



Casa abierta al tiempo



Introducción práctica a R y visualización de datos con ggplot2

M. en C. Luis Mario Hernández Soto

<https://www.researchgate.net/profile/Luis-Mario-Hernandez-Soto>



```
print("Hola mundo")
```

0 como un programador da la bienvenida a un curso

Antes que otra cosa. ¿Qué es programar?

- Escribir código para facilitar acciones específicas en una computadora, aplicación o programa
- Arte de los humanos comunicándose con las computadoras

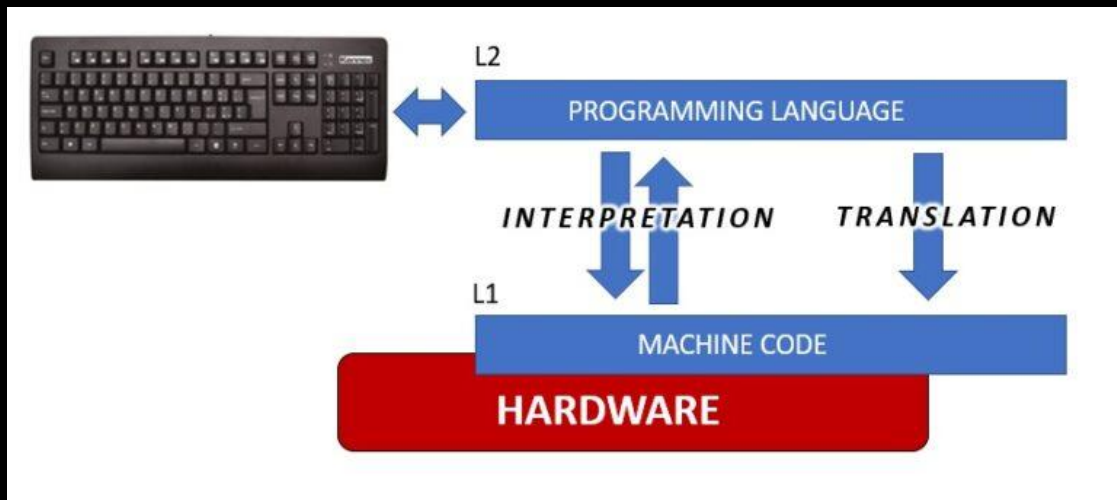
¿Código?

- Conjunto de líneas de texto que expresan, en un lenguaje de programación determinado, los pasos que debe seguir una computadora para la correcta ejecución de un programa o tarea específica.

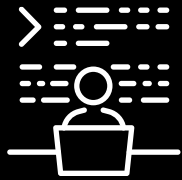
¿Lenguaje de programación?

- Serie de símbolos y reglas de sintaxis y semántica que definen la estructura principal del lenguaje y le dan un significado a sus elementos y expresiones.
- Permiten la comunicación usuario-máquina. Los compiladores (o intérpretes) convierten las instrucciones escritas en código en instrucciones escritas en lenguaje máquina (0 y 1).

Los lenguajes de programación se clasifican en diferentes niveles en función de su “proximidad” al hardware y su facilidad de uso para los programadores.



Niveles de programación



Bajo nivel

- Nivel más cercano a la máquina. Instrucciones en código binario.
- V: Máxima eficiencia y control sobre el hardware.
- D: Extremadamente difícil de escribir, leer y mantener. Altamente propenso a errores.



Medio nivel

- Permite manipulación directa de memoria y hardware, además de proporcionar estructuras de alto nivel.
- V: Balance entre control del hardware y abstracción. Más eficiente que los lenguajes de alto nivel, pero más fácil de usar que los de bajo nivel.
- Desventajas: No es tan fácil de usar como los lenguajes de alto nivel.



Alto nivel

- Nivel más cercano al humano. Fácil de leer y escribir ellos.
- V: Fácil de aprender y usar. Portable entre diferentes sistemas.
- D: Menos control sobre el hardware. Menos eficiente en rendimiento.

Ejemplos de “Hola mundo”

Bajo nivel (Machine, Ensamblador)

```
00000000 0000 0001 0001 1010 0010 0001 0004 0128
00000010 0000 0016 0000 0028 0000 0010 0000 0020
00000020 0000 0001 0004 0000 0000 0000 0000 0000
00000030 0000 0000 0000 0010 0000 0000 0000 0204
00000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
00000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfe
00000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
00000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
00000080 8888 8888 8888 8888 288e be88 8888 8888
00000090 3b83 5788 8888 8888 7667 778e 8828 8888
000000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
000000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
000000c0 8a18 880c e841 c988 b328 6871 688e 958b
```

```
1  section    .text
2  global     _start
3  _start:
4      mov     edx,len
5      mov     ecx,msg
6      mov     ebx,1
7      mov     eax,4
8      int     0x80
9      mov     eax,1
10     int     0x80
11  section    .data
12  msg        db  'Hello world',0xa
13  len        equ $ - msg
14
```

Medio nivel (C, Cobol)

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello World!\n");
6      return 0;
7  }
```

```
1  IDENTIFICATION DIVISION.
2  PROGRAM-ID.  HELLO-WORLD.
3
4  ENVIRONMENT DIVISION.
5
6  DATA DIVISION.
7
8  PROCEDURE DIVISION.
9  DISPLAY "Hello World!".
10 STOP RUN.
```

Alto nivel (R, SQL)

