

# Introducción a R y Tidyverse I

## Sesión 01

Laboratorio de Innovación en Salud

2022-09-29

 @healthinnovation  
 @innovalab\_imt  
 innovalab.info

# Acerca del curso

Este curso busca introducir al estudiante en el uso de R y el uso del metapaqute **tidyverse** para el manejo inicial de datos durante un proceso de investigación.

Al finalizar el curso de manera satisfactoria, el participante podrá:

- Importar conjuntos de datos de los principales formatos que se trabajan en proyectos de investigación.
- Dar formato inicial al conjunto de datos mediante reconocimiento de variables y modificaciones en las mismas.
- Ejecutar modificaciones en variables pre-existentes y nuevas variables que se introduzcan.
- Trabajar con múltiples bases de datos y unificarlas de acuerdo al objetivo de investigación planteado.

# Contenidos

- Introducción a ciencia de datos
- Manejo y reconocimiento inicial de Rstudio
- Creación y manejo de estructura de proyectos en rstudio
- Creación de vectores
- Uso de funciones y argumentos
- Instalación y manejo de paquetes
- Introducción a tidyverse

# Exploración competencial

# Introducción a ciencia de datos

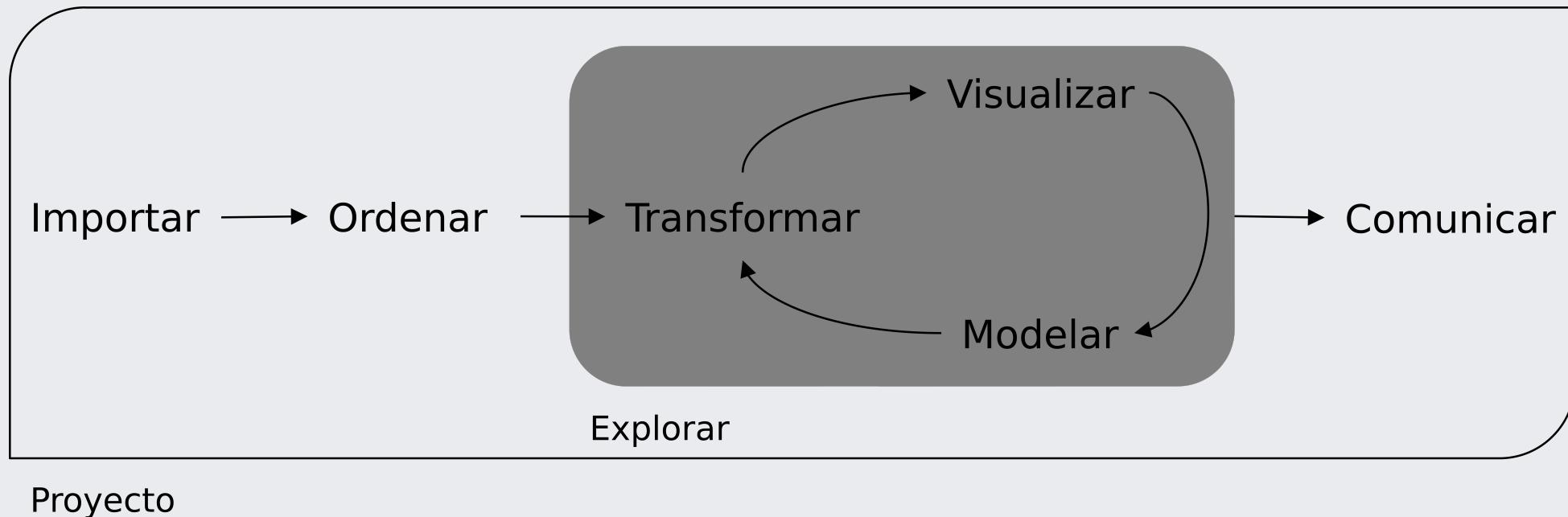
La Ciencia de Datos es una fusión entre múltiples disciplinas, incluyendo matemáticas, estadística, informática, y tecnología de la información.

La Ciencia de Datos permite extraer información relevante de los datos.

