarrow$ **Funciones**

# Resolver dudas de código



# Resolver dudas de código

## Podemos usar inteligencias artificiales

- YOU: You.com is a search engine that summarizes the best parts of the internet for you, with private ads and with privacy options. Our AI will help you find the most relevant results from the web and apps that you can sort and prioritize for optimum search experience. We believe that you should control your search: your privacy, results, and time.
- PHIND: Phind es un motor de búsqueda que simplemente les dice a los usuarios cuál es la respuesta. Optimizado para desarrolladores y preguntas técnicas, Phind responde instantáneamente a las preguntas con explicaciones detalladas y fragmentos de código relevantes de la web. Phind funciona con grandes modelos de lenguaje de IA. A diferencia de otros asistentes de IA, Phind obtiene información de Internet y siempre está actualizada. Es lo suficientemente inteligente como para generar respuestas basadas en información de múltiples fuentes. Con Phind, encontrar información es tan sencillo e informativo como hablar con un amigo.
- CHATGPT

# Librerías

Para hacer efectiva la actualización del software, y basado en el foco académico, R utiliza librerías como conjunto de funciones, datos y documentación que denominaremos paquetes o packages.

Cuando iniciamos R se carga solo un conjunto de funciones, datos y documentación que se conoce como R Base.

**Sin embargo, paquetes y librerías no son lo mismo**. La librería es código reutilizable que se puede intercambiar y reutilizar en diferentes programas para diferentes propósitos.


**Hay 19514 paquetes disponibles a la fecha.**

# Tipos de datos

R es capaz de manejar una variedad de tipos de datos, que se almacenan en diferentes estructuras de datos.

**Tipos de datos**:

- Double: Números reales `c(1,0.5)`
- Integer: Números enteros `c(-1L,-2L,5L)`
- Character: Cadenas de texto `c("texto1","texto2")`
- Logical: Valores booleanos `c(TRUE,FALSE)`
- Complex: Números complejos `c(1+0i, 2+4i)`
- Raw: Datos sin procesar

# Coerción

La coerción es una característica de los lenguajes de programación que permite, implícita o explícitamente, convertir un objeto de un tipo de datos en otro.

Cuando pedimos a R ejecutar una operación, intentará coercionar de manera **implícita**, sin avisarnos, los datos de su tipo original al tipo correcto que permita realizarla. En ocaciones se realizará de manera correcta en otras es posible obtengamos error.

No todos los tipos de datos pueden ser transformados a los demás. La coerción se realiza de los tipos de datos más restrictivos a los más flexibles.

`logical` → `integer` → `numeric` → `character`

# Coerción

Podemos hacer explícito lo implícito mediante las funciones `as ...`

- `as.interger()` → entero

```
as.integer("true")
```

```
[1] NA
```

- `as.numeric()` → numérico

```
as.numeric("cinco")
```

```
[1] NA
```

- `as.character()` → cadena de texto

```
as.character(5)
```

```
[1] "5"
```

# Coerción

Podemos hacer explícito lo implícito mediante las funciones `as ...`

- `as.factor()` → factor

```
as.factor("texto")
```

```
[1] texto
Levels: texto
```

- `as.logical()` → lógico

```
as.logical(1)
```

```
[1] TRUE
```

- `as.null()` → NULL

```
as.null("texto")
```

```
NULL
```

# Estructuras de datos en R

En R los distintos tipos de datos definidos anteriormente se organizan en estructuras. Las estructuras de datos son objetos que contienen datos.

- Vector: `c()`
- Matriz: `matrix()`
- Array: `data.frame()`
- Data Frame: `list()`
- Lista: `factor()`
- Factor: `as.Date()`
- Fechas: `ts()`
- Series de tiempo:
- Otros...

Cuando trabajamos con R, lo que estamos haciendo es manipular estas estructuras.

# Estructuras de datos en R: vectores

- Es la estructura de datos más simple y representa una secuencia de datos del mismo tipo.

