

Fundamentos R

Las herramientas del científico de datos

Juan Manuel Moreno — jmmoreno@profesorescol.imf.com



ÍNDICE

1. Objetivos unidad 4
2. Fundamentos de R
3. Arrays
4. Listas
5. Factores
6. Matrices
7. Dataframes
8. Estructuras de control
9. Funciones
10. Anexo Operadores

01

Objetivos unidad 4

- Conocer qué es el lenguaje de programación R y habituarse al IDE de programación RStudio.
- Aprender sobre la sintaxis básica de R, desde la creación de scripts y notebooks en donde el alumno podrá realizar anotaciones en Markdown, realizar comentarios, conocer los principales operadores, hasta, implementar variables y conocer los diferentes tipos de variables que existen en R.
- Distinguir y saber utilizar las siguientes estructuras de datos en R: Vectores, arrays, factores, listas, matrices y dataframes.
- Saber controlar el flujo de un programa a través de sentencias condicionales y bucles.
- Desarrollar funciones propias, diferenciando entre parámetros de entrada y salida, así como utilizar funciones propias que integra R internamente.



02

Fundamentos R

2.– Fundamentos R

2.1.– El lenguaje de programación R

- Lenguaje interpretado, de código abierto.
- Especialmente enfocado a estadística y matemáticas.
- Surge en 1992 bajo el nombre de S.
- Curva de aprendizaje corta.
- Sintaxis básica.
- Mantenido por la comunidad de R.
- **CRAN** (The Comprehensive R Archive Network) <https://cran.r-project.org/>





[CRAN](#)
[Mirrors](#)
[What's new?](#)
[Search](#)
[CRAN Team](#)

[About R](#)
[R Homepage](#)
[The R Journal](#)

[Software](#)
[R Sources](#)
[R Binaries](#)
[Packages](#)
[Task Views](#)
[Other](#)

[Documentation](#)
[Manuals](#)
[FAQs](#)
[Contributed](#)

[Donations](#)
[Donate](#)

The Comprehensive R Archive Network

Download and Install R

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 (2023-10-31, Eye Holes) [R-4.3.2.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#)

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

Supporting CRAN

- CRAN operations, most importantly hosting, checking, distributing, and archiving of R add-on packages for various platforms, crucially rely on technical, emotional, and financial support by the R community.

Please consider making [financial contributions](#) to the R Foundation for Statistical Computing.

What are R and CRAN?

2.2.– Instalación de R

- Para la instalación de Rstudio, es necesario primero descargar el intérprete de R, ya que de lo contrario, no es posible trabajar con R Studio.
- La descarga se realiza desde Download and Install R <https://cran.r-project.org/>

Download and Install R

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](#)
- [Download R for \(Mac\) OS X](#)
- [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.

2.2.– Instalación de R

- Pulsamos en install R for the first time

Subdirectories:

base	Binaries for base distribution. This is what you want to install R for the first time .
contrib	Binaries of contributed CRAN packages (for R \geq 2.13.x; managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R $<$ 2.13.x; managed by Uwe Ligges).
Rtools	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the [R FAQ](#) and [R for Windows FAQ](#).

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

2.2.– Instalación de R

- Pulsamos en Download R.X.X.X for Windows, la descarga comenzará automáticamente

R-4.3.2 for Windows

[Download R-4.3.2 for Windows](#) (79 megabytes, 64 bit)

[README on the Windows binary distribution](#)

[New features in this version](#)

This build requires UCRT, which is part of Windows since Windows 10 and Windows Server 2016. On older systems, UCRT has to be installed manually from [here](#).

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the [md5sum](#) of the .exe to the [fingerprint](#) on the master server.

Frequently asked questions

- [Does R run under my version of Windows?](#)
- [How do I update packages in my previous version of R?](#)

Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for Windows-specific information.

Other builds

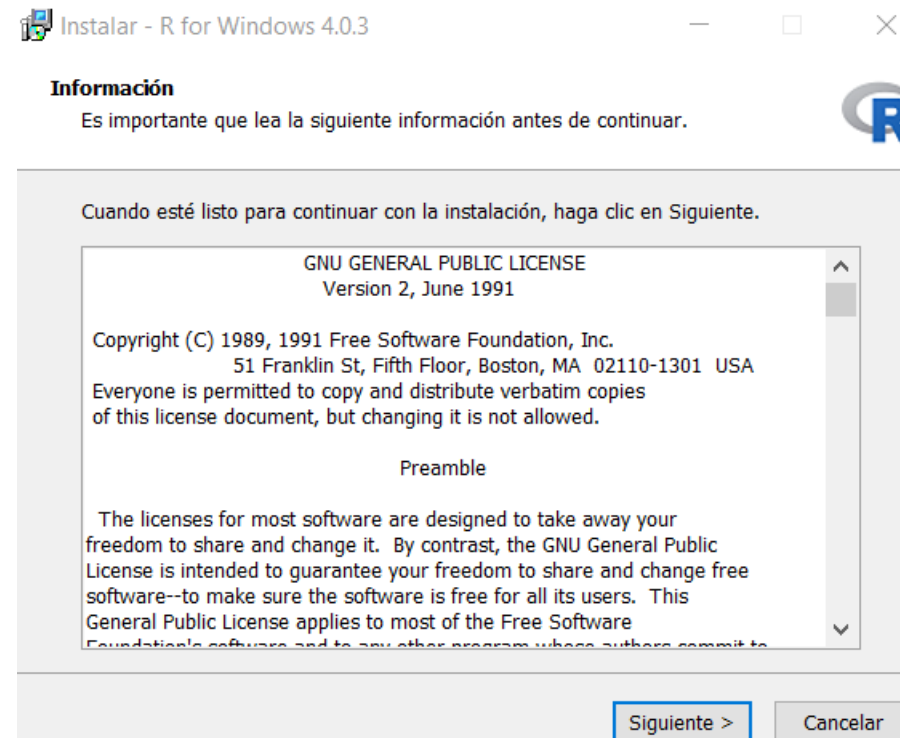
- Patches to this release are incorporated in the [r-patched snapshot build](#).
- A build of the development version (which will eventually become the next major release of R) is available in the [r-devel snapshot build](#).
- [Previous releases](#)

Note to webmasters: A stable link which will redirect to the current Windows binary release is
<http://<CRAN MIRROR>/bin/windows/base/release.html>.

2.– Fundamentos R

2.2.– Instalación de R

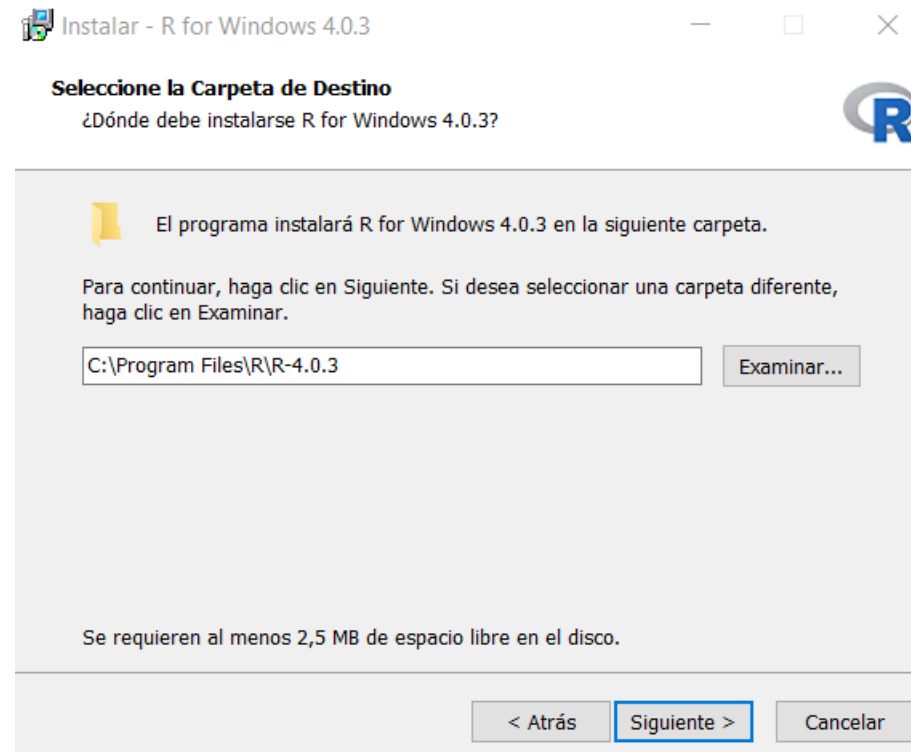
- Ejecutamos el instalable, seleccionamos el idioma, pulsamos en siguiente tras revisar la información sobre la función GNU.