```
1  print("Hello world")
```

```
1  SELECT 'Hello world!';
```




R

- Lenguaje de programación de código abierto ∴ GRATUITO.
 - Sucesor de lenguaje “S”.
- Línea de comandos (scripts).
- Enfoque en estadística, análisis de datos, data science y machine learning.
- Multiplataforma (Windows, macOS y Linux).

R

- Lenguaje interpretado, no compilado, (scripts ejecutados directamente sin necesidad de construir un programa como en C)
- Comunidad grande a nivel mundial.
- Paquetes y funciones programados por la comunidad.
- Gran capacidad para la generación de gráficos (**ggplot2**).

3,000,000

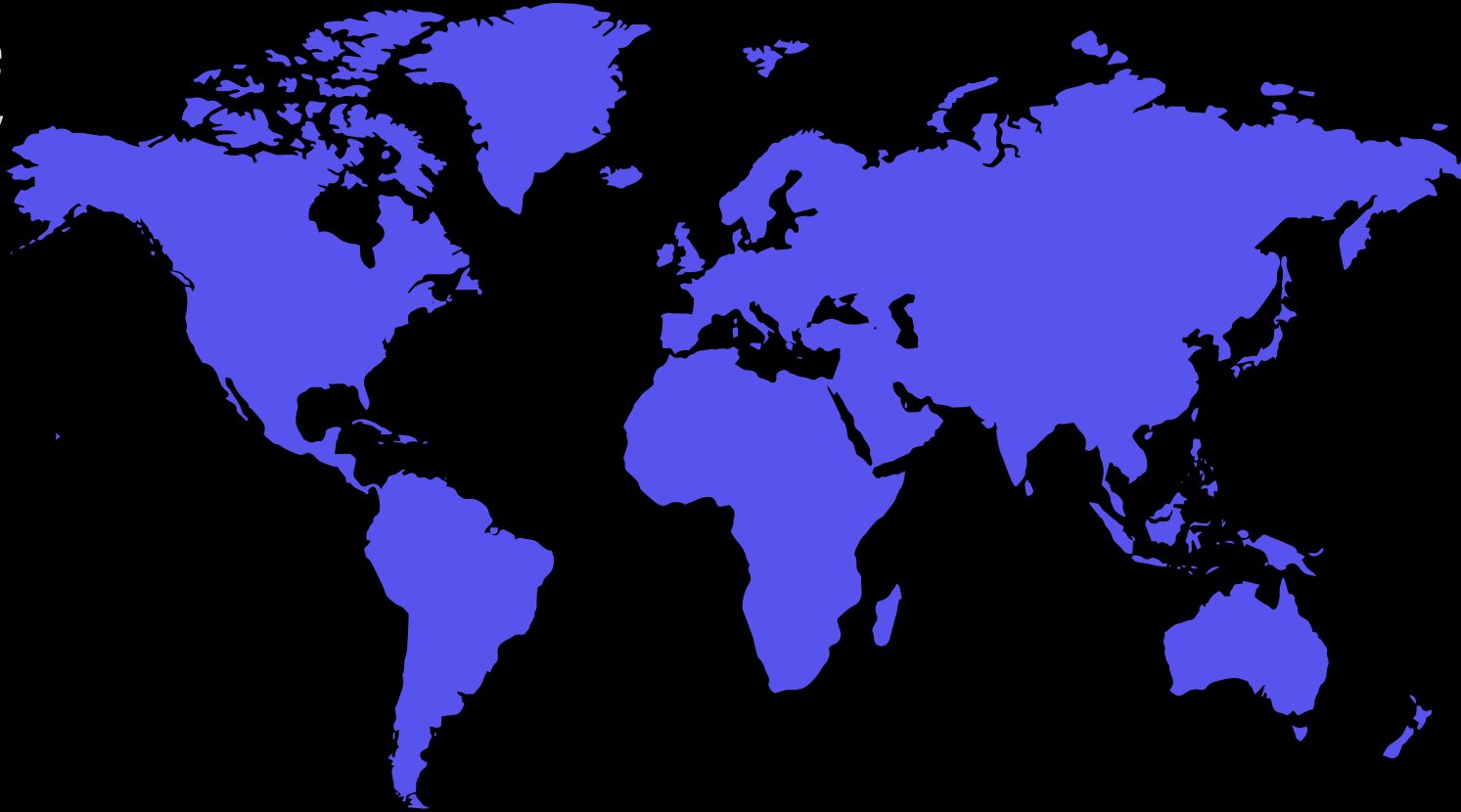
Usuario de R a nivel mundial

Febrero de 2023

Comunidad

Grupos de
trabajo y
desarrollo:

- R-Ladies
- Radicts
- IndyUseR
- Grupos locales

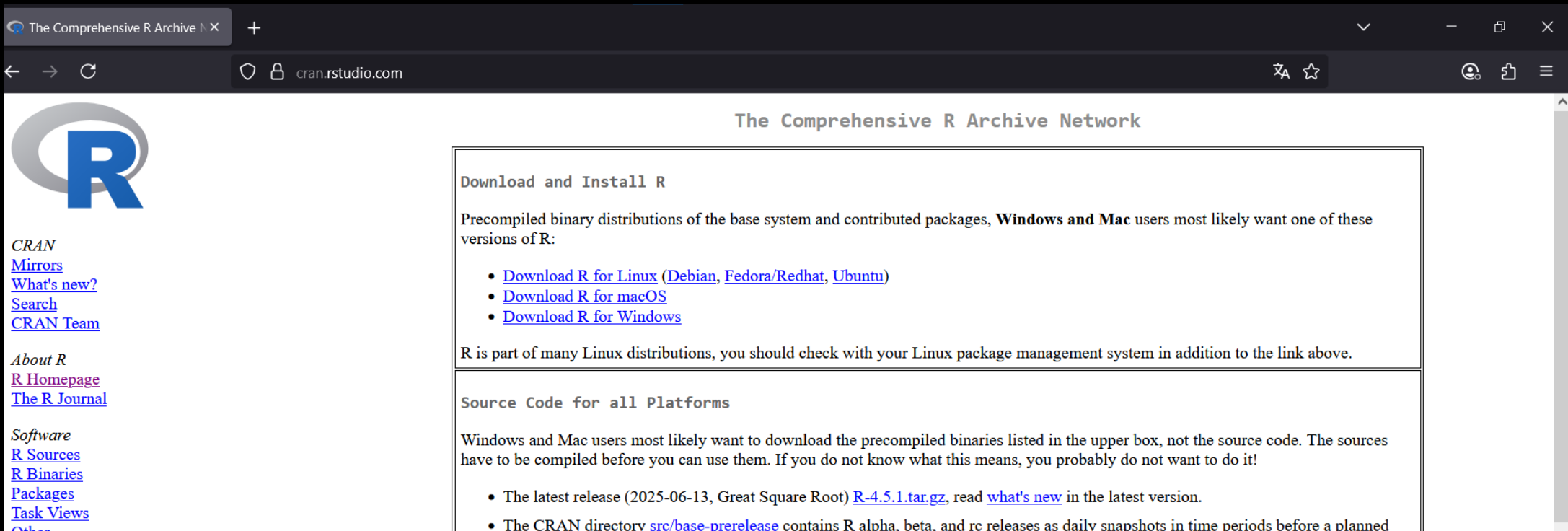


Patrocinadores:

- Microsoft
- Genentech
- Biogen
- Google
- Janssen
- Pfizer
- Merck

Instalación de R

- <https://cran.rstudio.com/>



The screenshot shows a web browser window with the address bar displaying "cran.rstudio.com". The page title is "The Comprehensive R Archive Network". The main content area is titled "Download and Install R" and contains the following text:

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux \(Debian, Fedora/Redhat, Ubuntu\)](#)
- [Download R for macOS](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2025-06-13, Great Square Root) [R-4.5.1.tar.gz](#), read [what's new](#) in the latest version.
- The CRAN directory [src/base-prerelease](#) contains R alpha, beta, and rc releases as daily snapshots in time periods before a planned

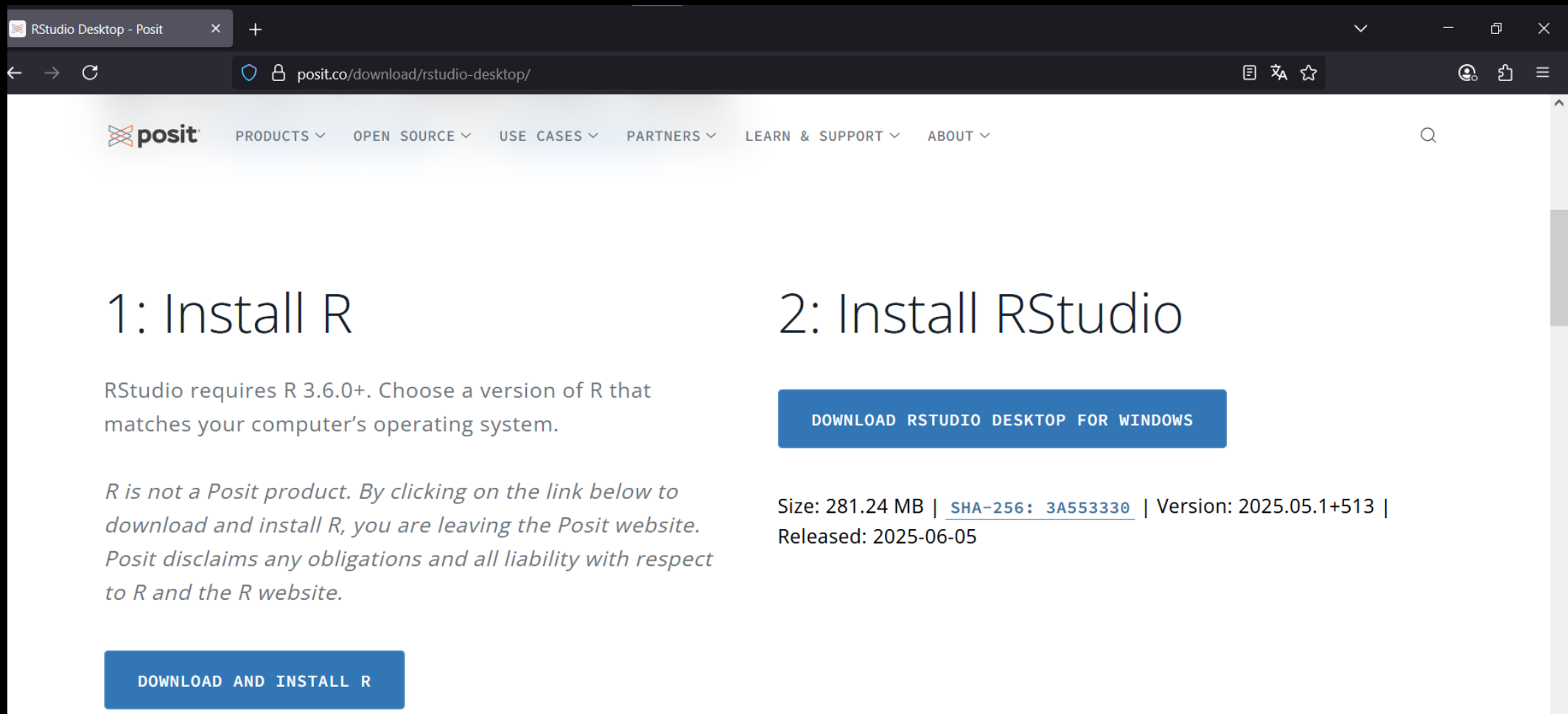
On the left side of the page, there is a sidebar with the R logo and links to "CRAN", "Mirrors", "What's new?", "Search", "CRAN Team", "About R", "R Homepage", "The R Journal", "Software", "R Sources", "R Binaries", "Packages", "Task Views", and "Other".

R

- Programar en un entorno completamente de línea de comandos puede ser complicado...
- Hay que usar un IDE (entorno de desarrollo integrado o *Integrated Development Environment* en inglés).
- RStudio es por mucho el más usado.

Instalación de RStudio

- <https://posit.co/download/rstudio-desktop/>



The screenshot shows a web browser window with the address bar displaying `posit.co/download/rstudio-desktop/`. The page features the Posit logo and a navigation menu with links for PRODUCTS, OPEN SOURCE, USE CASES, PARTNERS, LEARN & SUPPORT, and ABOUT. The main content is divided into two columns. The left column, titled '1: Install R', explains that RStudio requires R 3.6.0+ and provides a link to download and install R, accompanied by a disclaimer and a 'DOWNLOAD AND INSTALL R' button. The right column, titled '2: Install RStudio', features a 'DOWNLOAD RSTUDIO DESKTOP FOR WINDOWS' button and displays the software's size (281.24 MB), SHA-256 hash (3A553330), version (2025.05.1+513), and release date (2025-06-05).

posit[™] PRODUCTS ▾ OPEN SOURCE ▾ USE CASES ▾ PARTNERS ▾ LEARN & SUPPORT ▾ ABOUT ▾

1: Install R

RStudio requires R 3.6.0+. Choose a version of R that matches your computer's operating system.