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

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

```
c(1:8)
```

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

- Matemáticamente es una matriz de una dimensión.
- En términos estadísticos es lo que conocemos como variable. **¿Puede ser una variable aleatoria?**

# Estructuras de datos en R: vectores

Un vector puede ser de diferentes tipos de datos. En el caso de los caracteres se debe considerar comillas `" "`, de lo contrario R pensará que es otro objeto.

```
c("a", "b", "c")
```

```
[1] "a" "b" "c"
```

Es posible mezclar datos de diferentes tipos. Luego, R hará una conversión implícita en el tipo de datos más general.

```
c(1, 2, 3, 4, "a")
```

```
[1] "1" "2" "3" "4" "a"
```

# Estructuras de datos en R: matrices y array's

Una **matriz** es una representación de datos de dos dimensiones. Para crear una matriz usamos:

```
matrix(1:8, nrow=2, ncol=4, byrow=FALSE)
```

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

Un **array** es la generalización de una matriz al caso multidimensional. Su definición general es de la forma:

```
array(1:8, c(2,4))
```

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

# Estructuras de datos en R: marcos de datos o "data frames"

Conjunto de vectores de diversos tipos. Cada vector tiene el mismo número de elementos.

```
data.frame(
  edad=c(15,18,22),
  genero=c("Masculino", "Femenino", "No binario"),
  variable_ficticia=c(1, 2, "a"))
```

```
  edad     genero variable_ficticia
1   15  Masculino                 1
2   18   Femenino                 2
3   22 No binario                 a
```

La tabla de variables individuales es la estructura por excelencia en estadística. En R, esta noción se expresa mediante un **marco de datos**. Conceptualmente, es una matriz en la que cada línea corresponde a un individuo o unidad de análisis y cada columna corresponde a una variable medida en los individuos (por eso es un objeto bidimensional). Cada columna representa una sola variable, que debe ser del mismo tipo en todos los individuos.

# Estructuras de datos en R: listas

Una lista es una colección ordenada de objetos de cualquier naturaleza.

```
list(data.frame(edad=c(15,18,22),
                genero=c("Masculino",
                         "Femenino",
                         "No binario"),
                variable_ficticia=c(1,
                                    2,
                                    "a")),
  array(1:8, c(2,4)),
  matrix(1:8, nrow=2, ncol=4,
         byrow=FALSE),
  c(1, 2, 3, 4, "a"))
```