2.– Fundamentos R

2.2.– Instalación de R

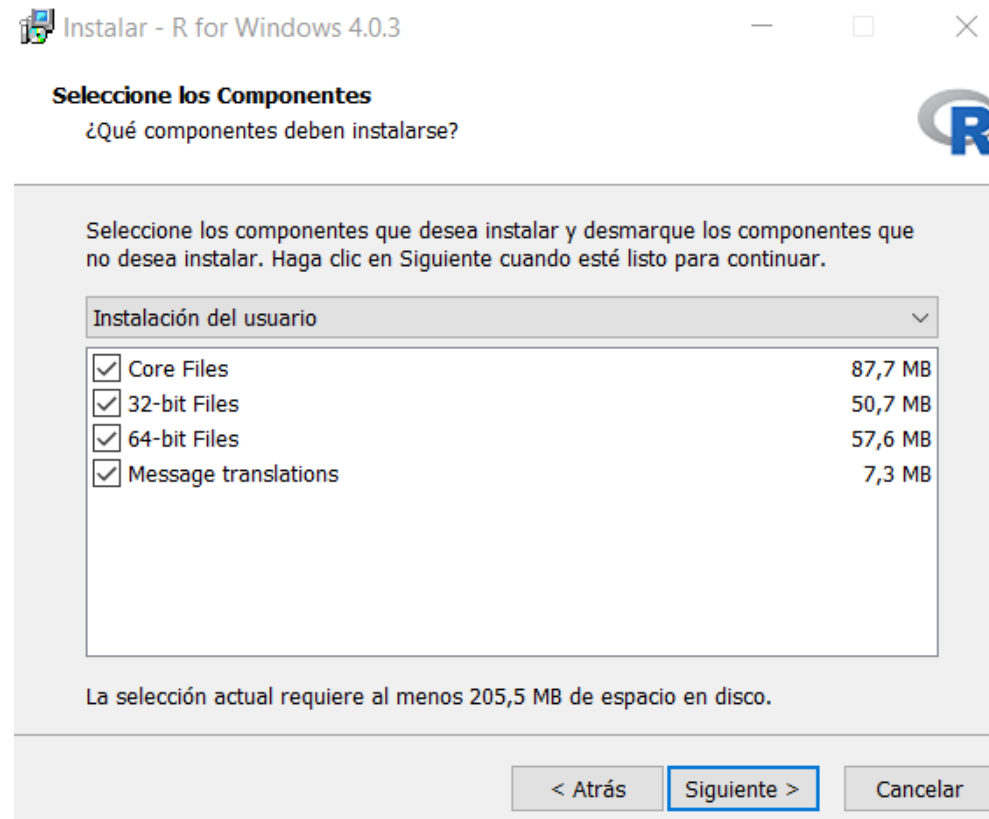
- Dejamos por defecto o modificamos la ruta de instalación de R.



2.- Fundamentos R

2.2.- Instalación de R

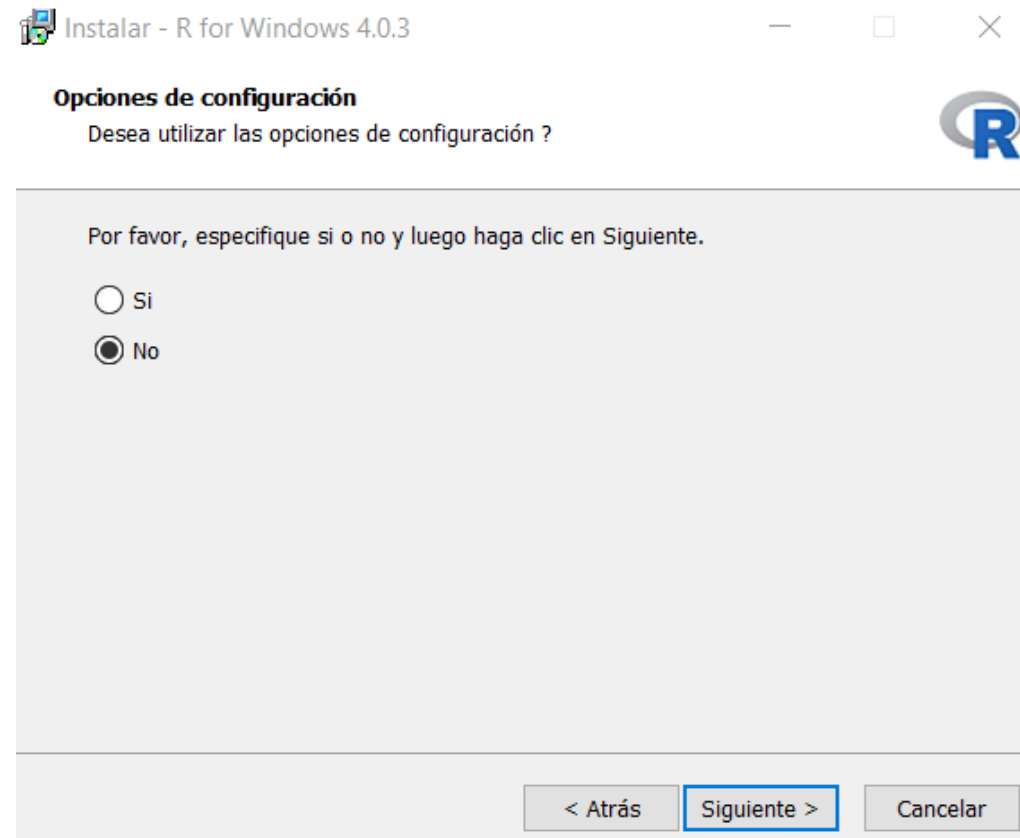
- Pulsamos en siguiente en la instalación de componentes de usuario.



2.- Fundamentos R

2.2.- Instalación de R

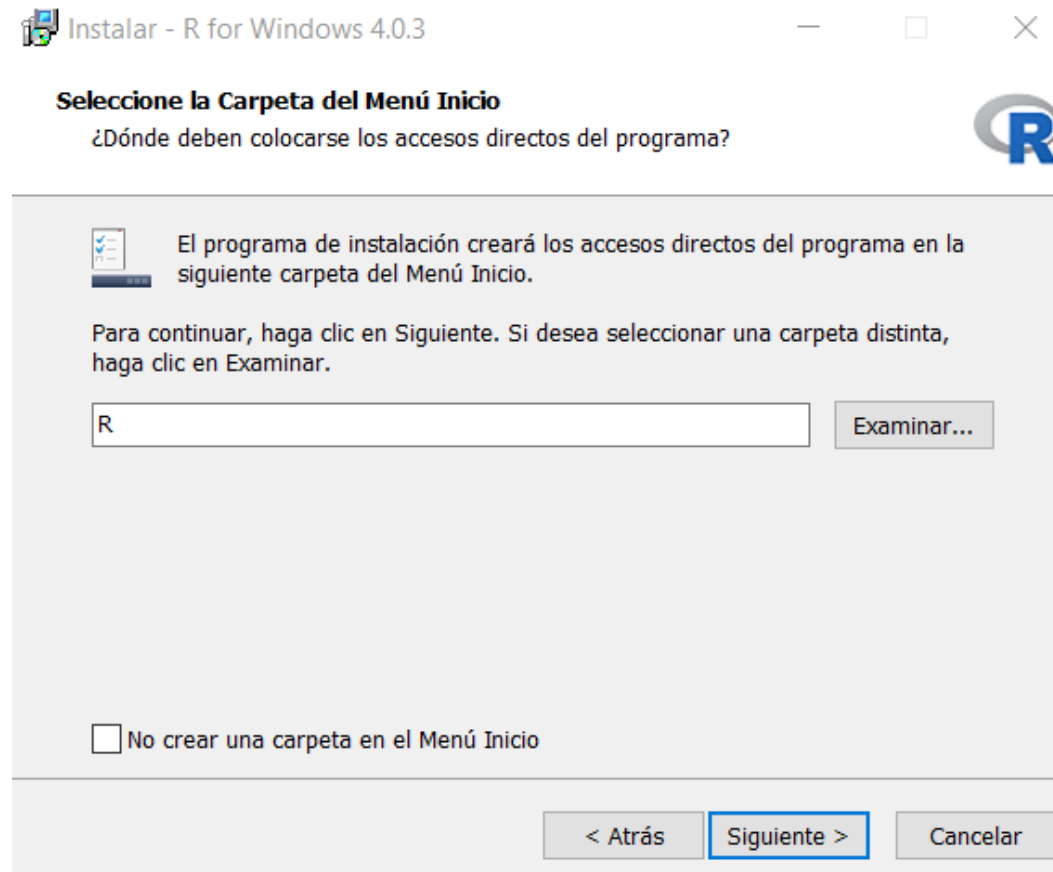
- Pulsamos en No en las opciones de configuración.



