Practica 1: Introducción a R

Bienvenido al curso,
Te recomendamos desarrollar esta guía de forma secuencial.
Cuando requieras apoyo, no dudes contactarte con los doecentes.

Antes de empezar: Descarga los archivos del curso

Descarga la carpeta comprimida del curso, cuyas indicaciones se encuentran en el Aula virtual.

Descomprime y extrae la carpeta, y guárdala en el escritorio de tu ordenador.

La estructura de la carpeta debería ser la siguiente

- Escritorio
  - EpiR-CDC
    - framingham_epi.xlsx
    - ubigeo_1891_distritos.xlsx

1. Conociendo R

¿Qué significa que R sea de "código abierto"?

- Que sus herramientas son creadas y revisadas por millones de usuarios. Esta ventaja permite a R responder rápidamente a necesidades emergentes, como lo fue la pandemia de COVID-19.

La base del software R está dirigida por un "grupo central" y se actualiza cada pocos meses. A cada **versión** se le asigna un número, como "R versión 4.1.2 (2021-11-01)".

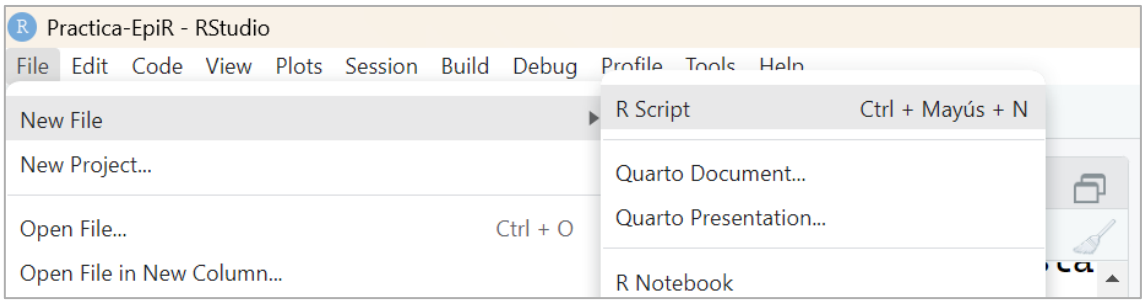
Ojo: R y RStudio suelen notificar sobre nuevas versiones disponibles y te pedirán que las descargues cuando sea necesario.

2. Conociendo RStudio

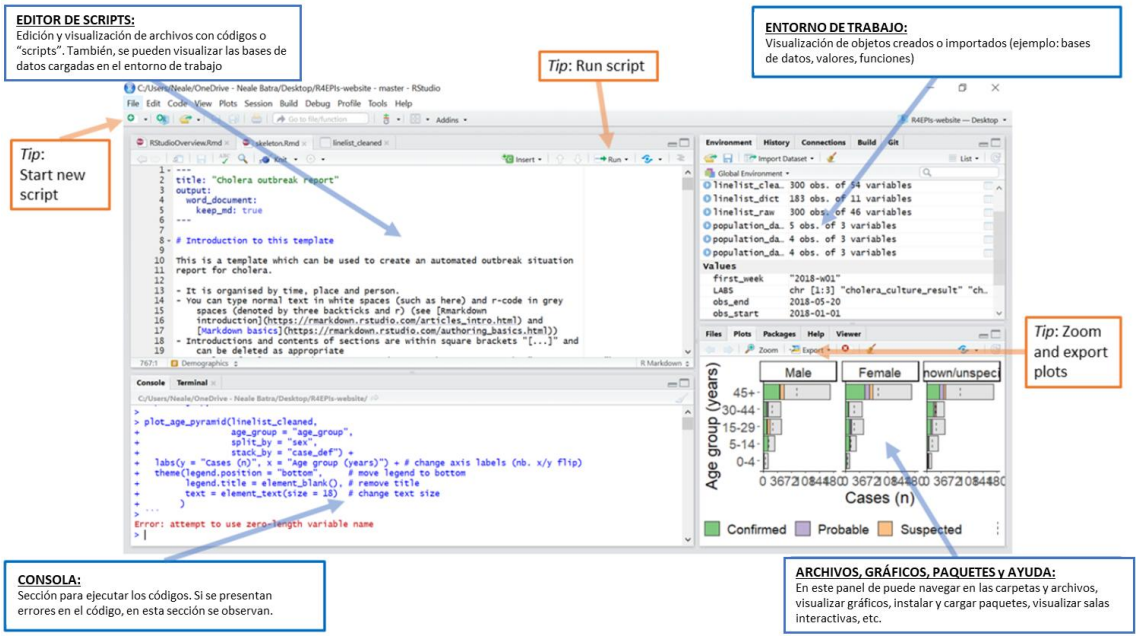
Abra RStudio y observe sus principales paneles. Debería ver 3 o 4 paneles:

