

BSM

Blockchain School
for Management

Tema 1 - Introducción Módulo II - Programación en R

Nicolás Forteza

2022-11-07

R es un entorno y lenguaje de programación dedicado a la manipulación de datos, el cálculo y visualización de gráficos. Nace como software libre a partir del lenguaje *S-Plus*.

Código libre bajo los términos o licencia GNU: que puede ser desarrollado y distribuido de manera independiente y gratuita.

El código libre tiene muchas ventajas.

Ventajas y consecuencias del código libre

Ventajas

- Gratuito.
- Libertad de uso.
- Libertad de estudiar y modificar el código fuente.
- Libertad de redistribuir copias a los demás.

Consecuencias

En consecuencia, muchos usuarios (profesionales, investigadores, académicos, etc.) desarrollan librerías, que junto con el código fuente de base, conforman el ecosistema R (en mayor parte) que uno tiene instalado en su ordenador.

Pero ... ¿qué es el código fuente?

Descargar RStudio (más tarde volveremos a la explicación de qué es RStudio).

Enlace de descarga.

RStudio instala R y una interfaz gráfica para poder usarlo.

El código fuente de R se encuentra (en este caso en mi equipo) en la siguiente ruta:

```
[1] "C:/Program Files/R/R-language/library"
```

En esta ruta se encuentra el código fuente: lo conforman las librerías y el código fuente de base (*base* es la librería con las principales funciones de R).

Un *paquete* o *librería* es una colección de funciones, datos, código, etc... codificado en un lenguaje de programación, en este caso R, que se puede instalar de manera **libre** y gratuita para hacer uso de él, dadas las funcionalidades que ofrece.

Existen librerías que son esenciales para poder llevar a cabo determinadas tareas. R es capaz de usar otros lenguajes (C por ejemplo) para mejorar sus cualidades (*performance*).

Existen librerías con diversos fines: análisis estadístico, visualización de datos, matemática financiera, física, etc.

CRAN es el acrónimo de *Comprehensive R Archive Network*.

Es el lugar donde se encuentran todas las librerías que los usuarios desarrollan, es decir, es el lugar de la distribución del código R.

En la sección de *librerías* se pueden ver todas las librerías existentes en R. Cada librería de CRAN tiene que tener una documentación respectiva en PDF, que se puede ver en RStudio.

¿Qué es?

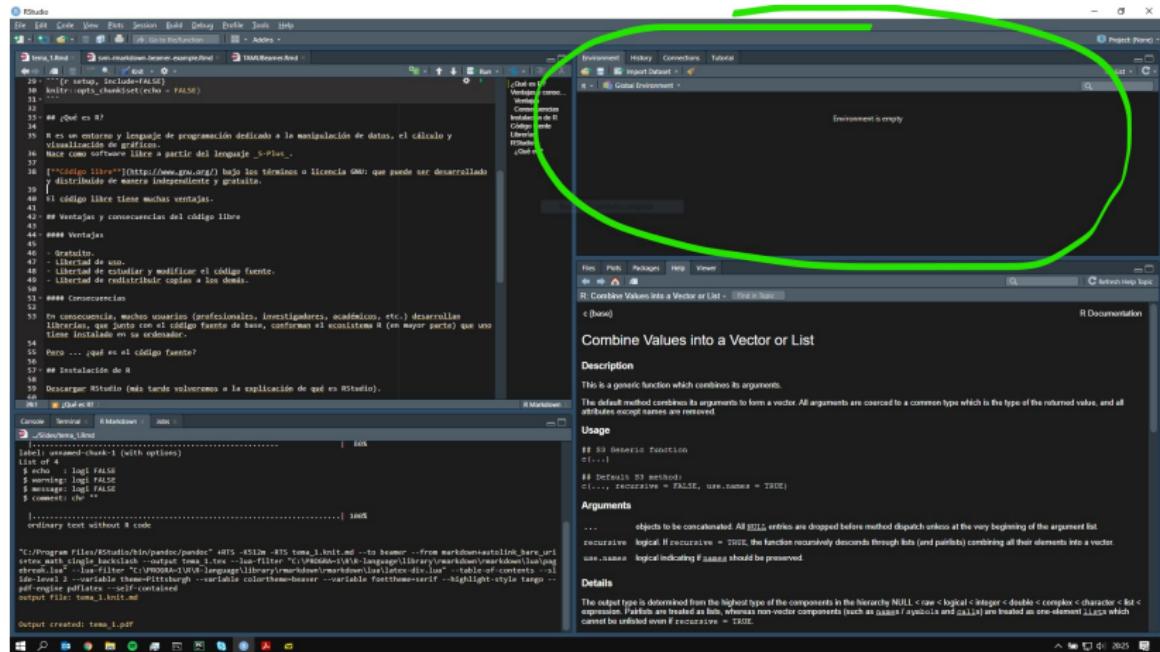
Es una interfaz gráfica.