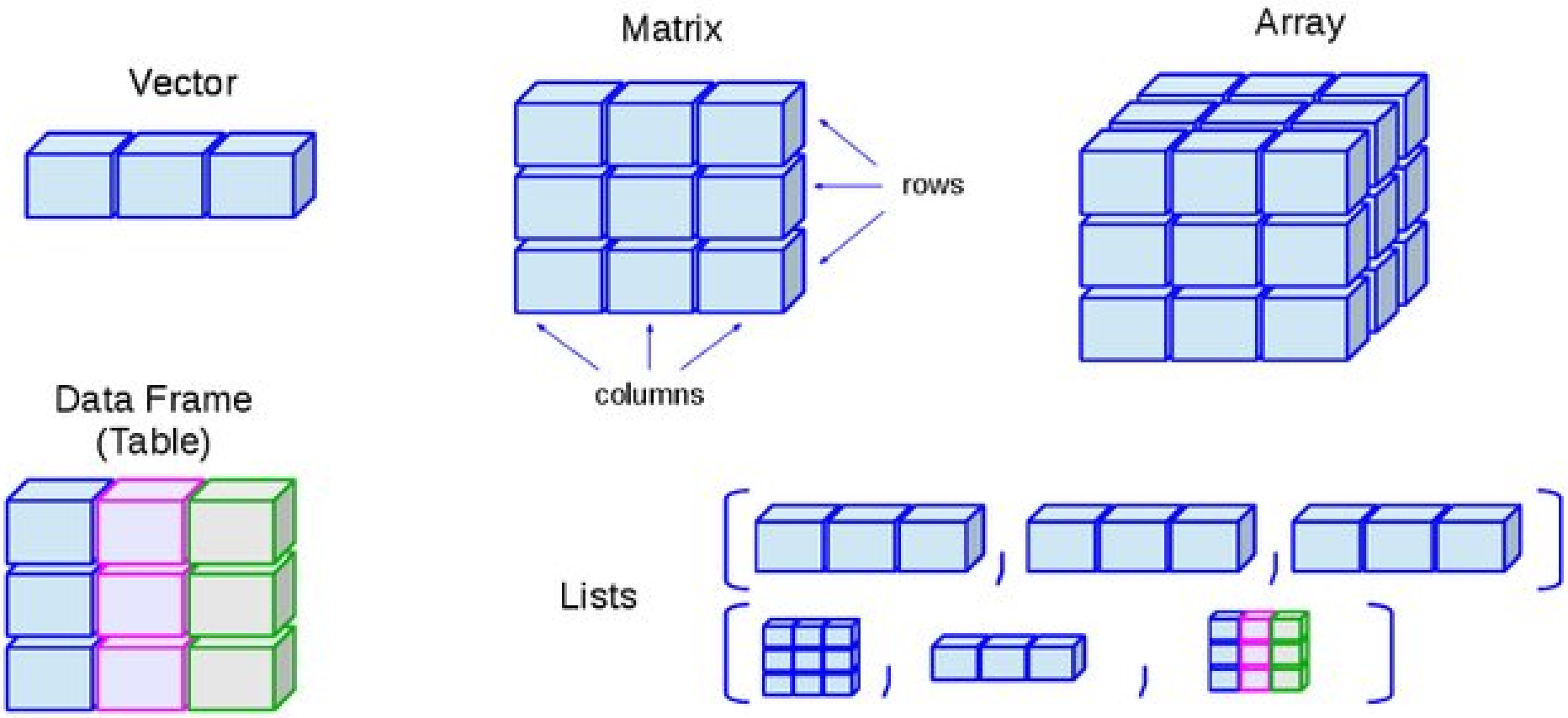
```
[[1]]
  edad     genero variable_ficticia
1   15  Masculino                 1
2   18   Femenino                 2
3   22 No binario                 a

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

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

[[4]]
[1] "1" "2" "3" "4" "a"
```

# Estructuras de datos: resumen



# Operadores

### Lógicos

- `&` → Y (Intersección)
- `|` → O (Unión)
- `!` → Negación
- `~` → Negación

### Aritméticos:

- `+` → Suma
- `-` → Resta
- `*` → Multiplicación
- `/` → División
- `^` → Potencia

### Relacionales

- `>` → Mayor que
- `<` → Menor que
- `>=` → Mayor o igual que
- `<=` → Menor o igual que
- `==` → Igual (doble igual)
- `!=` → No igual
- `~=` → No igual

Otros: operaciones matriciales, iteraciones, funciones, etc.

# Números especiales

- `inf`, `-inf` para representar el in□nito positivo y el infinito negativo, respectivamente.
- `NaN` es la abreviatura de "Not a Number" aparece cuando R no puede calcular un valor para un objeto.
- `NA` es la abreviatura de "Not Available" representa un valor ausente o perdido (missing). Problema muy común en el análisis de datos. En general, si nuestro cálculo implica un valor faltante, entonces también faltarán los resultados.

Ejemplo:

```
numeros ← c(1, NA, 5, 6, 7)
is.na(numeros)
```

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

```
mean(numeros)
```

```
[1] NA
```

> R tiene varias maneras de trabajar con el missing value

# Algunas funciones útiles

- `c()` → Concatena objetos (variables, textos, números, etc.)
- `help()` → Ayuda respecto de alguna función.
- `library()` → Carga de librerías.
- `ls()` → Lista de objetos
- `rm()` → Eliminar objetos
- `mean()` → Promedio aritmético de los valores de un vector
- `getwd()`, `setwd()` → Muestra y establece el espacio de trabajo, respectivamente.
- `dir.create()` → Crea un directorio dentro del espacio de trabajo.
- `typeof()` → Muestra la naturaleza de un objeto
- `as.numeric()` → Convierte un objeto a tipo numérico
- `matrix()`, `array()` → crea una matriz, una tabla multidimensional
- `list()` → crea una lista (colección de diferentes estructuras)
- `data.frame()` → crea una tabla de variables individuales
- `rbinom()` → Genera valores aleatorios de una distribución binomial.
- `set.seed()` → Establece una semilla para inicializar un generador de números aleatorios.