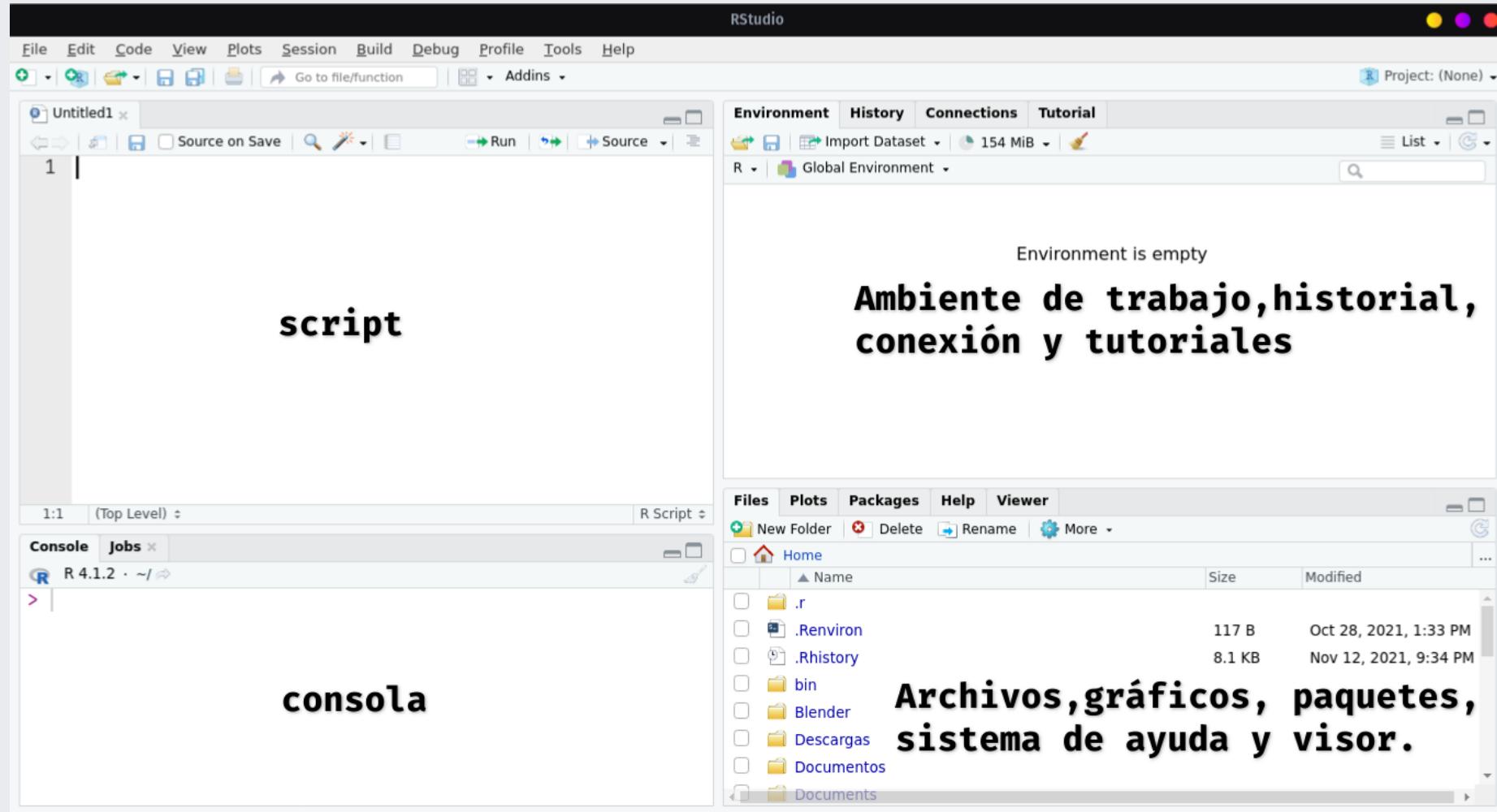


# ¿Por qué usar R para Ciencia de Datos?

R cuenta con las herramientas necesarias (entorno, librerías, y funciones) para desarrollar proyectos de Ciencia de Datos.



# Reconocimiento Rstudio



**script**

**Environment** History Connections Tutorial

Environment is empty

**Ambiente de trabajo, historial, conexión y tutoriales**

**consola**

**Files Plots Packages Help Viewer**

New Folder Delete Rename More

Home

Name	Size	Modified
.r	117 B	Oct 28, 2021, 1:33 PM
.Renvironment	8.1 KB	Nov 12, 2021, 9:34 PM
.Rhistory		
bin		
Blender		
Descargas		
Documentos		
Documents		

**Archivos, gráficos, paquetes, sistema de ayuda y visor.**

# Reconocimiento Rstudio

**SCRIPT**

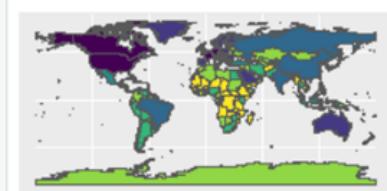
```
1  ### MI PRIMER SCRIPT
2
3  print('Hola mundo spatial')
4
5  library(ggplot2)
6  library(sf)
7  library(rnaturalearth)
8  vignette("rnaturalearth")
9
10 world <- ne_countries(scale = "medium", returnclass = "sf")
11 class(world)
12 |
13 ggplot(world)+ 
14   geom_sf(aes(fill = economy))+
15   scale_fill_viridis_d()
16
12:1 # MI PRIMER SCRIPT
```