R is not a Posit product. By clicking on the link below to download and install R, you are leaving the Posit website. Posit disclaims any obligations and all liability with respect to R and the R website.

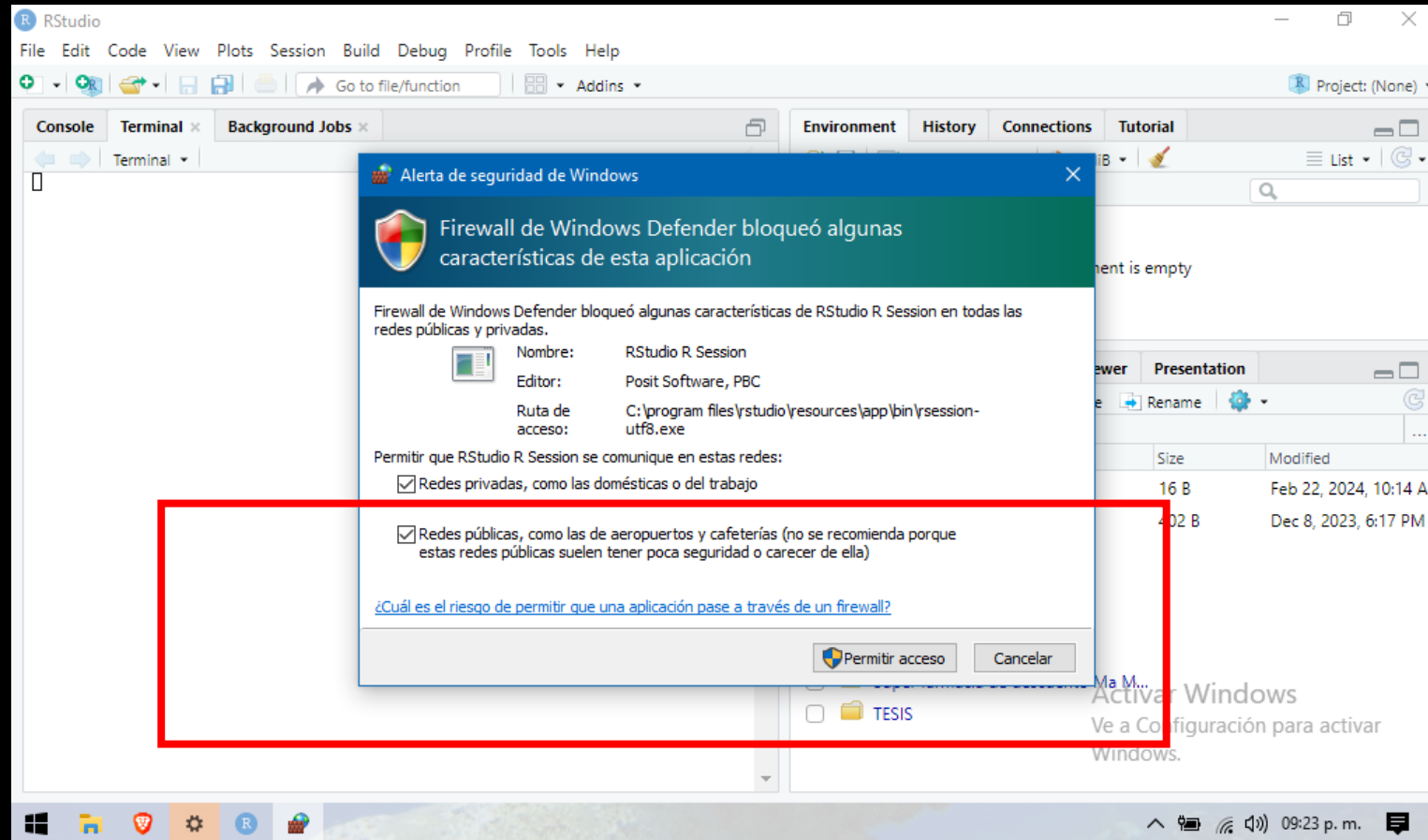
DOWNLOAD AND INSTALL R

2: Install RStudio

DOWNLOAD RSTUDIO DESKTOP FOR WINDOWS

Size: 281.24 MB | [SHA-256: 3A553330](#) | Version: 2025.05.1+513 | Released: 2025-06-05

Instalación de RStudio



Utilidades y facilidades de RStudio

- Cambiar el color de la ventana.
- Exportar y/o hacer zoom en gráficos.
- Instalar paquetes.
- Autocompletado de código
- Historial de comandos
- Panel de entorno.
- Atajos de teclado.

Conociendo R y RStudio

The screenshot displays the RStudio environment with the following components:

- Script:** The main editor window shows an R script named `cristian_rare.R`. The script includes comments in Spanish, library loading for `ggplot2`, `tidyverse`, and `ggsci`, setting the working directory, reading a TSV file, and creating a ggplot titled "Normalized rarefaction curve".
- Console:** The bottom-left pane shows the R version (4.1.1) and the current directory (`D:/Proyectos_y_revisiones/Académicos/Curso_R_poliinter pares/Proyecto/Poliinter pares/`). It also displays the standard R startup messages.
- Multipanel:** The bottom-right pane shows the file explorer with a table of files in the current directory.
- Workspace:** The top-right pane shows an empty environment.

Name	Size	Modified
..		
Poliinter pares.Rproj	218 B	Jun 12, 2024, 9:29 PM

Script

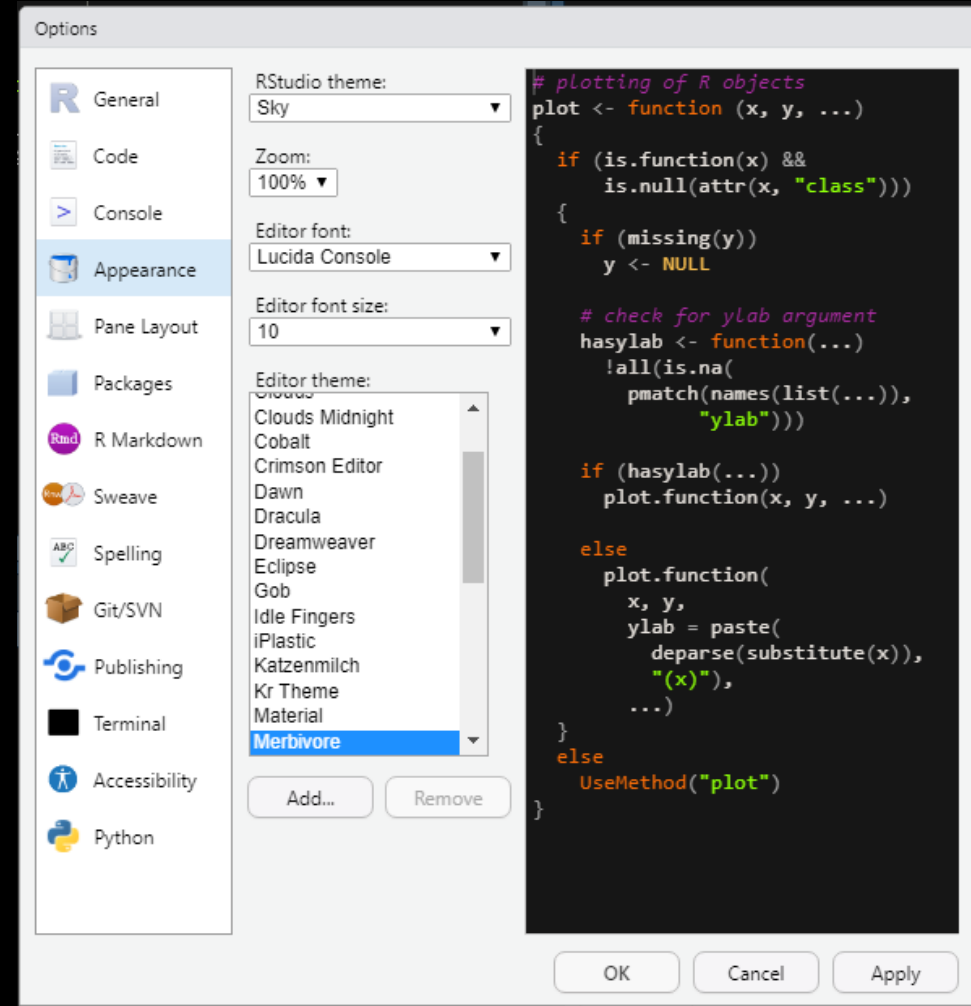
- Archivo de texto plano (extensión .R).
- Líneas de código (lista de comandos) que elaboramos.
- Contiene librerías, funciones, objetos, rutas de archivos o directorios, comentarios, operador de asignación (<- o =)
- Fecha y nombre de persona quien hace el código.
- Lista de librerías que serán utilizadas.
- Ruta de trabajo.
- **Llevar comentarios con el símbolo #.**
 - **Sirven para inhabilitar una línea o bien para dar alguna indicación, instrucción o nota. Úselos.**

Workspace

- Es un espacio en el cual se guardan los objetos creados en una sesión de R.
- Se guarda en la dirección de trabajo en formato RData.
- Este elemento será discutido en detalle más adelante.
- En R, los objetos que se encuentran en el workspace se pueden ver en la consola por medio de un comando o por medio de una ventana .

Conociendo RStudio

- Cambio de apariencia.
- Tools -> Global options -> Appearance



Proyecto en RStudio

- Antes que otra cosa. Hay que crear un proyecto.
 - **SU_PROYECTO.RData**
- Sirve para organizar el trabajo en R.
- Permite:
 - Mantener todos los archivos, datos y scripts.
 - En una carpeta dedicada.
- Facilitando gestión y acceso a análisis
- Evitando confusiones al trabajar con múltiples proyectos.
 - Puede tener muchos scripts en paralelo.

Aaaaahora sí

- Creemos un script porque tu sesión no viene por default.
- File → New File → R Script
 - Shift ⬆ + Ctrl + N (Windows).
 - Shift ⬆ + Command ⌘ + N (macOs).
- Lo ideal es que todos los comandos se corran desde un script. Se puede hacer desde la consola, pero no podremos recuperarlos después.

Asignación de valores

- Asigna un valor numérico, categórico, alfanumérico o lógico a un objeto.