# Errores frecuentes en el uso de R

La mayoría de errores corresponden a problemas de tipeo:

- Escribimos mal el nombre de una función u objeto
- Falta cerrar un paréntesis
- Falta una coma

En caso de que falte un paréntesis o una coma, el editor de RStudio nos lo advertirá.

A veces se generan problemas porque olvidamos correr una línea de código o porque sobreescribimos un objeto. En esos casos, lo mejor es reiniciar R y volver a ejecutar el código desde el principio.

> Equivocarse es normal y esperable. No te preocupes, respira y puedes googlear el error si no sabes como resolverlo (o no lo entiendes). Es muy probable que a alguien más ya le haya sucedido y otra persona lo ha solucionado. ¡Eso es lo bueno de tener una amplia comunidad!

*Nota*: también puedes usar chat GPT. Aunque es útil para entender errores más que resolverlos. Más información aquí: https://openai.com/blog/chatgpt

# Identifica los errores

```
install.packages(ggplot2)
```

```
librari(ggplot)
```

```
c(1.5, 5, 7, 2,)
```

```
nombres ← c(Manuel, "Inés", 'ana")
```

```
edad ← c(34, 67, 45, 55, 2)
sum(Edad)
maen(edad)
```

```
install.packages("ggplot2")
```

```
library(ggplot)
```

```
c(1.5, 5, 7, 2)
```

```
nombres ← c("Manuel", "Inés", "ana")
```

```
edad ← c(34, 67, 45, 55, 2)
sum(edad)
mean(edad)
```

# Sintaxis en R

La sintaxis define la forma correcta de escribir las sentencias y los datos de cualquier programa, especificando el orden y disposición adecuada. R permite a sus desarrolladores especficar su propia sintaxis. Como resultado, existe una gran variedad de sintaxis igualmente válidas.

- **Sintaxis de R base:** Se caracteriza por el uso de `$` y se asocia a la creación de subsetting con `[ ]`. La lógica de programación es de adentro hacia afuera. Casi todas las funciones de R aceptarán elementos que se entreguen con esta sintaxis.

- **Sintaxis de fórmula:** La sintaxis de fórmula se basa en el uso de `~` (virgulilla). Se usa para modelamiento, gráficos y otras funciones.

- **Sintaxis del Tidyverse:** La sintaxis del Tidyverse se ha popularizado en los últimos años porque permite leer y programar de forma más natural (izquierda a derecha) y tiene un ecosistema de paquetes que trabajan juntos y facilitan el trabajo. Uno de sus elementos fundamentales es "el pipe" `%>%`.

> La mayoría de las personas usan una combinación de estas herramientas, según lo que sea más razonable dado un objetivo de análisis.

# Operadores de acceso o selección

En la sintaxis de R Base, hay tres operadores que pueden ser usados para extraer subconjuntos de objetos:

- `[]`: siempre devuelve un objeto de la misma clase que el original, y puede ser usado para seleccionar múltiples elementos de un objeto.
- `[[]]`: es usado para extraer elementos de una lista o de un data frame, y devuelve elementos simplificados.
- `$`: es usado para extraer elementos de una lista o un data frame indicando su nombre.

# Manipulación de vectores

- **Selección por posición**
  - `vector[i]`: entregará el valor en la posición i
  - `vector[-i]`: entregará todos los valroes excepto la posición i

```
vector ← c("A", "B", "C", "D", "E")
vector[2]
```

```
[1] "B"
```

```
vector[-2]
```

```
[1] "A" "C" "D" "E"
```

```
vector[-c(2, 4, 5)]
```

```
[1] "A" "C"
```

- **Selección por comparación**: selecciona el elemento cuya condición establecida por el operador de comparación es TRUE.

```
vector
```

```
[1] "A" "B" "C" "D" "E"
```

```
vector[c(TRUE, TRUE, FALSE, FALSE, TRUE)]
```

```
[1] "A" "B" "E"
```

```
numeros ← c(2,3,4,5,10,100)
condicion ← numeros > 5
condicion
```