1) El panel **Consola** (inferior-izquierda)
2) El panel **Entorno de trabajo** (superior-derecha)
3) El panel de **Archivos, Gráficos, Paquetes, Ayuda y Visor** (inferior-derecha)
4) El panel **Fuente o Editor de Scripts** (superior-izquierda). Si no observa este panel, haga clic en ***File -> New file -> R Script***

Autor: Mg. Jenny Chirinos



Se abrirá un nuevo Script de R y conseguirá el aspecto clásico mostrado en la imagen de abajo:



Con ayuda del diagrama, tómese unos minutos para identificar la ubicación de los cuatro paneles en RStudio.

3. Proyectos en RStudio

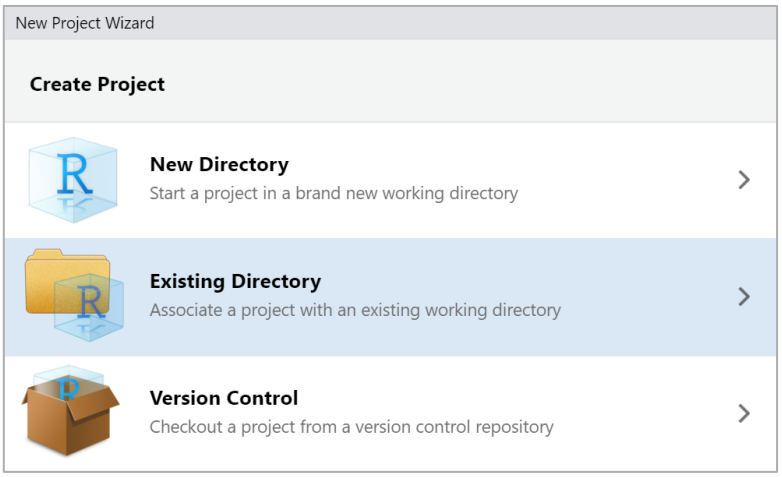
Durante este curso, crearás tres proyectos de RStudio dentro de la carpeta "EpiR-CDC":

1. Un proyecto para el desarrollo de las prácticas
2. Un proyecto para el desarrollo de las tareas
3. Un proyecto para el análisis e informe del trabajo final

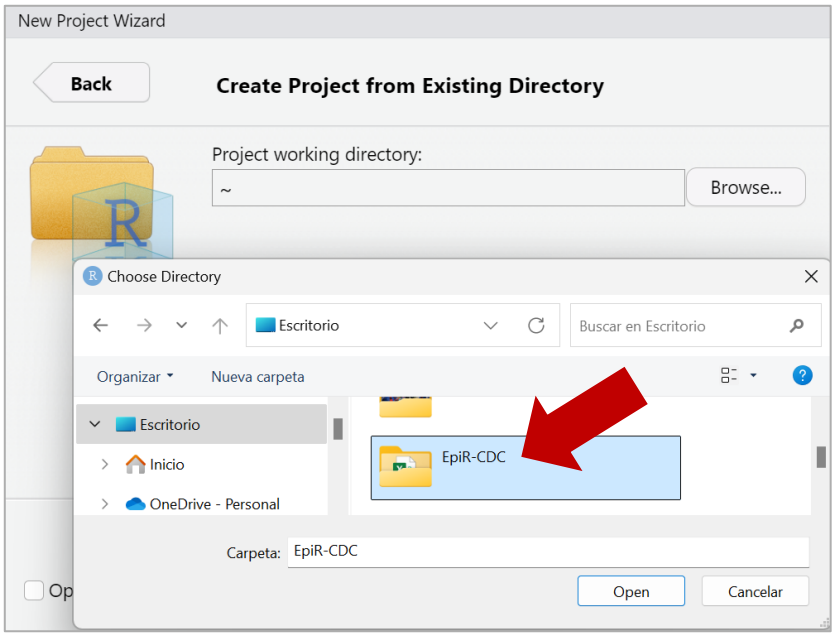
EJERCICIO 01: Crea tu primer proyecto en RStudio