```
> x <- 1
```

```
> x = 1
```

- Esta expresión significa que el objeto llamado x, tiene un valor asignado de 1.
- ¡CUIDADO! En R preferimos usar el operado de asignación
<-

Tips

- Para no escribir manualmente <- atajo Alt + -
- Botón “Run” sirve para correr una línea del script.
 - Atajo Alt + Enter (Windows) o Command + Enter (macOs).
- **A R no le gustan los objetos que empiezan con números o símbolos.**
- R es muy quisquilloso en el uso de mayúsculas y minúsculas. #
¡Qué NO hacer!

```
> 1m <- 5
```

```
> %w <- 4
```

Ejemplos de asignación de valores

```
>y <- 2
```

```
>w <- 2.3
```



```
>z <- 3 + 2i
```

```
>alfa <- "alfa"
```

```
>Beta <- "Y luego sigue gamma"
```

R es una calculadora con esteroides

- Operaciones matemáticas básicas



> $x + y$

> $w - y$

> $10 * w$

> $z + x$

> $\text{alfa} + w$

> $w \% y$

> $w \/% y$

> $\log_{10}(100)$

> $\text{sqrt}(25)$

Ver resultados de una operación

- Además, los resultados de las operaciones se pueden asignar a un objeto. Pero la consola no lo mostrará por default, habrá que llamarlo.