**CONSOLA**

```
> ### MI PRIMER SCRIPT
> print('Hola mundo spatial')
[1] "Hola mundo spatial"
>
```

**ENTORNO, HISTORIAL, CONEXIONES**

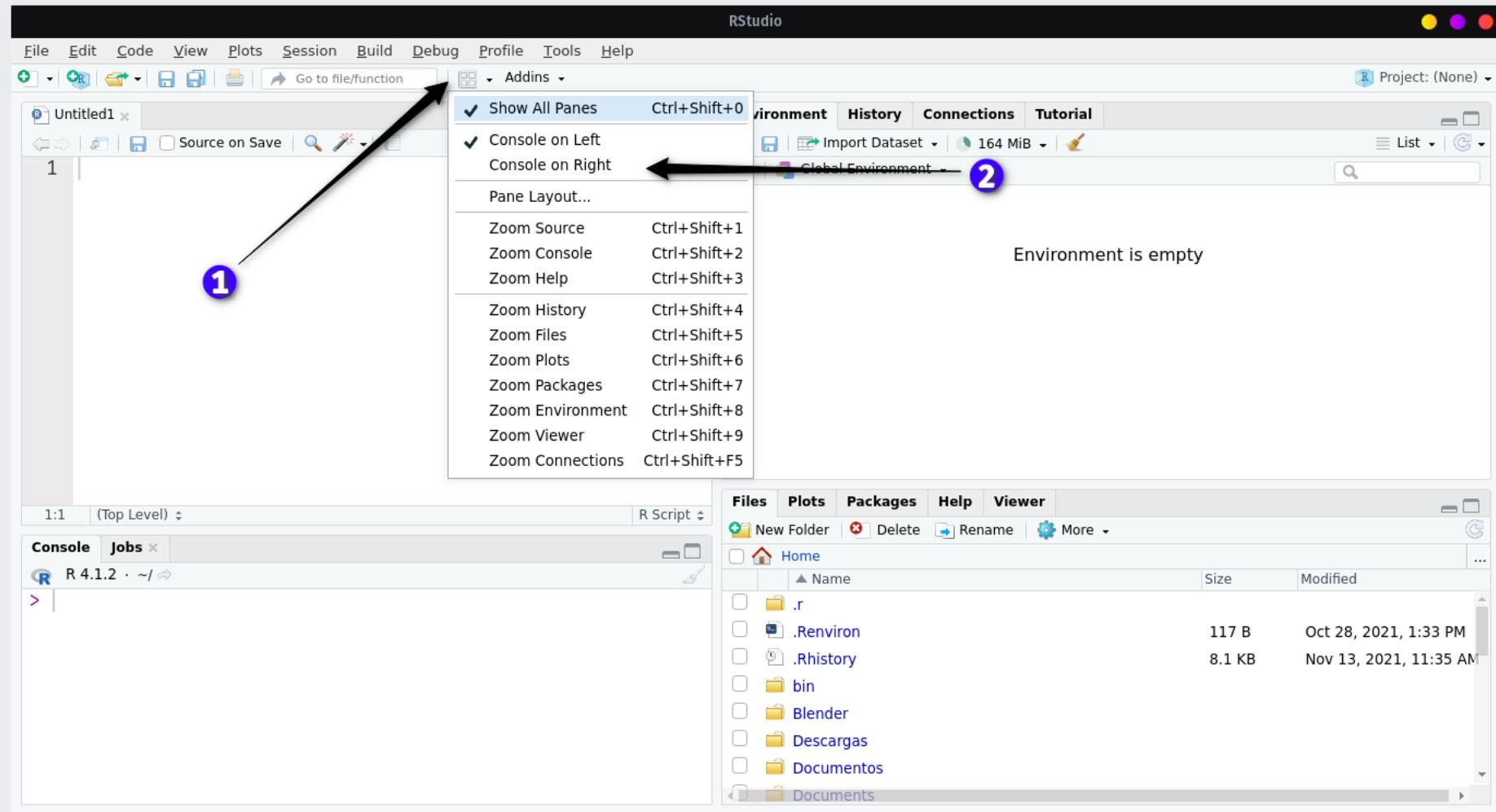
**GRÁFICOS, ARCHIVOS, SISTEMA DE AYUDA, VISUALIZADOR INTERACTIVO**



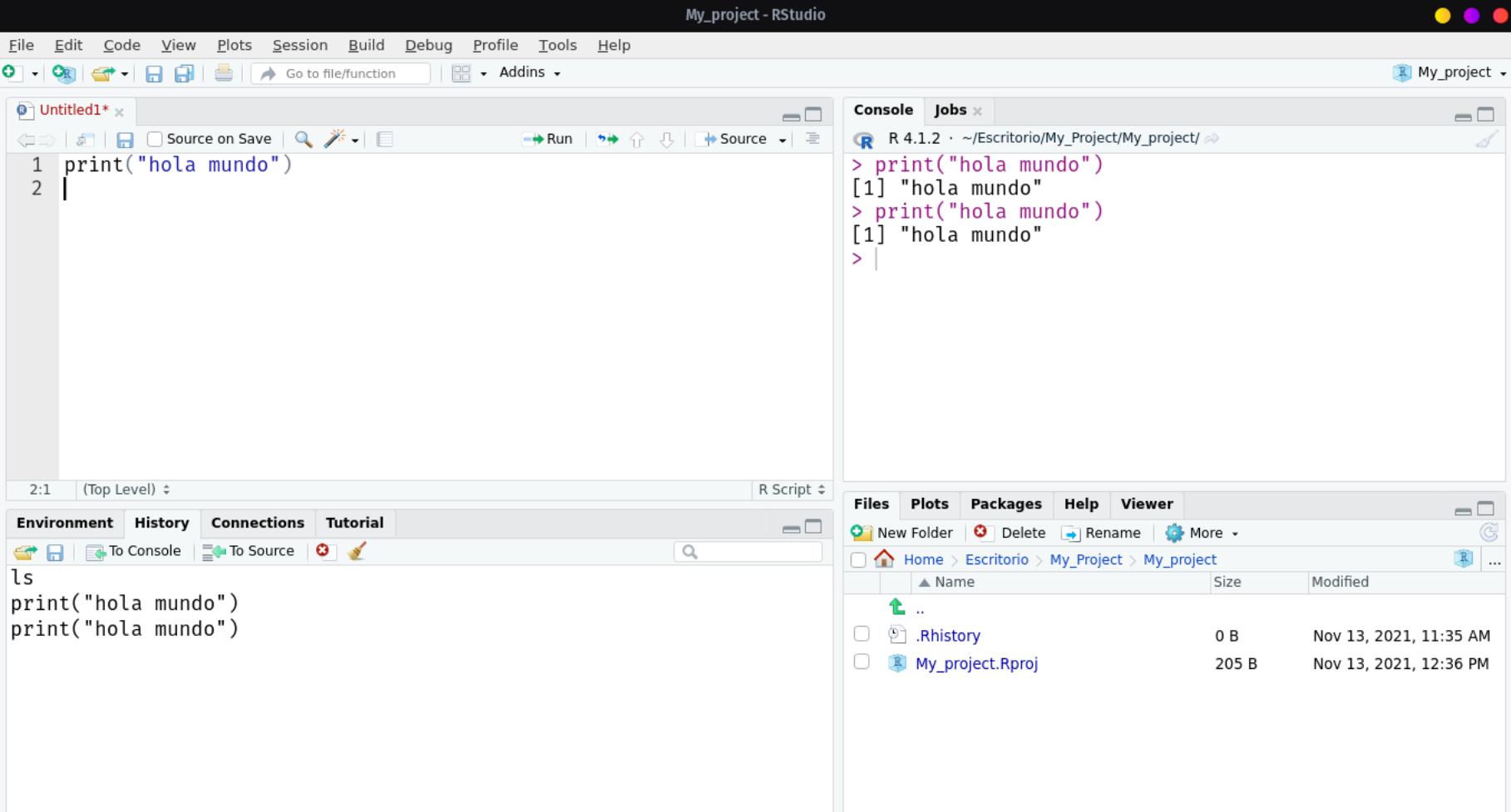
economy

- 1. Developed region: G7
- 2. Developed region: nonG7
- 3. Emerging region: BRIC
- 4. Emerging region: MIFT
- 5. Emerging region: G20
- 6. Developing region
- 7. Least developed region

# Mover paneles



# Mover paneles



The screenshot shows the RStudio interface with the following components:

- Code Editor (Left):** An untitled R script with the following code:

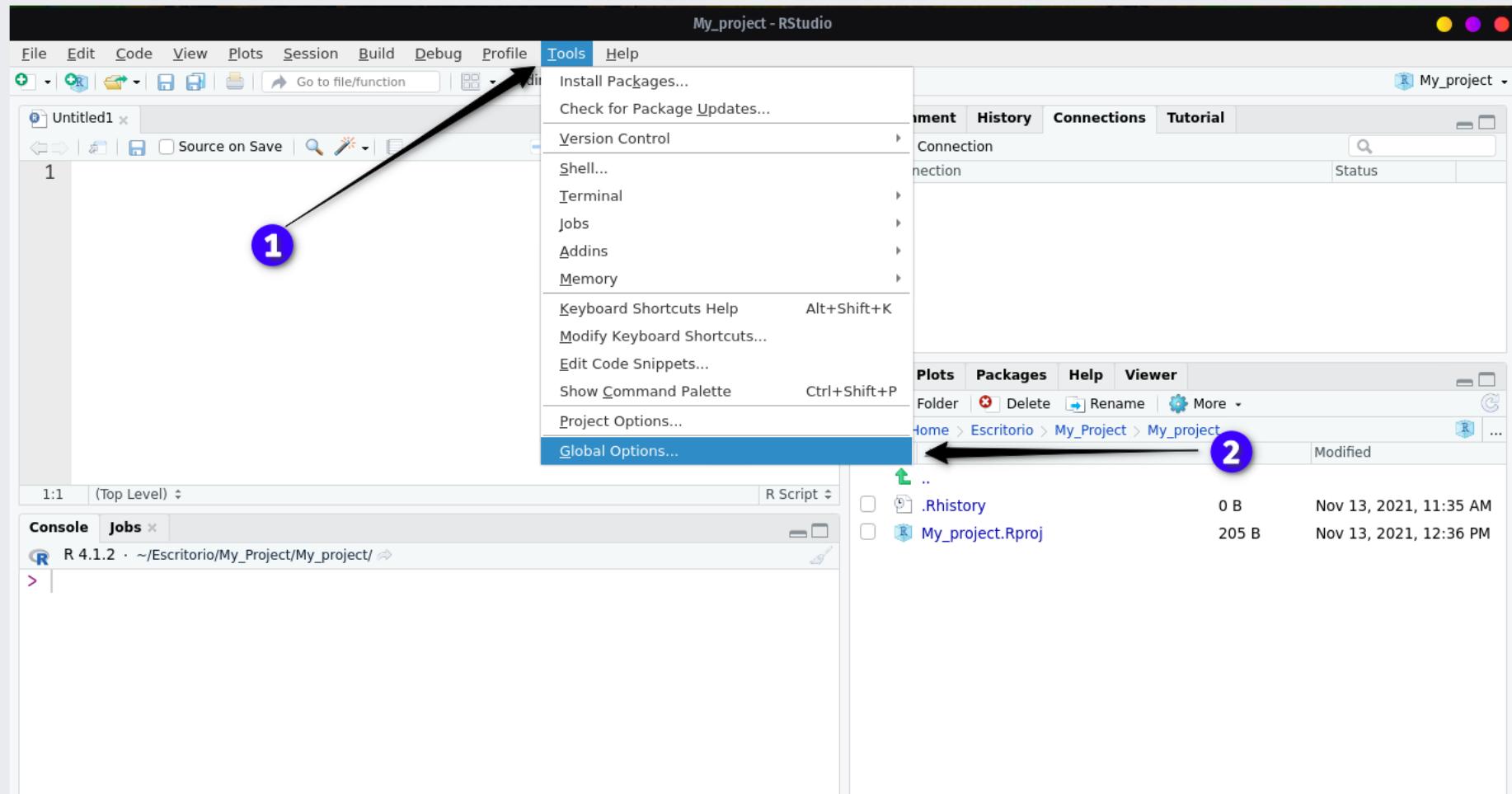
```
1 print("hola mundo")
2 |
```
- Console (Top Right):** Displays the output of the printed statements:

```
> print("hola mundo")
[1] "hola mundo"
> print("hola mundo")
[1] "hola mundo"
>
```
- Environment (Bottom Left):** Shows the current workspace with the following objects:

```
ls
print("hola mundo")
print("hola mundo")
```
- File Browser (Bottom Right):** Shows the project structure and files:

Name	Size	Modified
..		
.Rhistory	0 B	Nov 13, 2021, 11:35 AM
My_project.Rproj	205 B	Nov 13, 2021, 12:36 PM

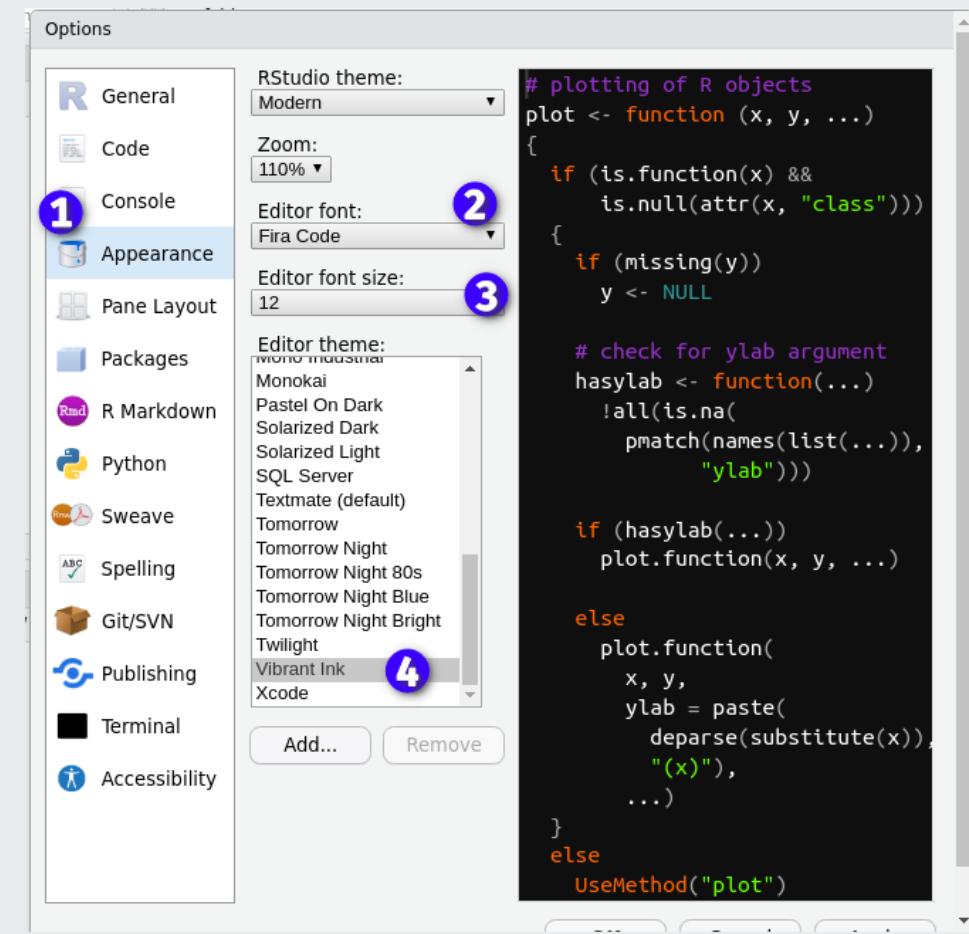
# Personalización de Rstudio



# Personalización de Rstudio

## Modificaciones sobre la interfaz de Rstudio

- Aumentar el zoom o utilizar el atajo:  
Ctrl y +
- Cambiar tipo de letra puede hacer un tanto más agradable la codificación. La letra Fira Code es bastante recomendable, pero se requiere instalar.
- El tema también puede ayudar en que la codificación sea más agradable. El paquete `rstthemes`, contiene muchos temas extras.