2.– Fundamentos R

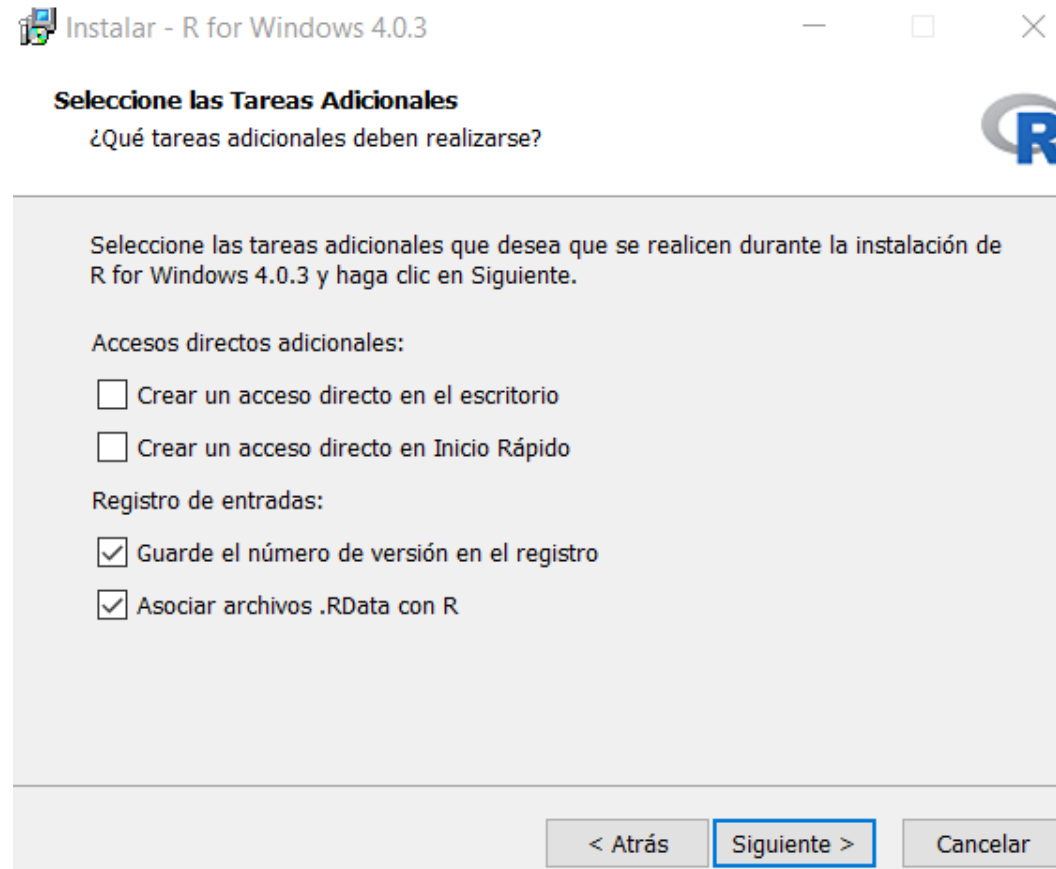
2.2.– Instalación de R

- Dejamos por defecto o cambiamos la carpeta del menú de inicio de R.



2.2.– Instalación de R

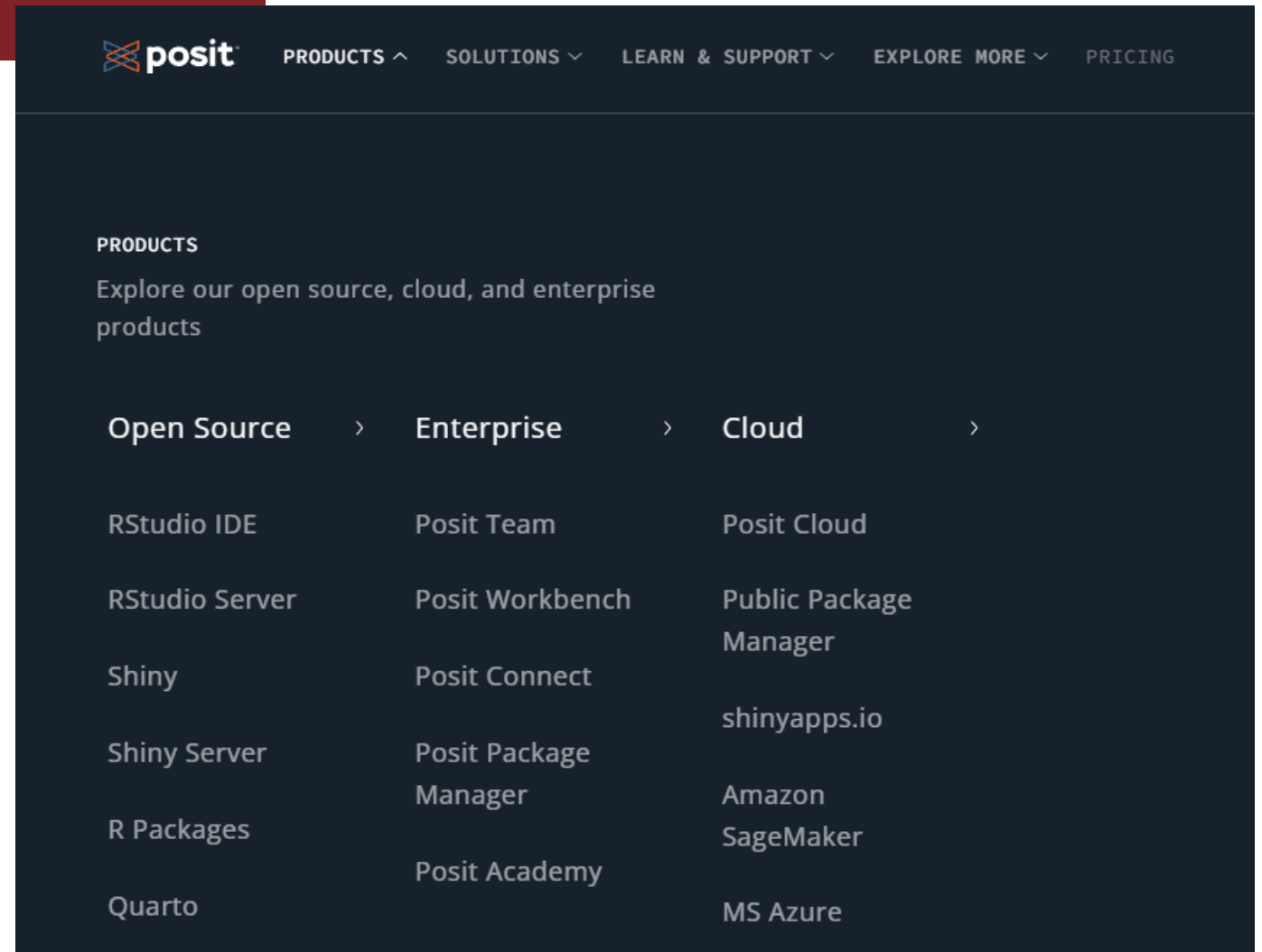
- Dejamos por defecto las siguientes opciones, no hace falta crear accesos directos, no vamos a utilizar la versión más primitiva de R.



2.– Fundamentos R

2.3.– Instalación de RStudio

- En la web <https://posit.co/> vamos a la sección Products – Rstudio IDE.



2.– Fundamentos R

2.3.– Instalación de RStudio

- Escogemos la versión Rstudio Desktop (Open Source Edition).

<p>Open Source Edition</p> <p>The Premier IDE for R</p> <p>Pricing</p> <p>Free</p> <p>DOWNLOAD RSTUDIO DESKTOP</p> <ul style="list-style-type: none">✓ Access the RStudio IDE locally✓ Syntax highlighting, code completion, and smart indentation	<p>RStudio Desktop Pro</p> <p>The RStudio IDE, superpowered for your professional workflow</p> <p>\$1,045 per year</p> <p>BUY NOW</p> <p>ALL OF THE FEATURES OF OPEN SOURCE; PLUS:</p>
---	---

2.3.– Instalación de RStudio

- Pulsamos Download Rstudio.