```
> suma <- x + y
```

```
> suma
```

```
> resta <- x - 4
```

```
> resta
```



```
> Multiplicacion <- w * 1000
```

```
> Multiplicacion
```

```
> multiplicacion
```

```
> raiz <- sqrt(81)
```

```
> raiz
```

Tips

- Buena práctica

> z != y & log(z) > 3



- # Mala práctica

> z!=y&log(z) > 3

Funciones

- El manejo de los datos se hace con funciones.
- Son :
 - Módulo de código autónomo
 - Realiza una tarea específica
 - Tomar una estructura de datos (valor, vector, dataframe, etc.), la procesar y devolver un resultado

Ejemplos de funciones

> print(x)

> help(print)

> c(1, 2, 3)

> usted <- readline("Introduce tu nombre: ")

Creemos un objeto de nombres

> nombres <- c("Julia", "Matías", "Mariana",
"Luis")

> nombres



Ejemplos de funciones

Creemos un objeto de con una serie de nombres.



- Función c (combine)

```
> nombres <- c("Julia", "Matías", "Mariana",  
  "Luis")
```

```
> nombres
```

- # Creemos un objeto equisye con nombres para la variable con nombres x e y, alternando entre ellos, numerados y separados por un _



```
> equisye <- paste(c("x","y"), rep(1:10, each=2), sep="_")  
  
> equisye
```

Objetos en R

- R es un lenguaje de programación orientado a la creación de objetos.
- Cualquier cosa a la que pueda asignársele una variable.
- Objetos permiten al usuario almacenar información necesaria para su proyecto.

Clases de objetos

- Numeric (num): números reales o decimales
 - 1.0 – 2.6 – 4.6
- Integer (int): números enteros
 - 1 – 8 – 467
- Complex (cplx): números complejos.
- Character (chr): caracteres.
- Logical (logi): resultados lógicos (TRUE o FALSE).

Cambio de clase

- Las clases se pueden usar a conveniencia del analista.
- Variables categóricas o numéricas.
- Intercambiables con las siguientes funciones
 - `as.numeric()`
 - `as.integer()`
 - `as.character()`
 - `as.logical()`
 - `as.complex()`.



Cambio de clase

Cambiamos la clase del objeto y de numeric a character

```
> class(y)
```

```
> y
```

```
> x + y
```

```
> y <- as.character(y)
```

Checamos la clase del objeto

```
> class(y)
```

```
> y
```

```
> x + y # ¿Qué pasó aquí?
```



¿Y esto es irreversible?

Regresemos el objeto a su clase original

```
> y <- as.numeric(y)
```



```
> class(y)
```

```
> y
```

```
> x + y
```

¿Y R solo es cuantitativo?

- ¡NO! R puede tomar variables.
- Clase factor.
- Seguiremos trabajando con el objeto nombres.
- Ahora le asignaremos una variable por sexo.

Sigamos

```
# Veamos nombres.
```

```
> nombres
```

```
> class(nombres)
```

```
# creamos el objeto sexo
```



```
> sexo <- factor(c("Femenino", "Masculino", "Femenino",  
"Masculino"))
```

```
# Revisamos la estructura del objeto sexo
```

```
> str(sexo)
```

```
# Revisemos la clase
```

```
> class(sexo)
```

Sigamos

Modifiquemos eso

```
> sexo <- as.factor(sexo)
```

Revisamos la nueva estructura del objeto sexo

```
> str(sexo)
```

Revisemos la nueva clase del objeto sexo

```
> class(sexo)
```

Niveles del objeto de clase factor

```
> levels(sexo)
```



Vectores

- Antes de seguir.
- Estos objetos simples que tienen un solo tipo de valor son llamados VECTORES.
- Los vectores son homogéneos y unidimensionales.
 - sexo y nombres tienen valores del mismo tipo.
- Hay más y más complejas estructuras de datos.

Listas

- Son estructuras de datos que pueden contener elementos de diferentes tipos, incluyendo números, caracteres, vectores, e incluso otras listas.
- Son heterogéneas y unidimensionales.



- # creamos el objeto l1 de clase lista

```
> lista1 <- list(usted, y, nombres)
```

```
> lista1
```

La consola respondió esto:

```
[[1]]
```

```
[1] "Mario"
```

```
[[2]]
```

```
[1] 2
```

```
[[3]]
```

```
[1] "Julia" "Matías" "Mariana" "Luis"
```

- Esa numeración nos permite acceder a partes específicas de la lista.
- # consultando el tercer slot del objeto lista1



```
> lista1[3]
```

```
[[1]]
```

```
[1] "Julia" "Matías" "Mariana" "Luis"
```

```
# Poniéndole el nombre a los slots del objeto lista1
```

```
> names(lista1) <- c("objeto_usted", "objeto_y",  
  "objeto_nombres")
```

```
> Lista1
```



```
# Esto permite llamar a un slot tanto por su índice o por su nombre  
usando el símbolo $.
```

```
> lista1$objeto_nombres
```

```
> lista1[3]
```

```
> lista1$objeto_nombres
```

```
> lista1$objeto_y
```

```
> lista1$objeto_usted
```

Matrices

- Una matriz en R es un conjunto de 2 dimensiones (filas y columnas) en los que cada elemento tiene la misma clase.
- Al igual que los vectores son homogéneas, pero a diferencia de estos, son bidimensionales.

Haciendo matrices

- Hagamos dos matrices ligeramente diferentes.

Una matriz 2 filas por 3 columnas ordenado por columnas

```
> matriz1 <- matrix( c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3, byrow =  
  FALSE)  
> matriz1  
> class(matriz1)
```



Haciendo matrices

Una matriz 2 filas por 3 columnas ordenado por filas

```
> matriz2 <- matrix( c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3, byrow =  
  TRUE)
```

```
> matriz2
```

```
> class(matriz2)
```



Haciendo matrices.

Haremos un objeto más complejo

Crearemos un vector llamado ventas_1 con los números 2 , 3 , 4.5 y 5 (millones de pesos).

```
> ventas_1 <- c(2, 3, 4.5, 5)
```

```
> ventas_1
```



Ahora repetiremos los valores 3 veces en un objeto llamado v2.

```
> v2 <- rep(ventas_1, times=3)
```

```
> v2
```

```
> # Repetimos cada elemento del objeto ventas_1 3 veces en un  
objeto llamado v2
```

```
> v3 <- rep(ventas_1, each=3)
```

```
> v3
```

creando una matriz uniendo los dos vectores como columnas.
Hay que recordar que las matrices son de datos homogéneos



```
> m2 <- cbind(v2, v3)  
> m2
```

Arrays

- Similar a las matrices.
- Tienen más de dos dimensiones.
- Esto nos permite construir cubos de datos u objetos de datos con más de dos dimensiones.

creando un array de dos matrices cada una con 3 filas y 3 columnas



```
> a1 <- array(c(5, 9, 3, 10, 11, 12, 13, 14, 15),dim = c(3,3,2))  
> a1
```

Comparemos matrices con arrays

```
> m2
      v2 v3
[1,] 2.0 2.0
[2,] 3.0 2.0
[3,] 4.5 2.0
[4,] 5.0 3.0
[5,] 2.0 3.0
[6,] 3.0 3.0
[7,] 4.5 4.5
[8,] 5.0 4.5
[9,] 2.0 4.5
[10,] 3.0 5.0
[11,] 4.5 5.0
[12,] 5.0 5.0
```

```
> a1
, , 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
```

Data.frames

- Similar a las matrices
 - Pero
 - Cada columna corresponde a una variable
 - Cada fila a las observaciones.
- Diferencia cada variable (columna) puede ser de diferente clase.

- Llegó la hora de unir todos los vectores con los que hemos trabajado.
 - ventas_1 (clase numeric) tenemos las ventas realizadas
 - nombres (clase character) tenemos los nombres de los clientes
 - sexo (clase factor) tenemos el sexo de cada cliente.
- Unamos estas variables en una sola base de datos para su análisis posterior, empleando la función `data.frame()`.