# ¡Hazlo tú mismo!

1. Mueve la consola a la derecha
2. Cambia el tema de tu rstudio
3. Cambia el zoom a tu agrado
4. Cambia el tamaño y tipo de letra a tu agrado

08 : 00

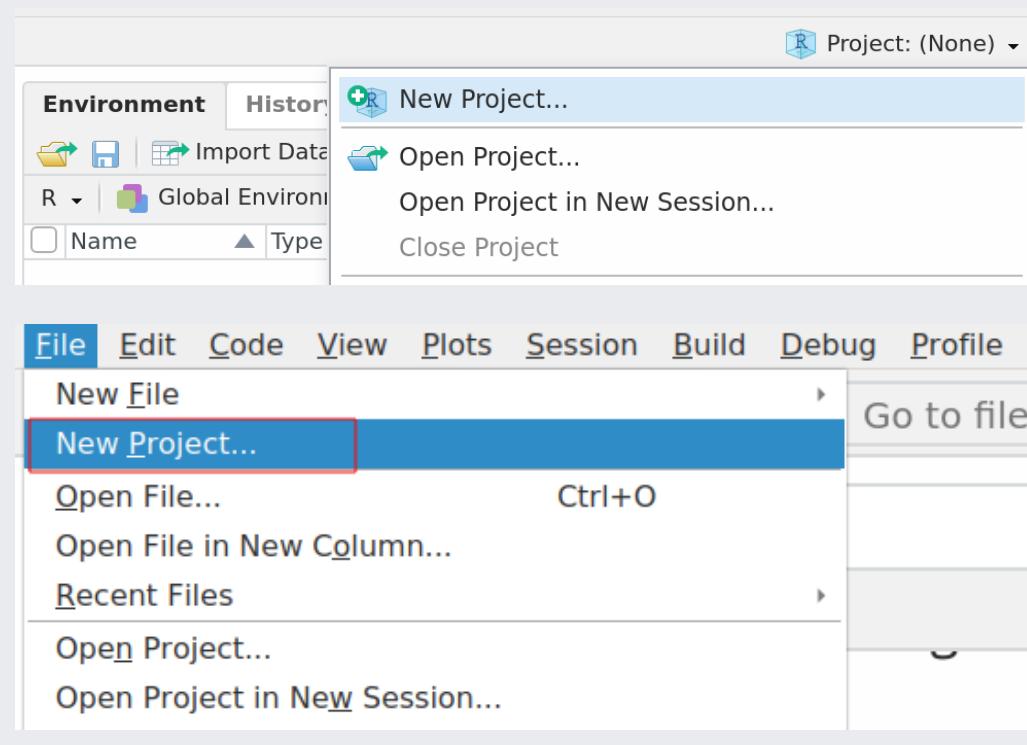
# ¿Por qué utilizar proyectos?

- Es más fácil poder compartir los proyectos y estos se encuentran listos para que otras personas puedan colaborar contigo.
- Cada proyecto se encuentra aislado. Los códigos en un proyecto no afectarán a ningún otro. Puedes tener muchos proyectos abiertos y los códigos del proyecto 1 no afectarán al proyecto 2.
- Muy útil para facilitar la importación de data.
- Mejora tanto la reproducibilidad como la colaboración.

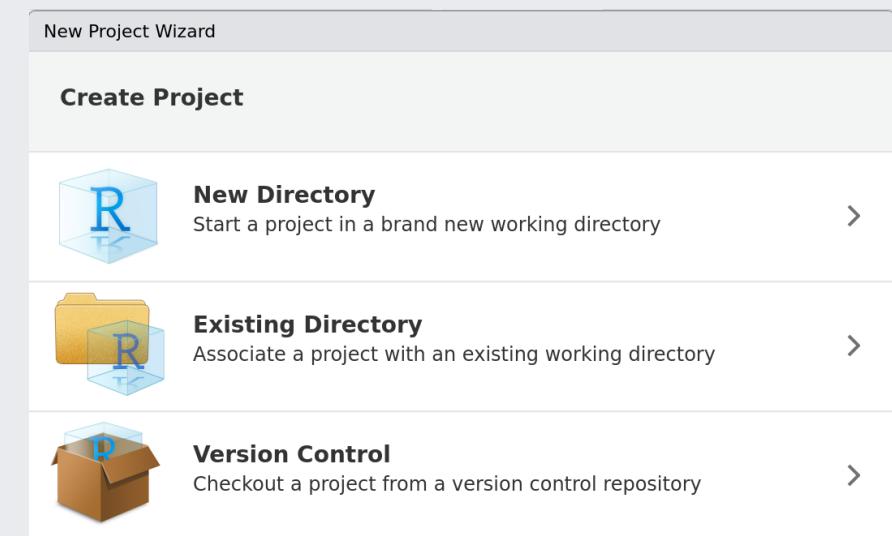
# Creación de proyectos

## PASOS:

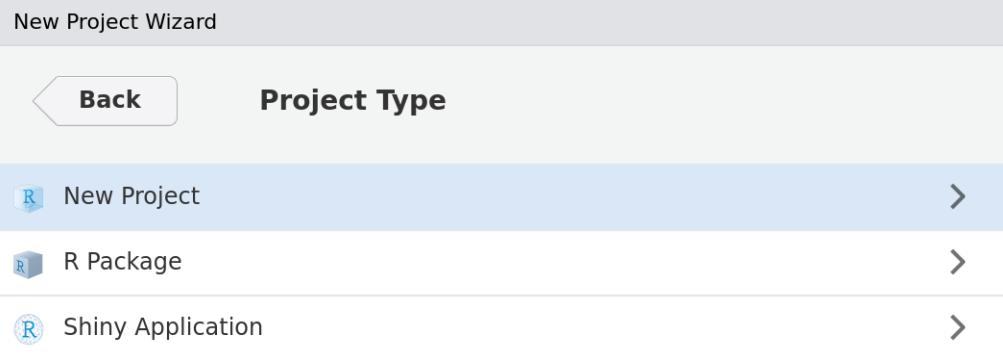
1. Seleccionamos “Project (None)” o “File” y luego, new project.



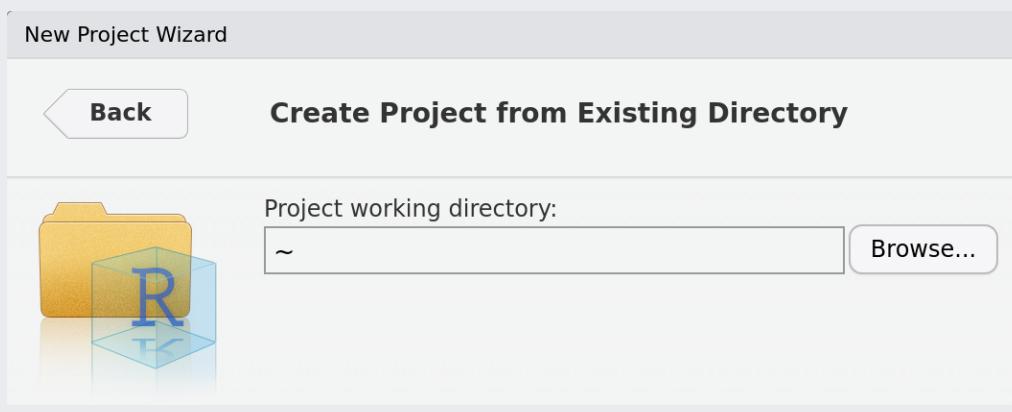
2. “New Directory” se utiliza para indicar dónde voy a almacenar mis archivos y para que R cree una nueva carpeta para mi proyecto, mientras que “Existing Directory” se utiliza si ya tengo una carpeta en la cual voy a almacenar mis archivos.  
Seleccionamos “New Directory”.