DOWNLOAD

RStudio IDE

The most popular coding environment for R, built with love by Posit.

Used by millions of people weekly, the RStudio integrated development environment (IDE) is a set of tools built to help you be more productive with R and Python. It includes a console, syntax-highlighting editor that supports direct code execution. It also features tools for plotting, viewing history, debugging and managing your workspace.

If you're a professional data scientist and want guidance on adopting open-source tools at your organization, don't hesitate to [book a call with us.](#)

DOWNLOAD RSTUDIO

DOWNLOAD RSTUDIO SERVER

2.3.– Instalación de RStudio

- El primer paso, ya lo tendríamos, vamos al segundo.

1: Install R

RStudio requires R 3.3.0+. Choose a version of R that matches your computer's operating system.

[DOWNLOAD AND INSTALL R](#)

2: Install RStudio

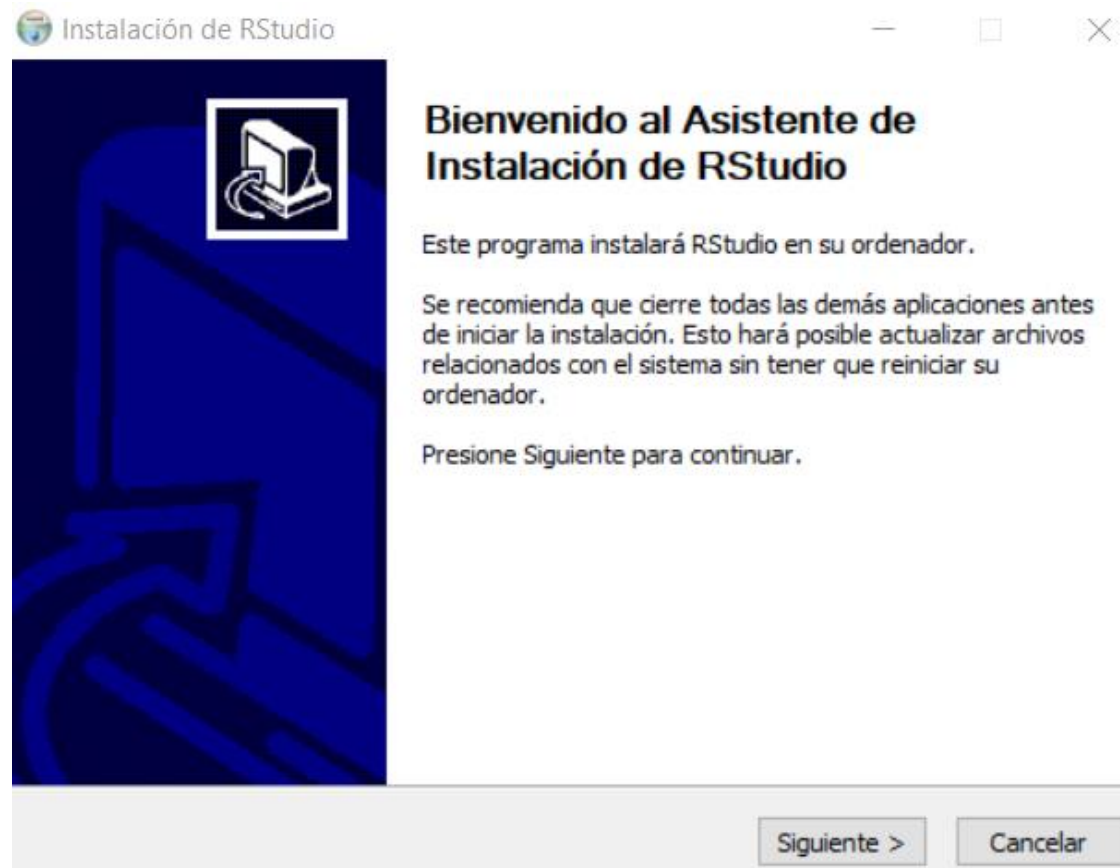
[DOWNLOAD RSTUDIO DESKTOP FOR WINDOWS](#)

Size: 214.34 MB | [SHA-256: FE62B784](#) | Version: 2023.09.1+494 |
Released: 2023-10-17

2.– Fundamentos R

2.3.– Instalación de RStudio

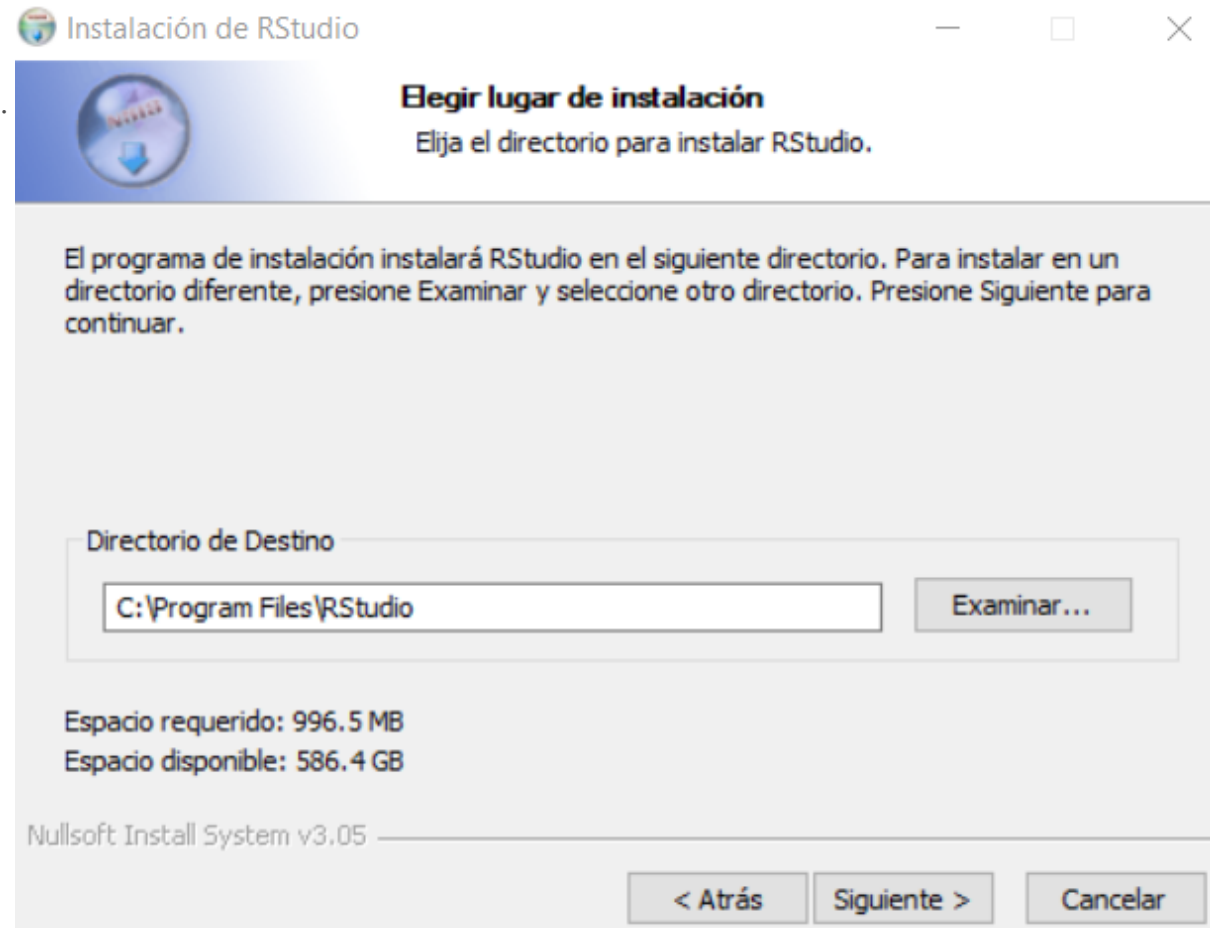
- Abrimos el asistente de instalación, pulsamos siguiente.



2.– Fundamentos R

2.3.– Instalación de RStudio

- Seleccionamos la ruta de instalación.