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

```
numeros[condicion]
```

# Manipulación de listas

En las listas, se pueden extraer subconjuntos usando los tres operadores que mencionamos, cada uno con un propósito diferente.

```
lista ← list(char = "a", vector = c(3,4,5), df = data.frame(id = 1:5, vector))

lista$vector
```

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

```
lista[2]
```

```
$vector
[1] 3 4 5
```

```
lista[["vector"]]
```

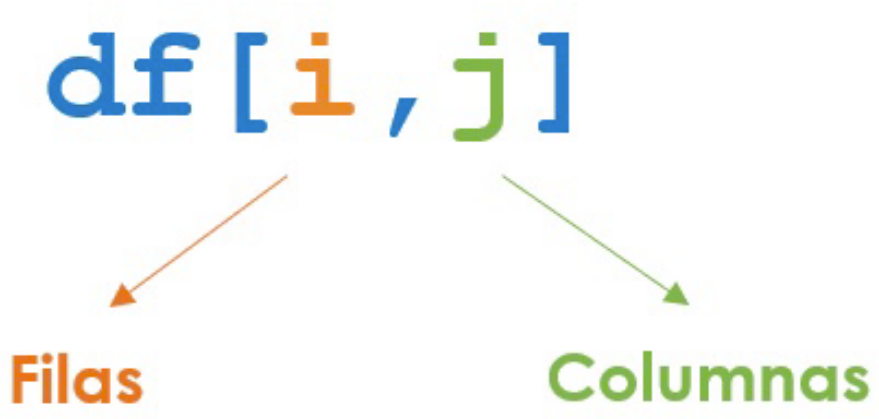
```
[1] 3 4 5
```

```
lista[[2]]
```

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

# Manipulación de marcos de datos

Recordemos que los marcos de datos son objetos bidimensionales. Por lo que operan bajo la lógica:




```
head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
```

# Manipulación de marcos de datos

df[ , 2]

```
mtcars[,2]
```

```
[1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8
4 4 4 8 6 8 4
```

df[2, ]

```
mtcars[2,]
```

```
              mpg cyl disp  hp drat    wt  qsec
Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02
```

df[2, 2]

```
mtcars[2,2]
```

```
[1] 6
```

# Manipulación de marcos de datos

Por otra parte, los objetos se pueden conservar encadenados a través de otro objeto con el operador `$`:

```
mtcars$cyl # Number of cylinders
```

```
 [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
```

Por otra parte, la función `attach()` reconoce cada variable del data frame como un objeto independiente.

# Importar y exportar

Temas a considerar:

- Tipo de archivo
- Título en la primera □la
- Separador
- Tipo de missing
- Decimales
- Etiquetas
- Fechas
- Otros...

**Importar datos es re-formartear una fuente de datos externa a R para que sea legible por el lenguaje**.

Para eso uno debe conocer el formato de datos que quiere importar: `.dta`, `.xlsx`, `.csv`, `.txt`, `.rds`, `.RData`, `.xpt`, `.mat`, `.mtp`, etc...

# Importar y exportar

A pesa de que el código para importar varía según formato podemos generalizarlo de la forma `read.formato()`. Por ejemplo:

```
read.table(file = "ruta/nombrearchivo.txt", header = TRUE, sep="nt", dec = ".", row.name
```

- `file=path/to/file` → Ubicación y nombre del archivo a leer.
- `header = TRUE` → Si se entregan los nombres de las variables en la primera línea.
- `sep="\t"` → Los valores en cada línea están separados por este caracter ("\t"=tabulación; "" = blanco; "," = , ; etc.).
- `dec="."` → Separador decimal para números ("." ó ",").
- `row.names = 1` → Si la primera columna entrega el nombre de los individuos.