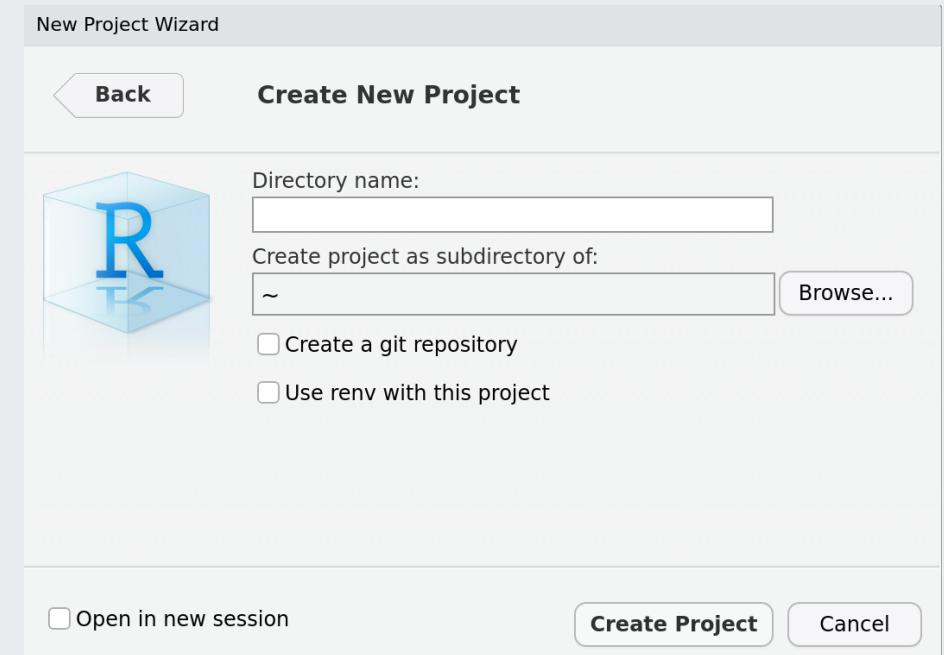
3. Aparecerán más opciones y seleccionamos nuevamente “New Project”



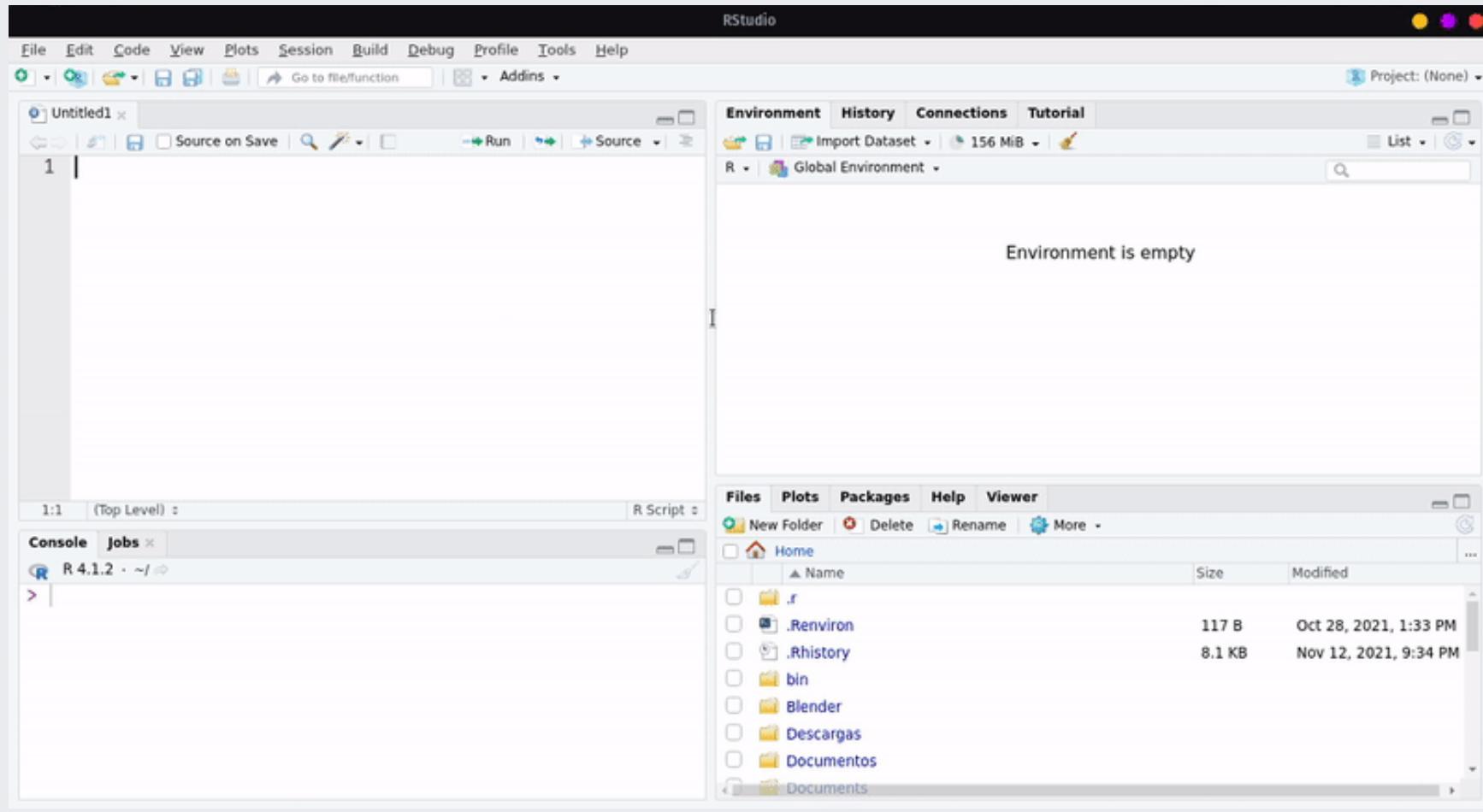
3. En caso de haber seleccionado “Existing Directory” aparecerá esto y buscamos el nombre de la carpeta que utilizaremos.



4. En “Directory name” ponemos el nombre de la carpeta que contendrá el archivo del proyecto, mientras que en “Create project as subdirectory of” seleccionamos dónde está la carpeta en la que trabajaremos.