¿A qué se parece?

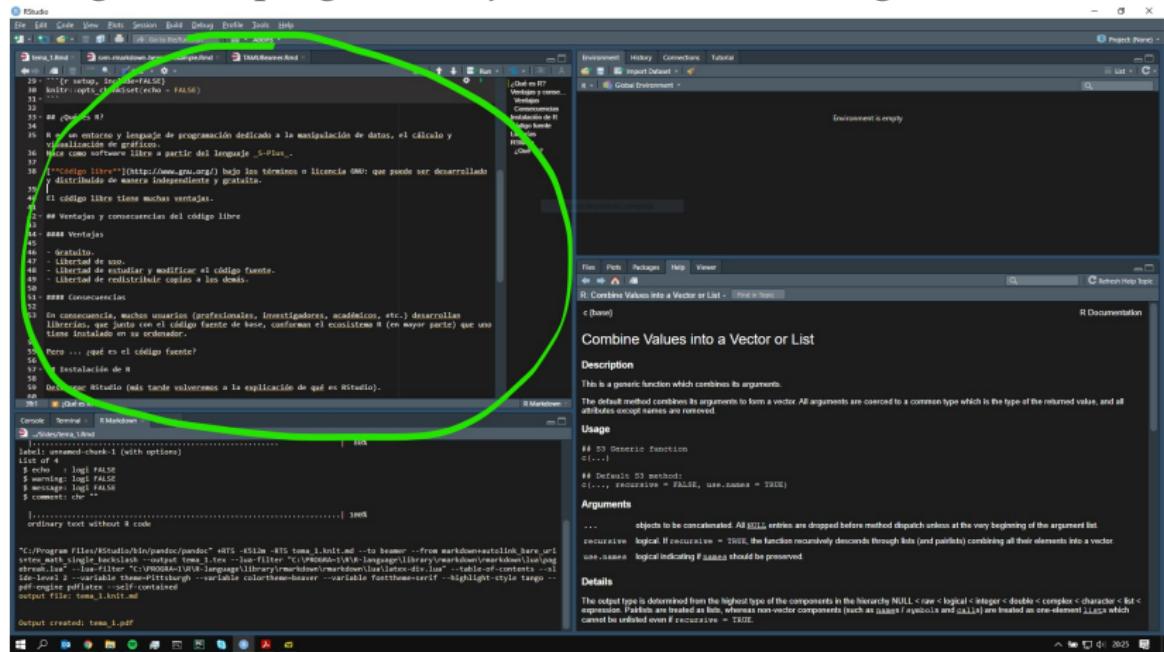
The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays R code from a file named `tma_1.Rmd`. The code includes setup instructions, a warning message about the use of `knitr`, and a note about the license. It also contains a section titled "#### Ventajas" and "#### consecuencias".
- Console:** Shows the command `knitr::knit("tma_1.md")` being run, followed by the output of the R code, which includes a warning about `knitr` and the generated `tma_1.html` file.
- Environment:** Shows the global environment is empty.
- Help:** A detailed help page for the `c` function is displayed, explaining its purpose, usage, arguments, and details about combining values into a vector or list.

En el apartado *environment* o *entorno* se pueden encontrar los datos o variables que tenemos cargadas en la memoria de la sesión. A mayor tamaño de los *objetos* almacenados en este lugar, mayor memoria RAM necesita el equipo.



En el visor de archivos y scripts se encuentra el espacio en el que podemos ir editando ficheros con extensión .R: desarrollar código, crear programas, ejecutar líneas de código ...



Partes de RStudio

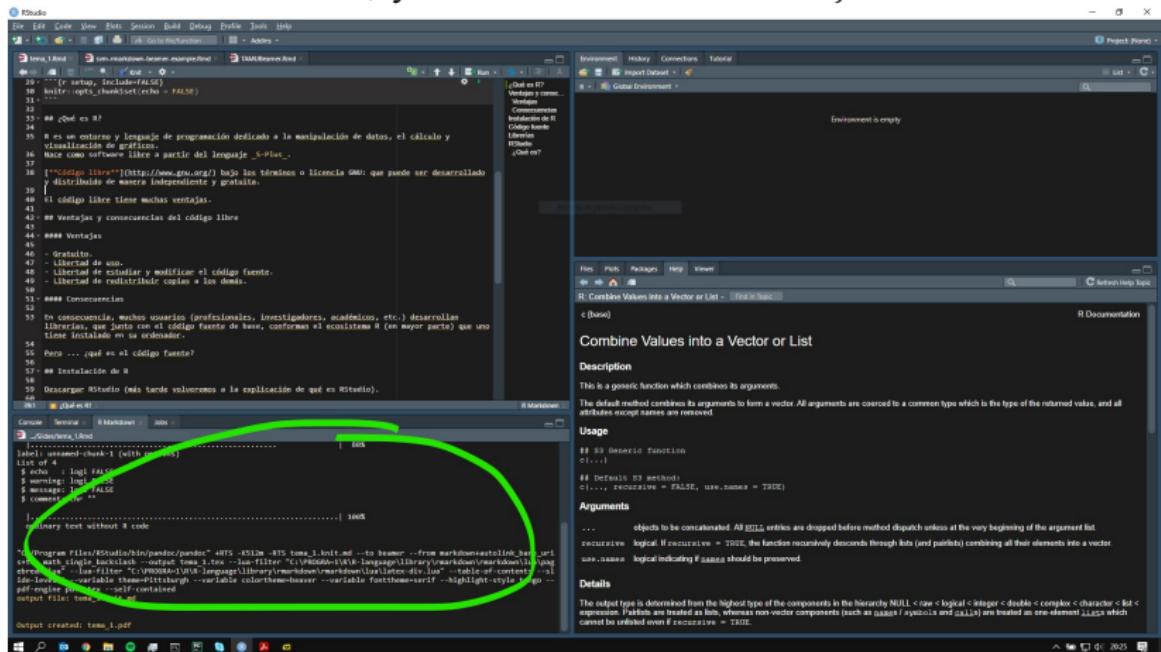
En el visor auxiliar podemos navegar por los directorios (volveremos a esto), ver gráficos, encontrar documentación de R.