Sigue estos pasos para crear tu primer proyecto de RStudio:

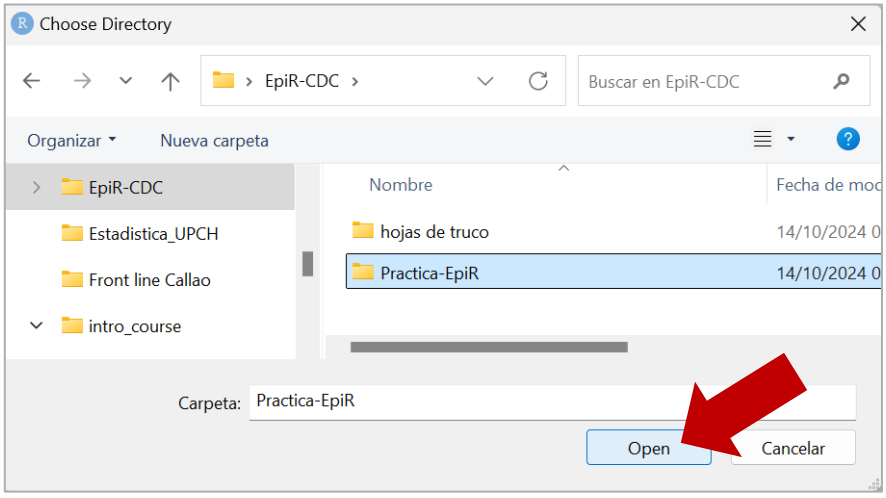
1. Abre RStudio (asegúrate de abrir RStudio y no solo R).
2. En RStudio, en la parte superior izquierda pulsa ***File -> New Project***. En la ventana emergente, selecciona "Directorio existente".



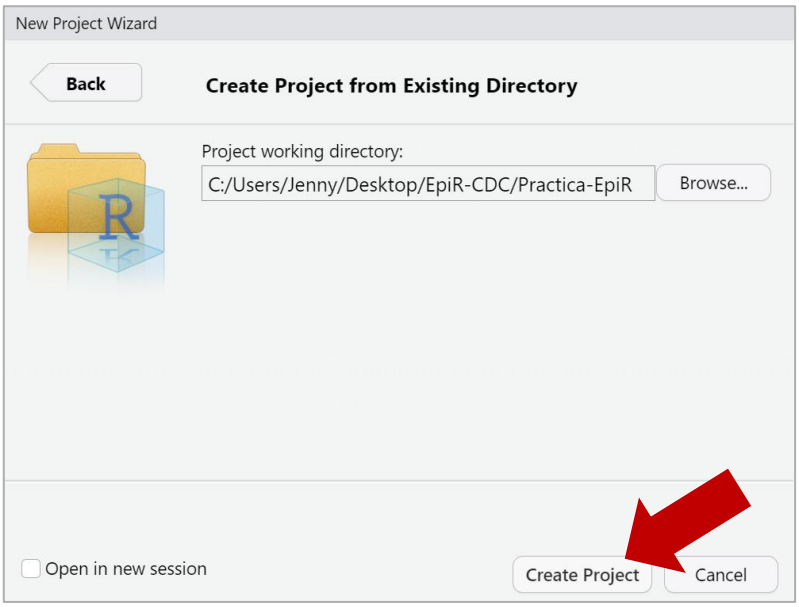
3. Ubica la carpeta **Epi-CDC** que descargaste y descomprimiste anteriormente (guardada en tu Escritorio), e ingresa dentro de ella con doble clic.