# Creación de proyectos



# ¡Hazlo tú mismo!

1. Crea una carpeta para todo el curso
2. Dentro crea una carpeta para la primera sesión.
3. Crea un proyecto dentro de esa carpeta creada, por ej. Sesión 01.RProj.

04 : 00

# Vectores atómicos

- Los vectores contienen información homogénea o siempre de un solo tipo de datos.
- Un vector puede contener 1 solo elemento o n-elementos.
- Existen hasta 6 tipos de datos que puede contener un vector:
  - Logical: TRUE, FALSE
  - Integer: 1, 5, 7
  - Double: 3.15, 10, 12.86
  - Character: "Marcos", "Laptop".
  - Complex: 3 + 2i

```
v_numeric <- 5  
v_numeric
```

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

```
v_numeric <- c(5, 10, 15)  
v_numeric
```

```
## [1] 5 10 15
```

```
typeof(v_numeric)
```

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

```
v_character <- c("Laptop", "Rstudio", "4.2")  
v_character
```

```
## [1] "Laptop" "Rstudio" "4.2"
```

# Vectores atómicos

Los vectores solo pueden tener **un solo tipo de información a la vez**, así que si dentro de un vector se ingresa un elemento tipo `numeric`, este inmediatamente será transformado a `character`.

```
v_character ← c("Laptop", "Rstudio", 4.2)  
v_character
```

```
## [1] "Laptop"  "Rstudio" "4.2"
```

```
typeof(v_character)
```

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

# Vectores atómicos

Crearemos 2 vectores llamados `nombres` y `edades`.

```
nombres ← c("Luis", "Mateo", "Carlos", "Eduardo")
edades ← c(28, 30, 40, 35)
```

A partir de esto, podemos construir algo con lo que probablemente estemos más familiarizados.

```
data.frame(nombres,
            edades)
```

```
##   nombres edades
## 1 Luis     28
## 2 Mateo    30
## 3 Carlos    40
## 4 Eduardo   35
```

# ¡Hazlo tú mismo!

1. Se formarán grupos de 6 personas en salas de **Zoom**.
2. Cada uno de los participantes crearán n vectores llamados: **nombres** y otros que sean de su interés, como por ej. **profesion**, **edades**, **universidad**, **distrito**, etc.
3. El contenido de los elementos de los vectores a crear será real y a partir de las preguntas que se puedan hacer entre sí, dentro del mismo grupo de personas.
4. Por último, dentro de **data.frame()** pondrán los vectores que hayan creado. Deberían tener como mínimo, 2 columnas, e idealmente 6 filas.

10 : 00

# ¿Qué es una función?

Las funciones son módulos de código autónomo que realizan una tarea específica y generalmente, toman algún tipo de estructura de datos ([vector](#), [dataframes](#), etc.), lo procesan y devuelven un resultado.

El uso general de una función es el nombre de la función seguida de paréntesis

```
function_name(input)
```

Los inputs se denominan argumentos e incluyen:

- El objeto físico (cualquier estructura de datos) en el que la función lleva a cabo una tarea
- Especificaciones que alteran la forma en que opera la función

```
> sd(|)
```

◆ X =
◆ na.rm =

x

a numeric vector or an R object but not a factor coercible to numeric by `as.double(x)`.

Press F1 for additional help

# Buscando ayuda sobre las funciones

La mejor forma de averiguar esta información es utilizar `? seguido del nombre de la función`. Al hacer esto, se abrirá el manual de ayuda en el panel inferior derecho de RStudio que proporcionará una descripción de la función, uso, argumentos, detalles y ejemplos:

```
?sd()
```

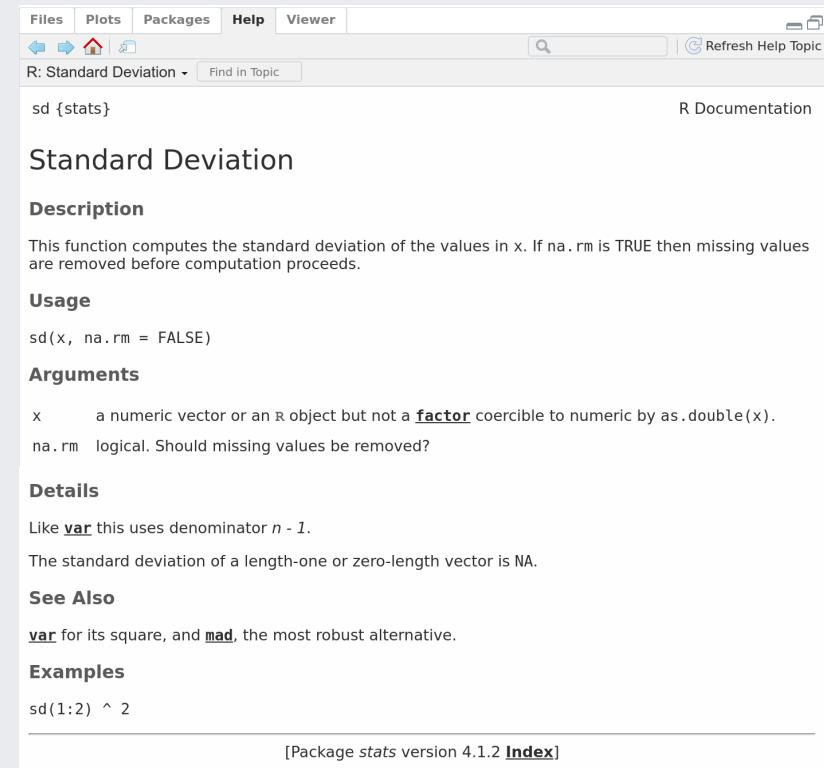
```
help(sd)
```

> `sd()` Presionar F1

Alternativamente, si está familiarizado con la función pero solo necesita recordar los nombres de los argumentos, puede usar:

```
args(sd)
```

```
## function (x, na.rm = FALSE)
## NULL
```



The screenshot shows the RStudio help viewer with the following content for the `sd` function:

- Files Plots Packages Help Viewer**
- R: Standard Deviation** | Find in Topic
- sd {stats}**
- Standard Deviation**
- Description**  
This function computes the standard deviation of the values in `x`. If `na.rm` is TRUE then missing values are removed before computation proceeds.
- Usage**  
`sd(x, na.rm = FALSE)`
- Arguments**
  - `x` a numeric vector or an R object but not a `factor` coercible to numeric by `as.double(x)`.
  - `na.rm` logical. Should missing values be removed?
- Details**  
Like `var` this uses denominator  $n - 1$ .  
The standard deviation of a length-one or zero-length vector is NA.
- See Also**  
`var` for its square, and `mad`, the most robust alternative.
- Examples**  
`sd(1:2) ^ 2`

[Package stats version 4.1.2 Index]

# Ejemplo de una función

Se tiene el número `3.15181930`, pero solo necesitamos dos decimales. Para ello, utilizaremos la función `round()` que redondea los números de acuerdo a la cantidad de decimales que asignemos. En este caso, solo necesitaremos 2.

```
round(3.15181930, digits = 2)
```

```
## [1] 3.15
```

Como se puede observar, se ha utilizado el argumento `digits` para regular la cantidad de decimales.