```
# creando un data frame
```

```
> d_ventas <- data.frame(ventas_1, nombres, sexo)
```

```
> d_ventas
```

```
> class(d_ventas)
```

```
# Veamos la estructura del data frame con la función str()
```

```
> str(d_ventas)
```

Estructura del dataframe.

```
> d_ventas
  ventas nombres      sexo
1    2.0   Julia Femenino
2    3.0  Matías Masculino
3    4.5 Mariana Femenino
4    5.0    Luis Masculino
> str(d_ventas)
'data.frame':   4 obs. of  3 variables:
 $ ventas : num  2 3 4.5 5
 $ nombres: chr  "Julia" "Matías" "Mariana" "Luis"
 $ sexo   : Factor w/ 2 levels "Femenino","Masculino": 1 2 1 2
> |
```

- Los data.frames no son inmutables. Vamos a modificar el nombre de la variable (columna) ventas_1 por ventas. Usaremos la función names()
cambiando el nombre de la primera variable.



```
> names(d_ventas)[1] <- "ventas"  
> d_ventas
```

Recuperando información de un data frame

- Igual que en lista, matrix y array. Se puede acceder a una variable (columna) particular.
- Se usa el símbolo \$



```
> class(d_ventas$sexo) #veamos la clase de la variable sexo
```

Recuperar un dato en particular

segundo dato de la variable que se encuentra en la tercera columna

> d_ventas[2,3]



También se puede ver una fila completa

extrayendo la segunda observación para todas las variables

> d_ventas[2,]

O una columna completa

extrayendo la tercera columna