2.– Fundamentos R

2.3.– Instalación de RStudio

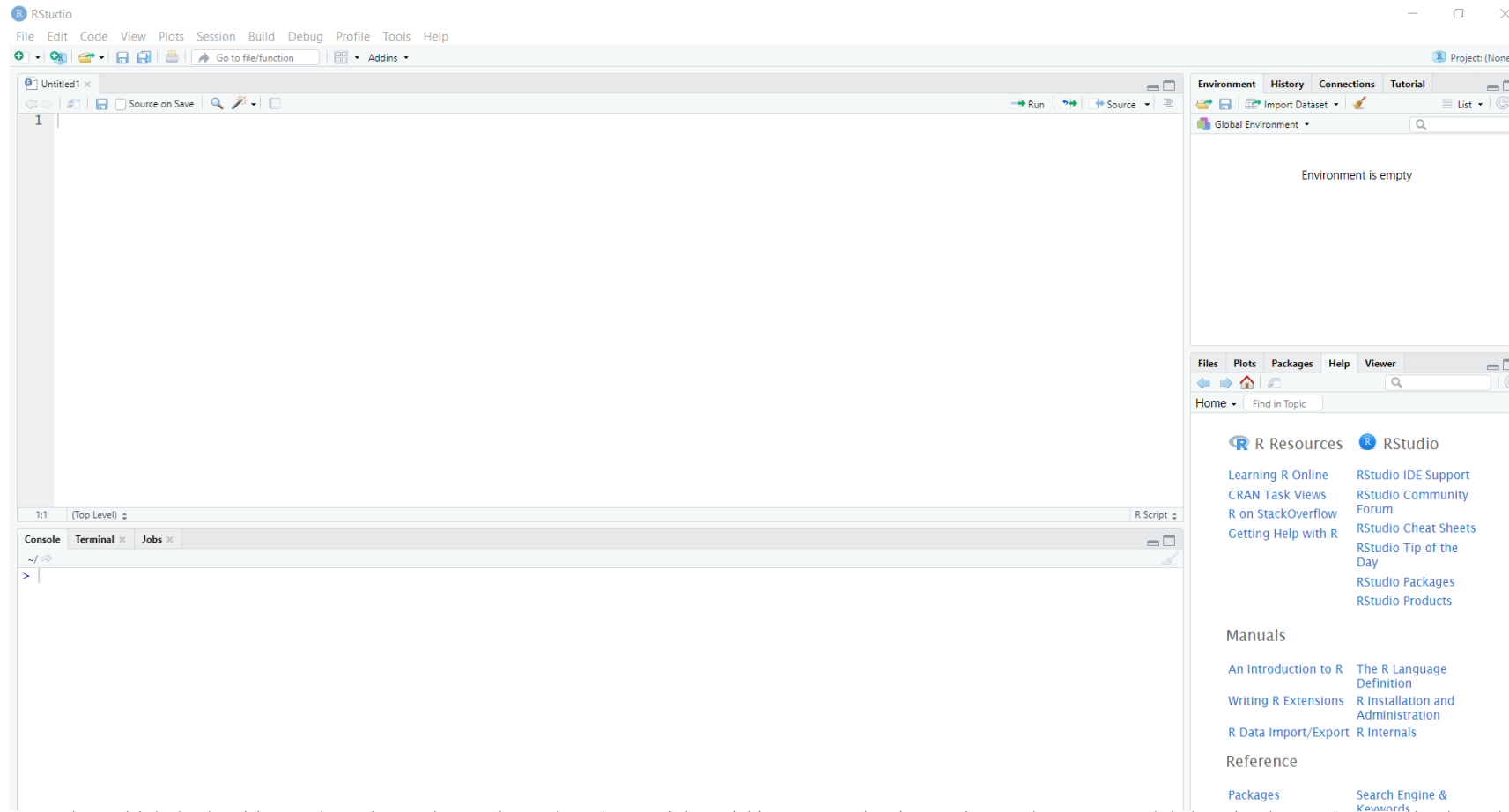
- Seleccionamos la carpeta del menú de inicio y pulsamos en Instalar.



2.– Fundamentos R

2.4.– RStudio

- En primera instancia aparecerá una ventana como esta en la que podremos ver



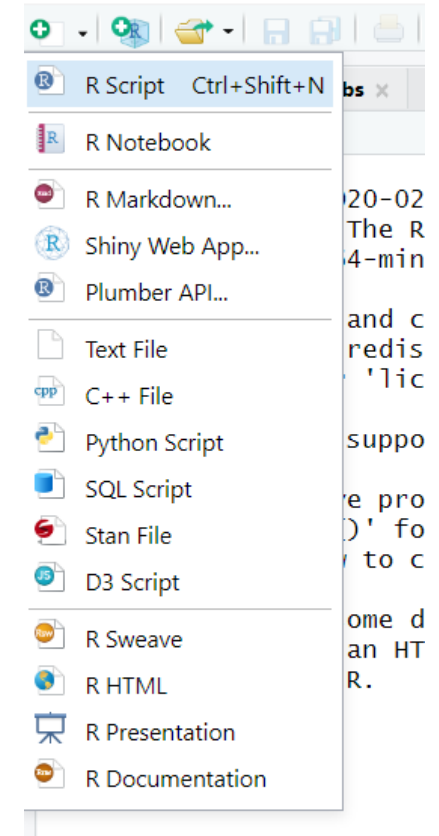
2.4.– RStudio

- El IDE de Rstudio se compone de cuatro cuadrantes principales.
 - **Zona de scripting:** (Superior izquierda). Donde escribimos el código fuente tanto en scripts (.R) como notebooks (.RMD)
 - **Zona de entorno (environment) e historial de comandos:** (Superior derecha): Cada vez que declaramos una variable, aparece en el entorno del sistema, también podemos acceder a todos los comandos que hayamos introducido.
 - **Zona de terminal:** (Inferior izquierda): Podemos tanto ejecutar código fuente como, ver la salida del código de los scripts y notebooks que estemos ejecutando.
 - **Zona de archivos, paquetes, ayuda y gráficos:** (Inferior derecha): Podemos ver tanto nuestro propio sistema de archivos, paquetes cargados en la sesión, gráficas que estemos ejecutando y, la ayuda de una función seleccionada.

2.– Fundamentos R

2.4.– Rstudio — Primeros pasos

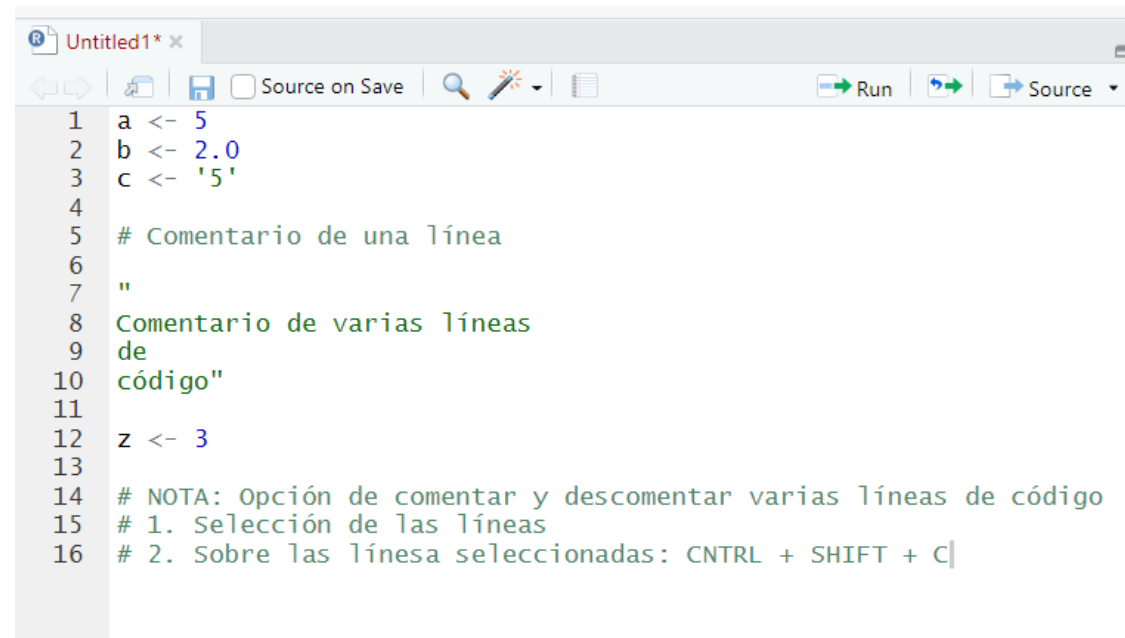
- Para lanzar un notebook o un script acudimos al botón New File...
- Seleccionamos R Script.
- Automáticamente apareceremos en la sección de scripting.
- Declaramos algunas variables, es importante recalcar que, en lugar de Python, en R, las variables se designan de forma direccional, con la siguiente nomenclatura `<-` o `->`, el símbolo igual, solamente se utilizará para pasar parámetros en funciones.
 - `a <- 5`
 - `b <- 2.0`
 - `c <- 'Hola Mundo'`