**Nota:** Si proporcionamos los argumentos en el mismo orden en el que han sido definidos, no es necesario nombrarlos

```
round(3.15181930, 2)
```

```
## [1] 3.15
```

# ¿Data.frame?

- Estructura de datos 2D
- Admite datos con diferente tipo de variable (lo opuesto a matrices)
- Similar a Microsoft Excel

**Se crean con la función:**

```
data.frame(  
  Var1 = elementos1,  
  Var2 = elementos2  
)
```

```
var1 ← c("Peru", "Argentina", "Bolivia")  
var2 ← rep("aceptado", 3)  
var3 ← seq(1000, 1200, 100)  
  
df ← data.frame(var1, var2, var3)  
df
```

```
##           var1     var2  var3  
## 1        Peru aceptado 1000  
## 2 Argentina aceptado 1100  
## 3    Bolivia aceptado 1200
```

```
df ← data.frame(  
  var1 = c("Peru", "Argentina", "Bolivia"),  
  var2 = rep("aceptado", 3),  
  var3 = seq(1000, 1200, 100)  
)  
df
```

# ¿Tibble?

- Son la versión mejorada del data.frame
- Disponible en el **paquete** tibble y por lo tanto en el tidyverse.

**Se crean con la función:**

```
library(tibble)
tibble(
  Var1 = elementos1,
  Var2 = elementos2
)
```

```
install.packages("tibble")
```

```
library(tibble)
tibble(
  var1 = c("Peru", "Argentina", "Bolivia"),
  var2 = rep("aceptado", 3),
  var3 = seq(1000, 1200, 100)
)
```

```
## # A tibble: 3 × 3
##   var1      var2     var3
##   <chr>     <chr>    <dbl>
## 1 Peru      aceptado 1000
## 2 Argentina aceptado 1100
## 3 Bolivia   aceptado 1200
```

# **data.frame() v.s. tibble()**

Ambas funciones tienen sus versiones `as.*` o `as_*` que permite transformar algo en lo que se desea. En este caso se estaría usando `as.data.frame` para convertir algo a **data.frame** y `as_tibble` para ese mismo objetivo.

```
class(iris)
```

```
## [1] "data.frame"

iris
## #> #> Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## #> #> 1 5.1 3.5 1.4 0.2 setosa
## #> #> 2 4.9 3.0 1.4 0.2 setosa
## #> #> 3 4.7 3.2 1.3 0.2 setosa
## #> #> 4 4.6 3.1 1.5 0.2 setosa
## #> #> 5 5.0 3.6 1.4 0.2 setosa
## #> #> 6 5.4 3.9 1.7 0.3 versicolor
## #> #> 7 4.6 3.4 1.4 0.2 versicolor
## #> #> 8 5.0 3.4 1.5 0.2 versicolor
```

```
as_tibble(iris)
```

```
## #> #> # A tibble: 150 × 5
## #> #> #> Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## #> #> #> <dbl> <dbl> <dbl> <dbl> <dbl>
## #> #> #> 1 5.1 3.5 1.4 0.2 setosa
## #> #> #> 2 4.9 3.0 1.4 0.2 setosa
## #> #> #> 3 4.7 3.2 1.3 0.2 setosa
## #> #> #> 4 4.6 3.1 1.5 0.2 setosa
## #> #> #> 5 5.0 3.6 1.4 0.2 setosa
## #> #> #> 6 5.4 3.9 1.7 0.3 versicolor
## #> #> #> 7 4.6 3.4 1.4 0.2 versicolor
## #> #> #> 8 5.0 3.4 1.5 0.2 versicolor
```

# ¡Hazlo tú mismo!

1. Crea un tibble a partir de los ganadores de las últimas copas mundiales. La estructura de la data debe contener las siguiente variables:
  - Año
  - Lugar
  - Ganador
2. Existe una base de datos integrada dentro de R que se llama `airquality`. Conviértela en un tibble, y guárdalo en otro objeto con distinto nombre.
3. Hacer lo mismo con la data `Titanic`.

08 : 00

# ¿Qué es un paquete?

Los paquetes son colecciones de funciones, datos y código compilado de R en un formato bien definido, creados para agregar una funcionalidad específica.

Hay un conjunto de paquetes estándar (o base) que se consideran parte del código fuente de R y están disponibles automáticamente como parte de su instalación de R.

# Instalación de paquetes desde CRAN

La forma de instalar un paquete dependerá de dónde se encuentre. Entonces, para los paquetes disponibles públicamente, esto significa a qué repositorio pertenece. La forma más común es usar el repositorio CRAN, luego solo necesita el nombre del paquete y usa el siguiente comando:

```
install.packages("paquete")
```

Después de ejecutar esto, recibirá algunos mensajes en la pantalla. Dependerán del sistema operativo que esté utilizando, las dependencias y si el paquete se instaló correctamente.

# Instalación de paquetes vía remotes

Cada repositorio tiene su propia forma de instalar un paquete a partir de ellos, por lo que en el caso de que utilice regularmente paquetes de diferentes fuentes, este comportamiento puede ser un poco frustrante. Una forma más eficiente es probablemente usar el paquete *remotes`* para simplificar este proceso.

```
install.packages("remotes")
```

Después de haber instalado *remotes`* podemos utilizar algunas de sus funciones para la instalación de paquetes:

- `remotes::install_bioc()` desde Bioconductor
- `remotes::install_github()` desde GitHub
- `remotes::install_version()` para instalar una versión específica de CRAN.

# Introducción a Tidyverse

El megapquete **tidyverse** permitirá hacer la gran mayoría de procedimientos necesarios para el análisis de datos, desde la importación, ordenamiento, limpieza, transformaciones, hasta la generación de gráficos (**ggplot2**).

La forma de instalarlo, es sencillo y se requiere ejecutar solo una vez por instalación de computadora/laptop:

```
install.packages("tidyverse")
```



# Retroalimentación

# ¡Gracias!

✉ [imt.innovlab@oficinas-upch.pe](mailto:imt.innovlab@oficinas-upch.pe)

🗣 [@healthinnovation](#)

🐦 [@innovalab\\_imt](#)

Estas diapositivas fueron creadas mediante el paquete `xaringan` y `xaringanthemer`.