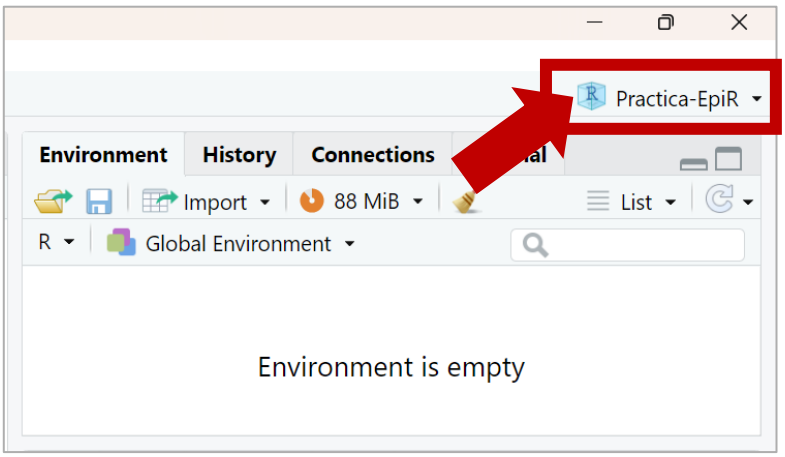
4. Dentro de la carpeta **Epi-CDC**, crea una subcarpeta con el nombre de **Practica-EpiR**, y da clic en Open.



5. Con ese paso se habrá seleccionado la subcarpeta **Practica-EpiR** como directorio para la creación del nuevo proyecto de R. Verifica la ruta y da clic a la opción ***Create Project***. Puede que RStudio se cierre brevemente y se vuelva a abrir.

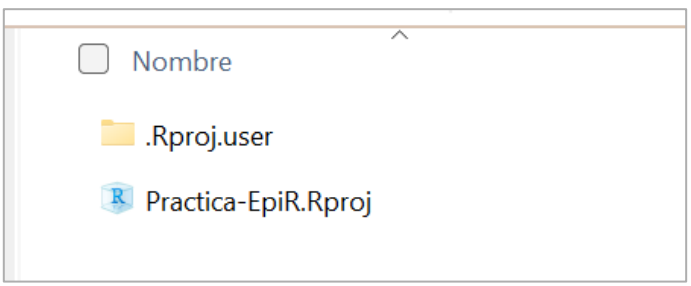


6. Verifica que estás en el nuevo proyecto creado, para lo cual deberás observar el nombre del proyecto en la esquina superior derecha de RStudio. Si no estás en un proyecto de RStudio, aparecerá "Proyecto: (Ninguno)". ***¿Qué ves en tu RStudio?***

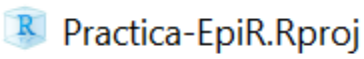


Adicionalmente, puedes explorar tu proyecto de RStudio:

Minimiza RStudio y abre la carpeta de archivos del curso que guardaste en tu escritorio, una vez dentro ingresa a la carpeta recién creada Practica-EpiR. El contenido de la carpeta debería parecerse a esto:



En la carpeta, deberías ver un pequeño archivo con un icono parecido a una "caja de R": ese es el archivo de **proyecto de RStudio (.Rproj)**.



Ojo: Para abrir el proyecto la próxima vez, solo deberás hacer doble clic en ese archivo. Se abrirá RStudio y todos los archivos de ese proyecto.