2.– Fundamentos R

2.4.– Rstudio — Primeros pasos

- Podemos escribir un comentario a través del operador `#` o de varias líneas mediante comillas dobles
" ... "
- Para comentar/descomentar varias líneas utilizaremos `CNTRL + SHFT + C`
- Para ejecutar cualquier línea de código o varias líneas seleccionadas pulsamos `CNTRL + INTRO`

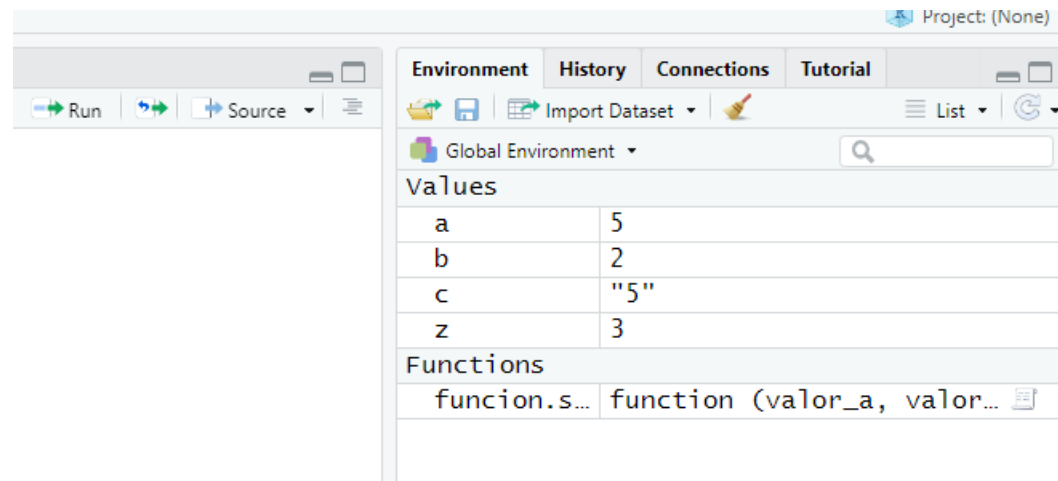
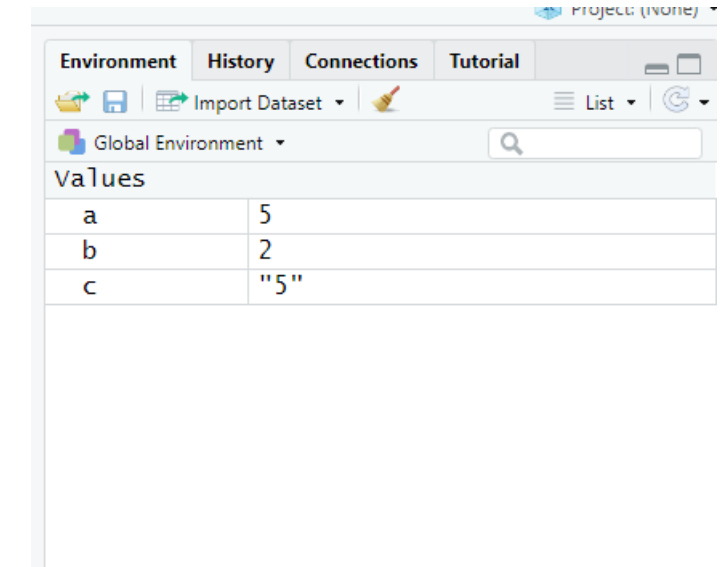


```
1 a <- 5
2 b <- 2.0
3 c <- '5'
4
5 # Comentario de una línea
6
7 "
8 Comentario de varias líneas
9 de
10 código"
11
12 z <- 3
13
14 # NOTA: Opción de comentar y descomentar varias líneas de código
15 # 1. Selección de las líneas
16 # 2. Sobre las líneas seleccionadas: CNTRL + SHFT + C
```

2.– Fundamentos R

2.4.– Rstudio — Primeros pasos

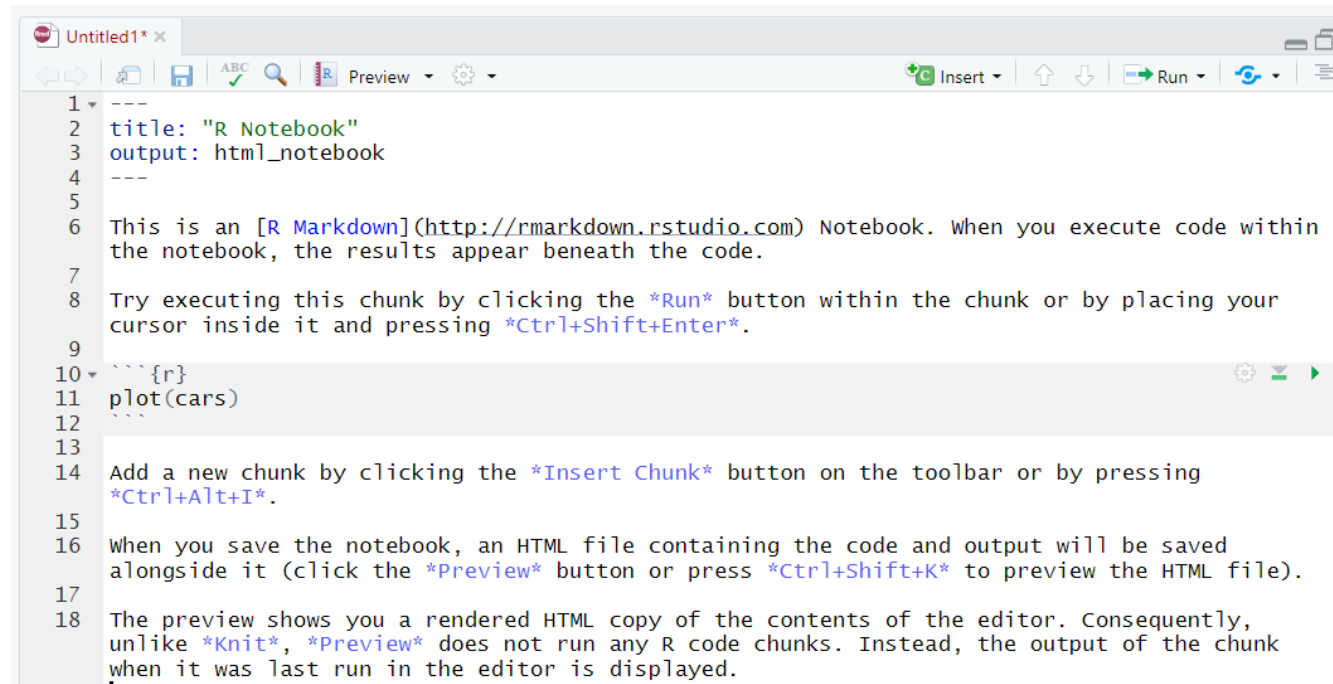
- Cualquier variable que declaremos en el sistema aparecerá en las variables de entorno declaradas en la sesión de R.
- Todo un script puede ejecutarse pulsando sobre el botón source y veremos ' como aparecen las variable declaradas en el environment.
- Si pulsamos sobre el botón con una brocha, podremos limpiar todas las variables declaradas (liberaremos memoria RAM).



2.- Fundamentos R

2.4.- Rstudio — Primeros pasos

- Podemos guardar tanto un notebook como un script mediante 'Save with encoding' (dejamos por defecto la codificación)
- Abrimos un R Notebook, aparecerá la siguiente ventana.