Para saber más de esta función pueden escribir en su consola de R para revisar la documentación:

```
?read.table # o
help(read.table)
```

# Funciones para importar datos

- `read.table()`: Lee conjuntos de datos presentados como tablas, como suele ser el caso de Estadística.
- `read.csv()`: Lee archivos delimitados por coma.
- `read.csv2()`: Lee archivos delimitados por punto y coma (común en países donde se utiliza , como separador decimal)
- `read.delim()`: Lee archivos con cualquier delimitador
- `read.delim2()`: Igual que el anterior, pero se usa cuando los números tienen comas como decimales en lugar de puntos.
- `read.ftable()`: Lee tablas de contingencia (frecuencias).

Hay otras librerías que permiten importar otros formatos de datos. Por ejemplo, el paquete `readxl` tiene una función que facilita la importación de tablas de excel: `readxl::read_excel()`

**También está la manera manual desde las opciones de R**

> File > Import Dataset > Form...

# Exploración

Lo primero que uno realiza en cual proyecto de análisis de datos es explorar sus datos:

- Vista panorámica de los datos: columnas, filas, estructura
- Vista previa de la base de datos
- Exploración de etiquetas
- Estadísticos descriptivos
- Valores fuera de rango o missing
- Imputaciones de datos

# Exploración

Algunas funciones útiles para la exploración de datos son:

- `head(df,k)`: Muestra los primeros k registros.

```
head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

- `dim(df)`: Filas y columnas de un objeto.

```
dim(mtcars)
```

```
[1] 32 11
```

# Exploración

- `tail(df,k)`: Muestra los últimos k registros.

```
tail(mtcars)
```

```
                mpg cyl  disp  hp drat    wt qsec vs am gear carb
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
```

- `length(df)`: Número de objetos dentro del objeto df.

```
length(mtcars)
```

```
[1] 11
```

# Exploración

Algunas funciones útiles para la exploración de datos son:

- `str(df)`: Estructura de la base de datos df.

```
str(mtcars)
```

```
'data.frame':    32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

- `class(df)`: Naturaleza o clase del objeto df.

```
class(mtcars)
```

# Exploración

Algunas funciones útiles para la exploración de datos son:

- `names(df)`: Nombres del objeto df.

```
names(mtcars)
```

```
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
[11] "carb"
```

**En el taller puede encontrar más funciones**

# Exportar bases de datos

**Exportar datos es darle un formarto a una fuente interna R para que sea legible por otro lenguaje o procesador**.

Para eso uno debe conocer el formato de datos que quiere importar: `.dta`, `.xlsx`, `.csv`, `.txt`, `.rds`, `.RData`, `.xpt`, `.mat`, `.mtp`, etc...

A pesa de que el código para importar varía según formato podemos generalizarlo de la forma `write.formato()`. Por ejemplo:

```
write.table(x=data, file = "ruta/nombrearchivo.txt", sep=" ", dec = ".", row.names = TRU
```

- `x`: corresponde al objeto de R (marco de datos) que queremos exportar
- `file=path/to/file` → Ubicación y nombre del archivo a leer.
- `sep="\t"` → Los valores en cada línea están separados por este caracter ("\t"=tabulación; "" = blanco; "," = , ; etc.).
- `dec="."` → Separador decimal para números ("." ó ",").
- `row.names = TRUE` → Si devuelve el nombre de las filas.

Para saber más de esta función pueden escribir en su consola de R para revisar la documentación:

```
?write.table; help(write.table)
```

# Funciones para exportar datos

- `write.table()`: Exporta conjuntos de datos presentados como tablas, como suele ser el caso de Estadística.
- `write.csv()`: Exporta archivos delimitados por coma.
- `write.csv2()`: Exporta archivos delimitados por punto y coma (común en países donde se utiliza , como separador decimal)
- `write.delim()`: Exporta archivos con cualquier delimitador
- `write.ftable()`: Exporta tablas de contingencia (frecuencias).

Hay otras librerías que permiten importar otros formatos de datos. Por ejemplo, el paquete `writexl` tiene una función que facilita la exportación a tablas de excel: `writexl::write_xlsx()`


# Clase 1

## Introducción a R

26 de julio, 2023

**José D. Conejeros** | jdconejeros@uc.cl | JDConejeros