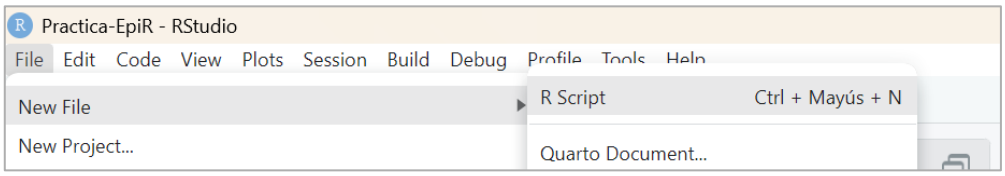
4. Ejecutando comandos de R

4.1. Scripts de R

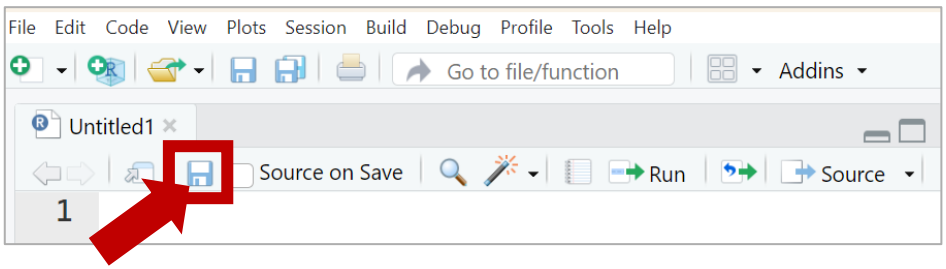
EJERCICIO 02: Crea y guarda un script de R

Sigue estos pasos para crear tu primer script de R:

1. Abre un nuevo script R haciendo clic en ***File -> New file -> R Script***.

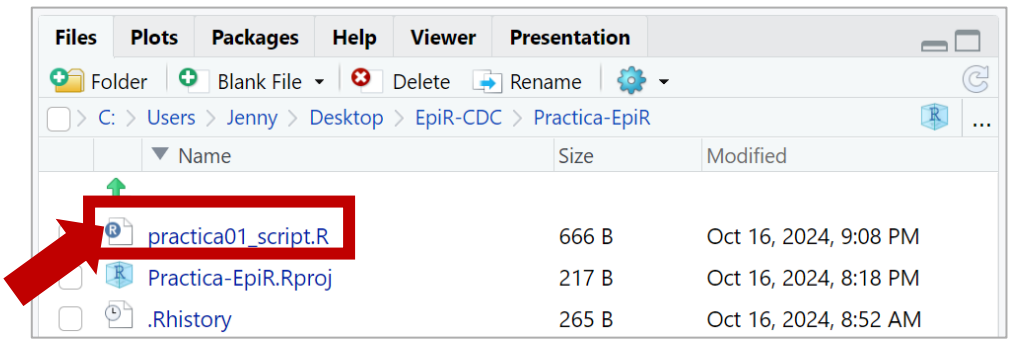


2. Haz clic en el icono de guardar situado encima del script, o haz clic en ***File -> Save as (Guardar como)***.



3. Nombra el script como "***practica01_script.R***" y asegúrate de que se haya guardado en el proyecto **"Practica-EpiR"**

4. Verifica que tu script haya aparecido en el panel "Archivos" (parte inferior derecha de RStudio) dentro de la carpeta "**Practica-EpiR**".



4.2. R como calculadora

R en su función más básica puede actuar como una calculadora

Recuerda que puedes ejecutar comandos de R a través de la consola o a través de un script. En ambos casos, los resultados siempre aparecerán en el panel de la Consola, aunque el comando se ejecute desde el script.

EJERCICIO 03: Realiza operaciones matemáticas desde un script

Presentación de caso: Cálculo del requerimiento de encuestas

Como parte de la investigación de un brote de EDA que afectó 3 localidades, se requiere gestionar el envío de encuestas impresas para la recolección de información en las 3 localidades. Basándose en la siguiente información, ¿cuántos juegos de encuestas se requiere enviar en total a las 3 localidades?

Datos:

- En la localidad 01 se encuestarán a 31 personas
- En la localidad 02 se encuestarán a 29 personas
- En la localidad 3 se encuestarán a 20 varones y el mismo número de mujeres
- Se ha considerado el envío de un 10% extra de encuestas

Escriba en un script un comando con las operaciones necesarias para el cálculo del número total de encuestas y ejecútelo.

4.3. Insertar comentarios en tu script

El símbolo "almohadilla" (#) desactiva cualquier texto escrito a la derecha del mismo y se utiliza para insertar **comentarios** a lo largo del script.

EJERCICIO 04: Añade comentarios a un script

Actualiza tu script generado como parte del ejercicio 03 incluyendo lo siguiente:

- Un encabezado que describa la finalidad del código (Cálculo del número de encuestas)

```
# Objetivo: Calcular número de encuestas
```

- Instrucciones detalladas sobre los comandos

```
# localidad 01 requerirá 31 encuestas
# localidad 02 requerirá 29 encuestas
# localidad 03 requerirá 20*2 encuestas
# Añadir el 10% extra de encuestas
```

5. Creación de objetos

En el proceso de creación de objetos en R, se emplea el operador de asignación (<–) para asignar valores al objeto

Atajo para operador de asignación: Alt + - (Windows) u Opción y - (Mac).