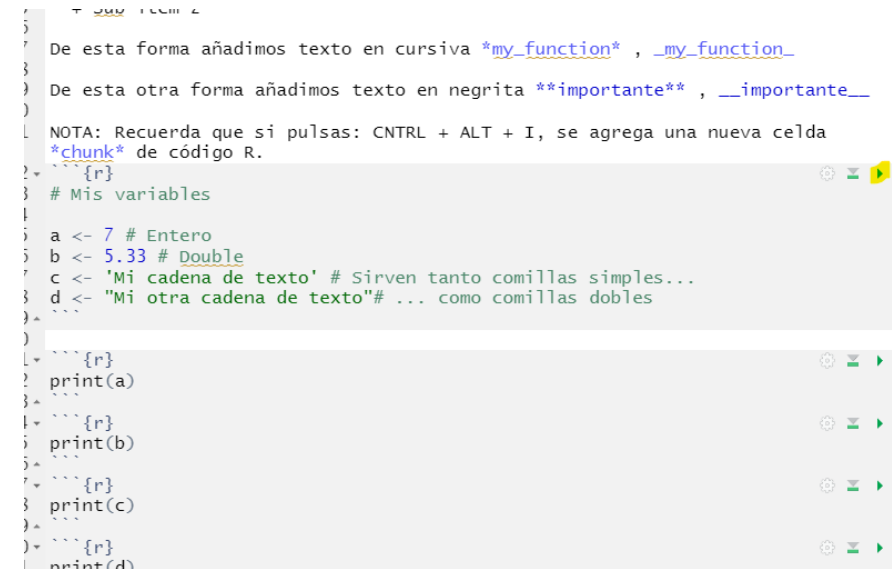


```
1 ---
2 title: "R Notebook"
3 output: html_notebook
4 ---
5
6 This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute code within
7 the notebook, the results appear beneath the code.
8
9 Try executing this chunk by clicking the Run button within the chunk or by placing your
10 cursor inside it and pressing Ctrl+Shift+Enter.
11
12 ```{r}
13 plot(cars)
14 ```
15
16 Add a new chunk by clicking the Insert Chunk button on the toolbar or by pressing
17 Ctrl+Alt+I.
18
19 When you save the notebook, an HTML file containing the code and output will be saved
20 alongside it (click the Preview button or press Ctrl+Shift+K to preview the HTML file).
21
22 The preview shows you a rendered HTML copy of the contents of the editor. Consequently,
23 unlike Knit, Preview does not run any R code chunks. Instead, the output of the chunk
24 when it was last run in the editor is displayed.
```

2.– Fundamentos R

2.4.– Rstudio — Primeros pasos

- Toda la información que aparece en un notebook en blanco puede ser eliminada, a diferencia de Jupyter, no tenemos que definir el tipo de la celda, si no que, todo lo que vaya fuera de una celda de código se interpretará como Markdown.
- Para agregar una nueva celda podemos pulsar CNTRL + ALT + I
- Para ejecutar una celda de código pulsaremos el botón 'Play' en verde o, seleccionando las líneas de código y CNTRL + INTRO



```
## Tarea 1: Primeros pasos
# De esta forma añadimos texto en cursiva *my_function* , _my_function_
# De esta otra forma añadimos texto en negrita **importante** , __importante__
# NOTA: Recuerda que si pulsas: CNTRL + ALT + I, se agrega una nueva celda
# *chunk* de código R.
{r}
# Mis variables
a <- 7 # Entero
b <- 5.33 # Double
c <- 'Mi cadena de texto' # Sirven tanto comillas simples...
d <- "Mi otra cadena de texto"# ... como comillas dobles
{r}
print(a)
{r}
print(b)
{r}
print(c)
{r}
print(d)
```

2.– Fundamentos R

2.4.– Rstudio — Primeros pasos

- Lo más normal, es guardar los notebooks para ser vistos como html, pdf o doc, para ello, vamos al botón Knit (previamente debemos haber guardado el archivo) y, pulsaremos la salida que queramos.

R Notebook

Juan Manuel Moreno

MI PRIMER NOTEBOOK

FUNDAMENTOS DE R

Elementos de Markdown

Esto, se muestra como una cadena de texto simple.

Ahora agreagamos una lista de elementos:

- Primer elemento
- Segundo elemento
- Tercer elemento

Ahora realizamos una lista numerada:

1. Uno
2. Dos
3. Tres

R Notebook

Juan Manuel Moreno

MI PRIMER NOTEBOOK

FUNDAMENTOS DE R

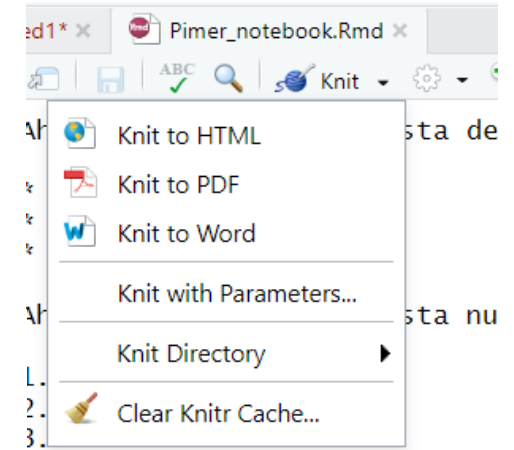
Elementos de Markdown

Esto, se muestra como una cadena de texto simple.

Ahora agreagamos una lista de elementos:

- Primer elemento
- Segundo elemento
- Tercer elemento

Ahora realizamos una lista numerada:



2.4.– Rstudio — Primeros pasos

- Podemos guardar una sesión de trabajo en la que queden almacenadas en memoria las variables que hayamos declarado, para ello, pulsamos sobre Sesión – Save Workspace As
- Revisar otras opciones como Restart R and clear outputs. Restar R and run all chunks

2.6– Estructuras de datos

- Principalmente trabajaremos en R con las siguientes estructuras de datos:
 - Vectores
 - Arrays
 - Listas
 - Factores
 - Matrices
 - Dataframes

2.7 – Vectores

- Sucesión de elementos de una dimensión.
- Diferentes tipos de datos en un vector.
- Se designan con la función `c()`
- Se pueden indexar
- IMPORTANTE: Los índices en R van de 1 a n, la posición 0 designa el tipo del vector, si tiene varios tipos, simplemente tomará el tipo del dato en primera posición
- Una operación afecta a todos los elementos de un vector.

2.7 — Funciones sobre vectores

- **length**: Longitud de un vector
- **sum**: Suma de los elementos de un vector
- **prod**: Producto de los elementos de un vector
- **min**: Elemento mínimo del vector
- **max**: Elemento máximo del vector.
- **range**: Devuelve el rango del vector, de su elemento min a max
- **mean**: Media del vector.
- **var**: Varianza del vector
- **sd**: Desviación estándar.
- **cumsum**: Suma acumulada elemento a elemento
- **cumprod**: Producto acumulado elemento a elemento.
- **sqrt**: Raíz de todos los elementos del vector.

2.7 — Funciones para crear vectores

- Rango personalizado de n a m
- Función seq (inicio, fin, elemento de división)
- Función rep (podemos incluir secuencias para repetir)
- Función sample (secuencia, número de elementos, repetición o no)
- Filtrar con booleanos.
- Para ordenar un vector utilizaremos la función sort



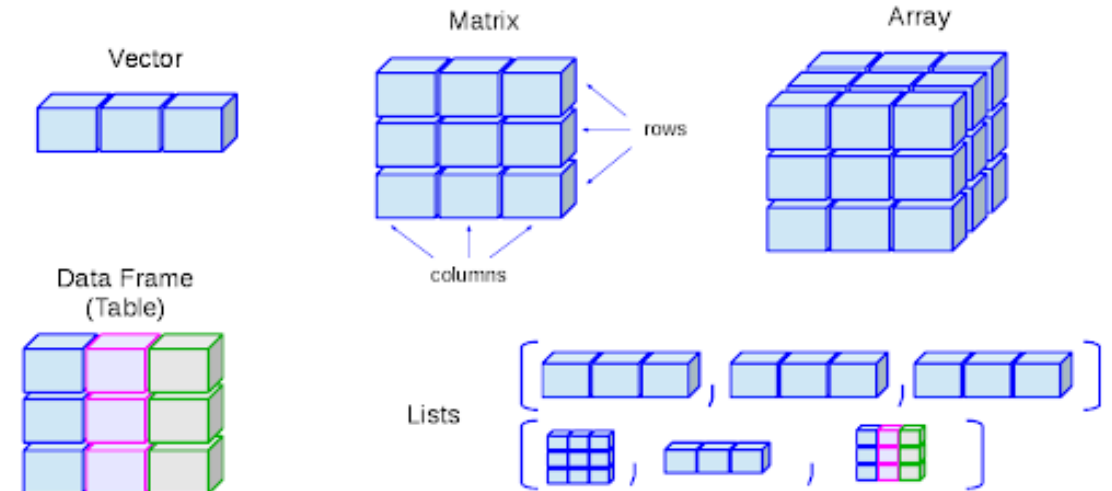
03

Arrays

3.- Arrays

3.1.- Arrays

- Todos sus elementos deben ser del mismo tipo.
- Son indexables
- Se emplea la función **array**.
- Para fijar la longitud de los elementos de las dimensiones de un array se emplea el parámetro **dim**.
- Un array se puede redimensionar, asignando un vector como nuevas dimensiones (en forma de filas x columnas).
- La estructura será la siguientes (**filas**, **columnas**, n **dimensión**)



3.2.– Funciones sobre Arrays

- Cuando seleccionemos una fila o, una columna o, un conjunto de valores de un array, lo que tendremos será un vector, por lo que podremos aplicar cualquier función vista sobre vectores.
- Algunas funciones específicas sobre arrays son:
 - **colSums**: Suma de las columnas de la dimensión n.
 - **rowSums**: Suma de las filas de la dimensión n.
 - **colMeans**: Media de las columnas de la dimensión n.
 - **rowMeans**: Media de las filas de la dimensión n.
 - **aperm**: Permutar la dimensiones de un array.
 - **class**: Devuelve la clase `array`




04

Listas

4.1.– Listas

- Son similares a los vectores.
- Colección de objetos (similar a una columna de un dataframe).
- Se pueden indexar por `[]` o mediante `[[]]`.
- Para acceder a un objeto se puede, acceder desde el operador `$nombre_objeto` o, `[nombre_objeto]`
`[[nombre_objeto]]`
- Para crear una lista se utiliza la función `list()`
- En función del tipo de dato que tengamos almacenado en la lista, podemos aplicar cualquier función de los vectores.