```
> d_ventas[,3]
```

```
> d_ventas[,2]
```

“No reinventes la rueda”

- En otras palabras, no pierdas mucho tiempo creando lo que alguien más ya hizo.
- Práctica de desarrollar soluciones o algoritmos desde cero, en lugar de utilizar soluciones existentes y bien establecidas.



“No reinventes la rueda”.

Pros:

- Entendimiento completo del problema.
- Personalización a necesidades específicas.
- Aprendizaje profundo.
- Libertad Creativa.

Contras:

- Tiempo y recursos.
- Calidad y mantenimiento dando soluciones menos eficientes.
- Reinención de problemas ya resueltos por otros.
- Omisión de aprender de soluciones existentes y mejores prácticas.

Paquetes en R

- Las expresiones (o funciones) son conjuntos de instrucciones que se utilizan para que R realice operaciones sobre objetos.
- R tiene un core básico. Con funciones como las operaciones aritméticas.
- Comunidad expande esas capacidades para hacer tareas mucho más complejas.

Paquetes

- Los usuarios han construido paquetes.
 - Diversas funciones:
 - Tareas generales (gráficas, instalación de herramientas).
 - Tareas específicas (Estadísticas particulares, implementación de modelos matemáticos).
- Los paquetes de R son un conjunto de funciones y bases de datos desarrollados por la comunidad.



Paquetes

Files

Plots

Packages

Help

Viewer

+

Install

↺

Update

🔍

🔍

	Name	Description	Version		
User Library					
<input type="checkbox"/>	AnnotationDbi	Manipulation of SQLite-based annotations in Bioconductor	1.56.2	🌐	🔍
<input type="checkbox"/>	AnnotationFilter	Facilities for Filtering Bioconductor Annotation Resources	1.18.0	🌐	🔍
<input type="checkbox"/>	ape	Analyses of Phylogenetics and Evolution	5.6-2	🌐	🔍
<input type="checkbox"/>	aplot	Decorate a 'ggplot' with Associated Information	0.1.4	🌐	🔍
<input type="checkbox"/>	askpass	Safe Password Entry for R, Git, and SSH	1.1	🌐	🔍
<input type="checkbox"/>	assertthat	Easy Pre and Post Assertions	0.2.1	🌐	🔍
<input type="checkbox"/>	babelgene	Gene Orthologs for Model Organisms in a Tidy Data Format	22.3	🌐	🔍
<input type="checkbox"/>	backports	Reimplementations of Functions Introduced Since R-3.0.0	1.4.1	🌐	🔍
<input type="checkbox"/>	base64enc	Tools for base64 encoding	0.1-3	🌐	🔍
<input type="checkbox"/>	BH	Boost C++ Header Files	1.78.0-0	🌐	🔍
<input type="checkbox"/>	Biobase	Biobase: Base functions for Bioconductor	2.54.0	🌐	🔍
<input type="checkbox"/>	BiocFileCache	Manage Files Across Sessions	2.2.1	🌐	🔍
<input type="checkbox"/>	BiocGenerics	S4 generic functions used in Bioconductor	0.40.0	🌐	🔍
<input type="checkbox"/>	BiocIO	Standard Input and Output for Bioconductor Packages	1.4.0	🌐	🔍
<input type="checkbox"/>	BiocManager	Access the Bioconductor Project Package Repository	1.30.17	🌐	🔍
<input type="checkbox"/>	BiocParallel	Bioconductor facilities for parallel evaluation	1.28.3	🌐	🔍
<input type="checkbox"/>	BiocVersion	Set the appropriate version of Bioconductor packages	3.14.0	🌐	🔍
<input type="checkbox"/>	biomaRt	Interface to BioMart databases (i.e. Ensembl)	2.50.3	🌐	🔍
<input type="checkbox"/>	Biostrings	Efficient manipulation of biological strings	2.62.0	🌐	🔍
<input type="checkbox"/>	biovizBase	Basic graphic utilities for visualization of genomic data.	1.42.0	🌐	🔍
<input type="checkbox"/>	bit	Classes and Methods for Fast Memory-Efficient Boolean Selections	4.0.4	🌐	🔍

Instalación de paquetes

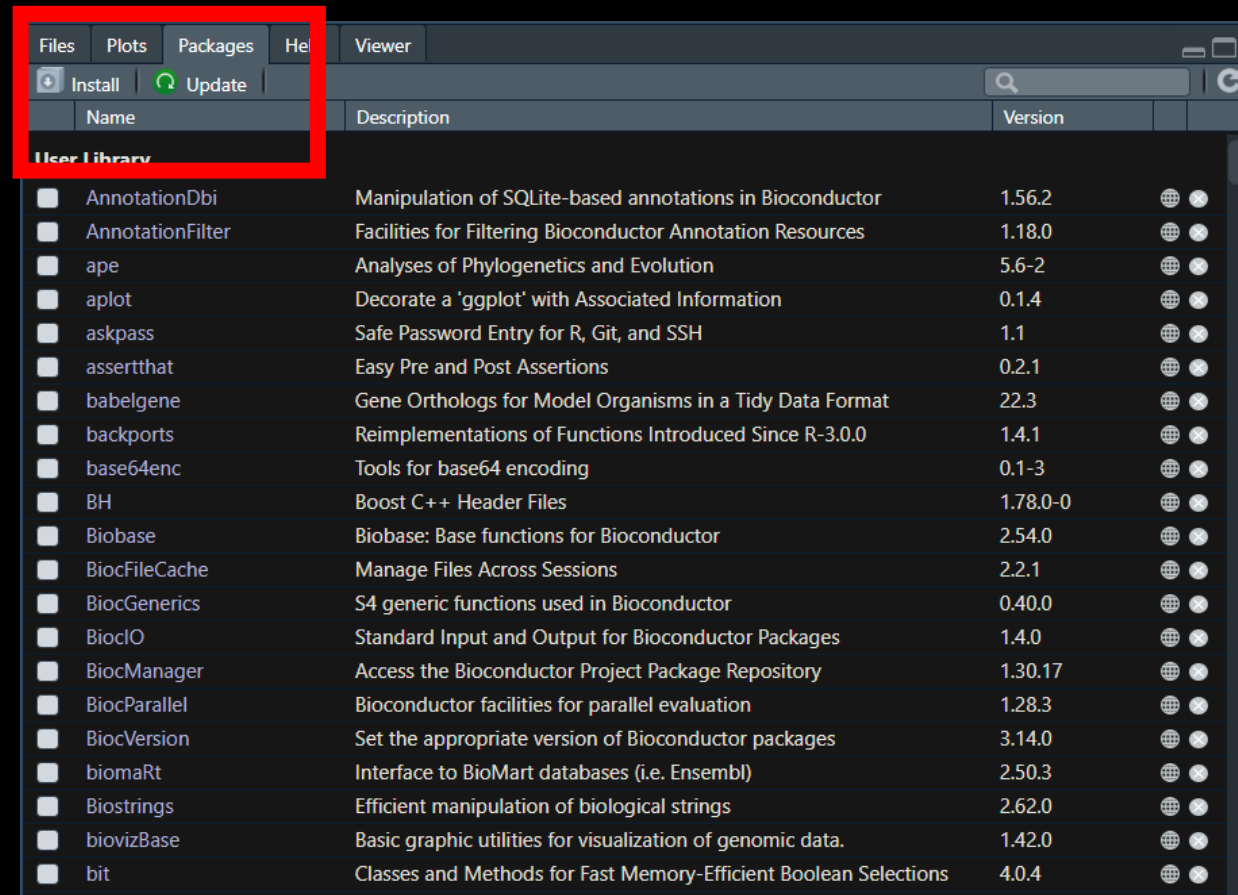
- Varias formas.
- Usar la función `install.packages()` .
 - El argumento es el nombre del paquete y se expresa entre comillas; es decir,
 - > `install.packages("nombre del paquete")`



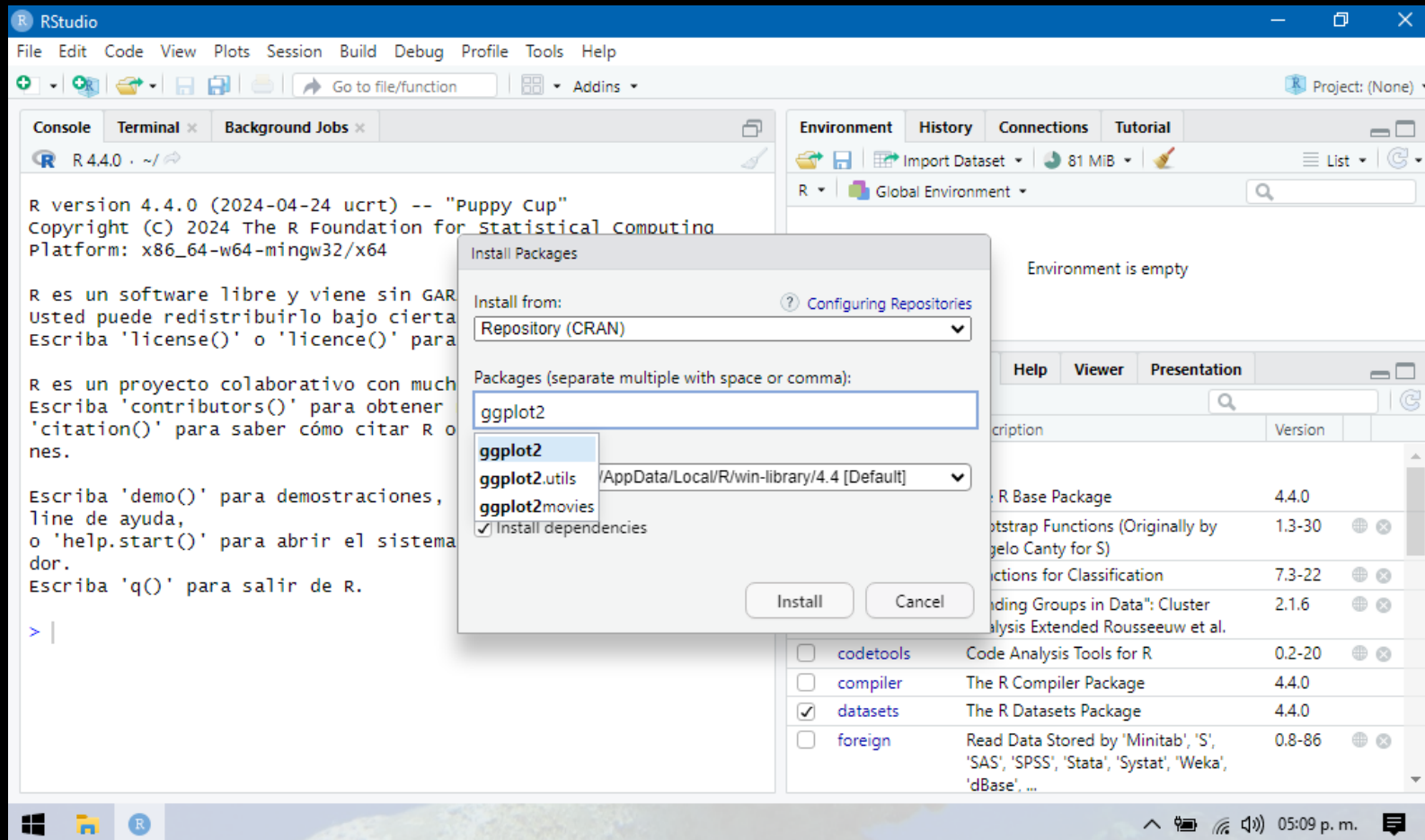
> `install.packages("ggplot2")`

Instalación de paquetes

- Alternativamente se pueden instalar desde el multipanel o desde la pestaña Tools, Install packages.

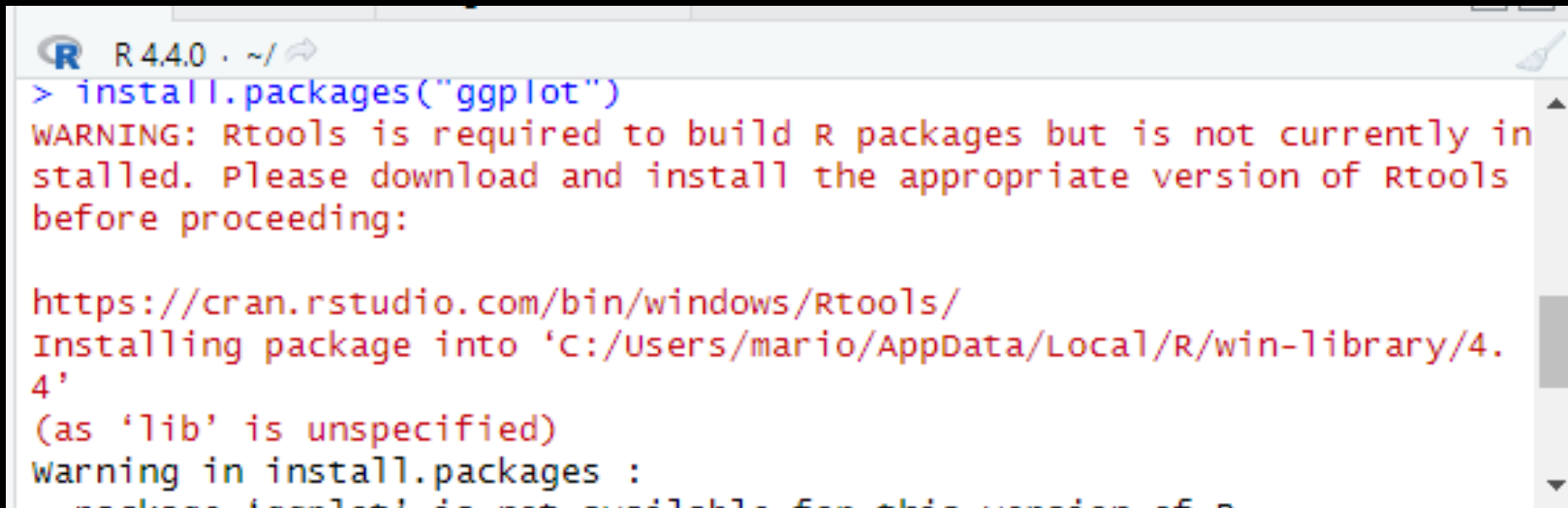


- Adicionalmente ofrece instalar **dependencias**.



Instalación de paquetes

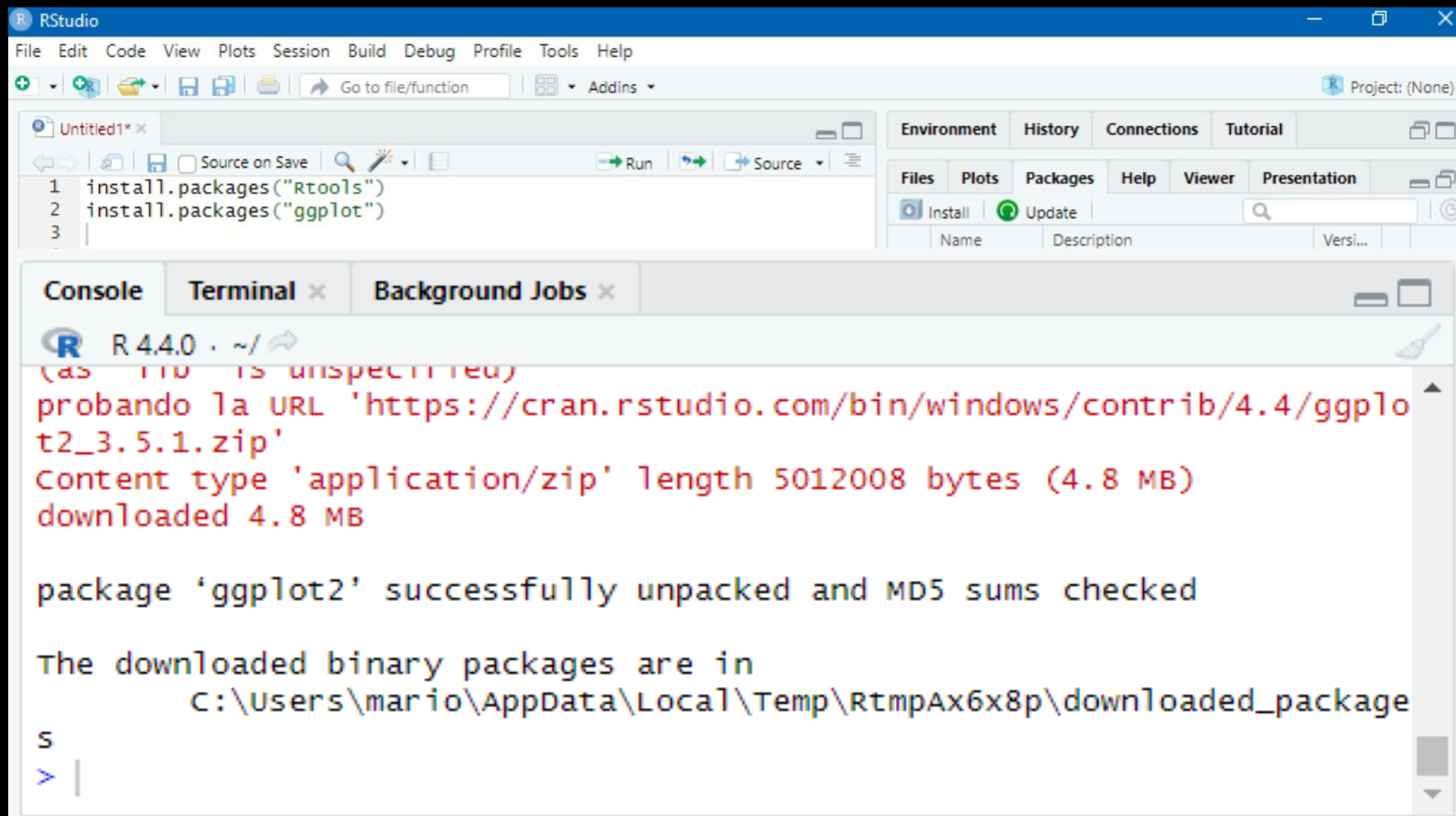
- Esto último es particularmente útil si vemos errores como este:

A screenshot of an R console window. The title bar shows 'R 4.4.0 . ~/'. The console displays the command '> install.packages("ggplot2")' in blue. Below it, a warning message in red text states: 'WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before proceeding:'. This is followed by a URL in red: 'https://cran.rstudio.com/bin/windows/Rtools/'. Then, it says 'Installing package into 'C:/Users/mario/AppData/Local/R/win-library/4.4'' in red, followed by '(as 'lib' is unspecified)' in red. The final line shows 'warning in install.packages : ' in red, followed by a partially visible line in red: 'package 'ggplot2' is not available for this version of R'.