The screenshot shows the RStudio interface with several panes open:

- Environment** pane: Shows the Global Environment tab with the message "Environment is empty".
- Help** pane: Shows the R: Combine Values into a Vector or List page. A large green circle highlights the title "Combine Values into a Vector or List" and the "Description" section.
- Documentation** pane: Shows the R Documentation page for the same topic, with a green circle highlighting the "Arguments" section.
- Code Editor** pane: Shows a script named "toma_1.Rnw" containing R code related to R setup, licensing, and freedom.
- Console** pane: Shows the command "knitr::knit('toma_1.tex')" being run.
- Output** pane: Shows the command and its execution results, including the creation of "toma_1.pdf".

Partes de RStudio

En la consola ejecutar código (bien desde la misma consola o desde un archivo .R) y ver el resultado de lo ejecutado.



Primer programa en R

Crear un *script* de R: archivo > nuevo archivo > R script.

Un *script* es un archivo en el que podemos codificar sentencias, para después ejecutarlo.

Escribimos lo siguiente: `print("Hello world!")`

El comando `print()` muestra un objeto en la consola.

Primer programa en R

`print()` es, en realidad, una función de R (ya llegaremos a esto).

Para ver qué argumentos recibe dicha función, pulsar F1 cuando el cursor está encima de la palabra `print` dentro de tu script.

Se abre en tu visor una réplica exacta del PDF de la documentación de esa función (en este caso, una función de R *base*, la librería básica).

En esta documentación vienen referencias y ejemplos muy útiles.

Para ejecutar el código, tenemos varias opciones:

- Ctrl + Enter (el cursor tiene que estar sobre la línea que queremos ejecutar en la consola).
- Ejecutar el script entero desde una consola de R.

El segundo método no se suele usar (depende de lo que se esté haciendo), pero consiste en lo siguiente:

- Guardamos archivo en una ruta determinada.
- Ejecutamos lo siguiente:

```
source("ruta/donde/se/encuentra/el/archivo")
```

Un directorio es una ruta, un espacio dentro de tu sistema de carpetas (*Finder* en mac, *Explorador de archivos* en Windows).

Tenemos que siempre tener claro en qué directorio estamos trabajando.

Para saber en qué directorio nos encontramos, ejecutar en la consola:

```
getwd()
```

El resultado de este comando te devuelve el *working directory*

-> ¿Por qué es importante manejar bien los directorios?

Tener el *working directory* en el lugar adecuado nos ayuda a:

- No tener fallos en cargas de datos.
- Organizar nuestro código y proyectos de análisis de datos.

Para fijar el *working directory* en una ruta que necesitamos:

```
# cambiar la ruta por la vuestra  
setwd("C:/Users/Q31984/Documents")
```

Ejercicio 1.1

- 1-. Crea una carpeta dentro de tu carpeta de este curso que se llame Tema 1.
- 2-. Crea un script de R.
- 3-. Codifica una sentencia que imprima el *working directory* en el que estás actualmente.
- 4-. Ejecuta la sentencia.
- 5-. Cambia tu working directory a la ruta del Tema 1.
- 6-. Ejecuta de nuevo la línea de código que escribiste.

Resultado del ejercicio

```
print(getwd())
setwd("C:/Users/Q31984/Documents") # no imprime nada
print(getwd())
```

- R es un lenguaje de programación de código libre.
- RStudio es la interfaz donde interactuamos con R.
- `print()` es una función que muestra en consola un objeto de R.
- Tener fijado el *working directory* correctamente es crucial en nuestro análisis de datos.

¿Preguntas?