EJERCICIO 05: Crea objetos de tipo numérico e imprime sus valores

Creación de objetos:

1. Observa tu **panel de Entorno**, no debería haber ninguna entrada allí, porque aún no hemos definido ningún objeto.
2. Ahora, digita el siguiente código en tu script de R, y ejecutalo.

```
casos_confirmados <- 18
```

Ojo: Recuerda que para ejecutar un comando en tu script, resalta todo el comando o coloca el cursor en el comando y pulsa **"Ejecutar"**

3. Observa que el objeto `casos_confirmados` está ahora en el **panel de Entorno**, con el valor de 18.
4. Ahora, escribe el siguiente código en tu script de R, y ejecuta el comando.

```
casos_sospechosos <- 32
```

5. Observa que ahora en el Entorno se encuentran los 2 objetos creados. Asimismo, comprueba que los comandos ejecutados han aparecido en el panel de la Consola.
6. Finalmente, calcule el número total de casos (suma de casos confirmados y casos sospechosos) creando el tercer objeto `casos_totales` y empleando para el cálculo los dos objetos anteriormente creados.

```
casos_totales <- casos_confirmados + casos_sospechosos
```

Impresión de valores:

7. Imprime los valores asignados de los 3 objetos creados de la siguiente forma: En el script en una nueva línea, escriba el nombre del primer objeto creado `casos_confirmados` y ejecútelo. Observe la salida en el panel de la consola de R, se ha impreso únicamente el valor de 18.
8. Repita el mismo procedimiento para los otros dos objetos y observe que valores se imprimen en el panel de la Consola.

6. Funciones

¡El verdadero poder de R proviene de *las funciones!*

EJERCICIO 06: Utiliza las funciones `min()` y `max()`

1. Con las funciones indicadas, halla el valor mínimo y el valor máximo de los siguientes números 20, 35, 18, -10 y 99.

```
# Calcular el valor mínimo
min(20, 35, 18, -10, 99)
# Calcular el valor máximo
max(20, 35, 18, -10, 99)
```

Nota: Para las funciones estadísticas, asegurarse de envolverlos con la función c()

```
mean(1, 6, 12, 10, 5, 0)    # ¡¡¡ INCORRECTO !!!
mean(c(1, 6, 12, 10, 5, 0)) # CORRECTO
```

7. Función `c()` y creación de vectores

Un **vector** es una unidad de varios valores, *que deben ser de la misma clase* (todos numéricos, todos nominales, todos lógicos, etc.) y *deben estar separados por comas*.

Creamos vectores usando la función `c()` (combinar). Llamamos esta función y le damos como argumento los elementos que deseamos combinar en un vector, separados por comas.

Ejemplo 01: Creación de un vector con las edades de adultos mayores

```
# Crear un vector de tipo numérico con las edades de adultos may
ores
edades <- c(75, 90, 80, 85, 72)
```

Ejemplo 02: Creación de un vector con las nombres de distritos de Lima

```
# Crear un vector de tipo nominal con 4 distritos de Lima
distritos <- c("Breña", "Ate", "Miraflores", "Surquillo")
```

Ojo: Observa que los valores en el vector de tipo nominal deben estar escritos entre comillas.

EJERCICIO 07: Crear vectores con información rutinaria

1. Con ayuda de la función c(), crea tu primer vector con los nombres de tus compañeros de sala, escríbelos en minúsculas. Llama a este vector ***nombres***. No olvides colocar las comillas para ingresar cada valor dentro de la función.

2. Crea dos vectores más, para conocer las edades y las ciudades de tus compañeros. Llamarás a estos vectores: ***edades*** y ***ciudades.*** Para el caso de los valores del vector ciudades, escríbelos en mayúsculas.
3. Aplica la función `toupper()` en tu primer vector creado y observa como se cambian todos los caracteres a mayúsculas. Por si te resulta difícil descifrar el código, este es: **`toupper(nombres)`**
4. Aplica la función `tolower()` en el vector ciudades y observa como se cambian todos los caracteres a minúsculas. Por si te resulta difícil descifrar el código, este es: **`tolower(ciudades)`**

Ojo: Aunque el vector contenga valores digitados entre comillas, al teclear la tecla *nombre del objeto* o *nombre* del vector, no se utilizan comillas.