```
R 4.4.0 . ~/  
> install.packages("ggplot2")  
WARNING: Rtools is required to build R packages but is not currently in  
stalled. Please download and install the appropriate version of Rtools  
before proceeding:  
  
https://cran.rstudio.com/bin/windows/Rtools/  
Installing package into 'C:/Users/mario/AppData/Local/R/win-library/4.  
4'  
(as 'lib' is unspecified)  
warning in install.packages :  
package 'ggplot2' is not available for this version of R
```

Instalación de paquetes

- Si la instalación es correcta primero mostrará el proceso y después volverá el prompt.



The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains the code `install.packages("Rtools")` and `install.packages("ggplot2")`.
- Console:** Displays the output of the installation process for `ggplot2`. The text is as follows:
`R 4.4.0 ~/
(as rtd is unspecified)
probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.4/ggplot2_3.5.1.zip'
Content type 'application/zip' length 5012008 bytes (4.8 MB)
downloaded 4.8 MB

package 'ggplot2' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\mario\AppData\Local\Temp\RtmpAx6x8p\downloaded_packages
s
> |`
- Environment/History/Connections/Tutorial:** These panels are visible on the right side of the interface.

Library

- Funciones y material (datos de ejemplo) de un paquete de R se almacenan en una **library**.
- Ruta en computadora que aloja los paquetes deseados.
- Se cargan con:

```
> library("nombre de la library")
```



```
> library("ggplot2")
```

Gracias por su atención