8. Argumentos de las funciones y documentación

EJERCICIO 08: Revisa documentación de una función y analiza sus argumentos

1. Digita el código de R que te permitirá revisar la documentación de la función **`hist()`** y ejecútalo.
2. Analiza el siguiente código e identifica cuales son los argumentos empleados:

```
hist(x=sqrt(islands), breaks = 12, freq = TRUE, col = "lightblue
", border = "pink")
```

9. Paquetes: instalación y cargado

Un **paquete de R** es *un paquete o colección de funciones relacionadas que podemos descargar y utilizar*. Una vez que un paquete se ha **instalado** se almacena en **tu "biblioteca" de R**. Puedes acceder a las funciones que contiene **"cargando"** el paquete para utilizarlo durante tu sesión actual de R.

`install.packages( )` es la función {base} de R para instalar paquetes

`library( )` es la función {base} de R para cargar paquetes

Ejemplo:

```
install.packages("janitor")

install.packages(c("janitor", "rio", "here", "tidyverse"))
```

Una vez descargados, se debe cargar los paquetes c*ada vez que inicies R.*

Ejemplo:

```
library(janitor)

library(rio)
```

EJERCICIO 09: Instala y carga nuevos paquetes **con funciones {base} R**

1. Empleando la función `install.packages( )` digita y ejecuta una nueva línea de código cerca de la parte superior de tu script para **instalar el paquete pacman**. Si una ventana emergente te pregunta si quieres reiniciar R, di "No".

   Ojo: Dentro de la función `install.packages( )`, los nombres de los paquetes deben ser escritos entre comillas.

2. Con ayuda de la función `library( )` escribe y ejecuta una nueva línea de código para **cargar el paquete pacman** recién instalado.

3. **Verifica las funciones** con las que cuenta el paquete {pacman} que ahora tienes disponibles para su uso.

El paquete "pacman" te ayuda eficazmente a *instalar y cargar* otros paquetes. Su función p_load() instala y carga automáticamente cada paquete *sólo si no está ya instalado*.

Ejemplo:

```
pacman::p_load(ggplot2, dplyr)
```

Tambien se puede ejecutar directamente:

```
p_load(ggplot2, dplyr)
```

EJERCICIO 10: Instala y carga nuevos paquetes con la función pload() de pacman

1. **Escribe y ejecuta una nueva línea de código para instalar los paquetes** rio(), here(), janitor(), tidyverse() empleando la función pload() de pacman
2. **Escribe y ejecuta el siguiente comando** library(pacman) para cargar el paquete pacman recién instalado.
3. **Verifica si los paquetes fueron instalados y cargados correctamente en la pestaña Packages**

10. Importar una base de datos

{rio} es el paquete para importar datos fácilmente

La función **`import()`** del paquete **{rio}** espera recibir un valor de nominal con la ruta del archivo que deseas importar. El archivo de datos puede estar guardado en la carpeta raíz del proyecto de RStudio o en una subcarpeta, por lo que sólo tienes que digitar la ruta y proporcionar el nombre del archivo y su extensión, entre comillas.

Impresión de la base de datos en la Consola:

```
import("bases/base01.csv")
```

Guardado de la base de datos como un objeto (para manipulación):

```
db <- import("bases/base01.csv")
```

EJERCICIO 11: Importa una base de datos a R

1. En tu proyecto de R, crea una carpeta que contenga la base de datos de Framingham (llama a esta carpeta "base_cruda")
2. Elabora el código de R que te permita importar esta base de datos, llama al objeto creado ***fram_01***
3. Verifica que el objeto creado se observe desde el Panel de Entorno de trabajo

Para ver la base de datos, haz clic en el nombre del objeto (*fram_01*) en el panel Entorno. Esta acción abrirá una nueva pestaña que muestra la base de datos.

¡Listo, estas preparado(a) para empezar a limpiar, manipular y analizar una base de datos!