05

Factores


5.1.– Factores

- Tipo especial de vector utilizando exclusivamente para variables categóricas.
- Para crear un vector categórico se emplea **factor()** como parámetro recibe un vector.
- Para conocer el nombre de las categorías (factores), se emplea la función **levels**.
- Si queremos eliminar una categoría utilizamos la función **droplevels**.
- Cualquier vector se puede transformar en categórico con la función **as.factor**

```
var.cat <- factor(c("ENERO", "FEBRERO", "MARZO", "ABRIL"))  
var.cat
```

```
## [1] ENERO  FEBRERO MARZO  ABRIL  
## Levels: ABRIL ENERO FEBRERO MARZO
```

```
var.cat = droplevels(x = var.cat, "MARZO")  
var.cat  
## [1] ENERO  FEBRERO <NA>  ABRIL  
## Levels: ABRIL ENERO FEBRERO
```



06

Matrices

6.1.– Matrices

- Similar a una dimensión de una array en la que nos encontramos una matriz (es decir, filas y columnas)
- Para designar una nueva matriz, utilizamos la función **matrix**.
- Para designar el número de filas el parámetro **nrow** y, el de columnas, **ncol**.

Creamos una matriz.

```
my.mat <- matrix(1:30, ncol = 6, nrow = 5, byrow = F)
```

Probar byrow TRUE

```
my.mat
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    6   11   16   21   26
## [2,]    2    7   12   17   22   27
## [3,]    3    8   13   18   23   28
## [4,]    4    9   14   19   24   29
## [5,]    5   10   15   20   25   30
```


6.2.– Funciones sobre Matrices

- **ncol**: Devuelve el número de columnas
- **nrow**: Devuelve el número de filas
- **diag**: Devuelve la diagonal de la matriz
- **crossprod**: Realiza el producto entre dos matrices.
- **t**: Devuelve la traspuesta de una matriz.
- **svd**: Valores singulares de una matriz.
- **eigen**: Realiza el cálculo de autovalores y autovectores.
- **as.matrix**: Transforma un vector como matriz



07

Dataframes

7.- Dataframes

7.1.- Dataframes

- Similar a una hoja de Excel o, una base de datos SQL.
- Todos sus elementos tienen un índice común.
- Indexable por filas y columnas.
- Cada columna de un dataframe, se denomina también variable.
- Aceptan diferentes tipos de datos.
- Acceso mediante indexación `[]` o con el operador `$`

The diagram illustrates a DataFrame structure. At the top, the word "Columns" is written in blue, with four arrows pointing down to the column headers: "Name", "Team", "Number", "Position", and "Age". On the left side, the word "Rows" is written in orange, with four arrows pointing right to the row indices: 0, 1, 2, and 3. The table itself has a light green background and a thin black border. The data is as follows:

	<i>Name</i>	<i>Team</i>	<i>Number</i>	<i>Position</i>	<i>Age</i>
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

Additional annotations include a purple box around the "Jonas Jerebko" row, a purple box around the "8.0" value in the "Number" column, and a purple box around the "NaN" value in the "Age" column. A purple line connects these three boxes, with the word "Data" written in purple at the bottom right of the line. In the bottom right corner of the table area, there is a green logo consisting of two interlocking circles.

7.2.– Leyendo un csv como dataframe

- Además de crear dataframes de cero, trabajaremos principalmente con csv.
- Para leer un dataframe como csv se utiliza la función **read.csv** comúnmente con los siguientes parámetros:
 - **sep**: Tipo de separador de los datos (tabulador, coma, punto y coma, otros...)
 - **header**: Si el dataframe tiene cabecera o no.
 - **na.strings**: Indicar si hay un string que comúnmente se emplee como valor para designar valores nulos.

```
df <- read.csv("FB.csv")
```

7.3.– Funciones sobre dataframes


- **class**: Para saber la clase (tipo) del dataframe.
- **attach / detach**: Permite integrar todas las columnas del dataframe como variables sin necesidad de llamar al dataframe.
- **summary**: Muestra el resumen estadístico
- **str**: Devuelve el tipo de variables involucradas en el dataframe.
- **attributes**: Devuelve el nombre de las filas y columnas.
- **dim**: Dimensiones del dataframe en filas x columnas.
- **colnames**: Nombre de las columnas de un dataframe.
- **rownames**: nombre de las filas de un dataframe.
- **nrow**: N° filas.
- **ncol**: N° columnas.
- **head**: Primeras posiciones del dataframe.

7.4.– Funciones sobre dataframes

- **head**: Primeras posiciones del dataframe.
- **tail**: Últimas posiciones del dataframe.
- **rbind**: Añadir una nueva fila (También se puede por índice)
- **cbin**: Añadir una nueva columna (También se puede por índice)
- **<- NULL**: eliminar una variable
- **subset**: Obtener un subconjunto del dataframe
- **order**: Ordenar dataframe por una columna.
- **names**: Nombres de las columnas.
- **as.data.frame**: Convertir otra estructura de datos como dataframe.
- También es posible utilizar funciones de **arrays** y de **vectores**

7.5.– Gestión de nulos

- Comenzar con un resumen estadístico (**summary**).
- A través de la función **is.na()**
- Detectar las filas en las que hay valores nulos con **complete.cases**.
- **<- NA**: Se asigna como valor nulo.
- Para **borrar** nulos se puede realizar:
 - `nombre_df[complete.cases(nombre_df),]`
- Para **reemplazar** un valor nulo por otro valor:
 - `nombre_df $nombre_columna[which(is.na(nombre_df $ nombre_columna))] <- valor de reemplazo`



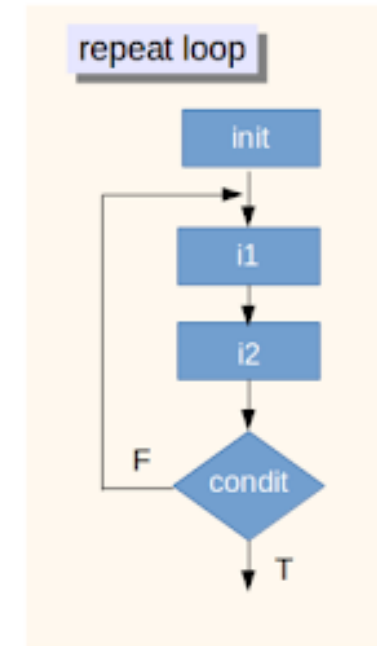
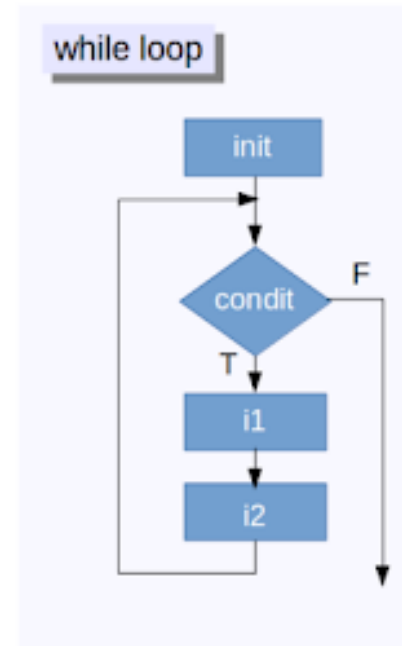
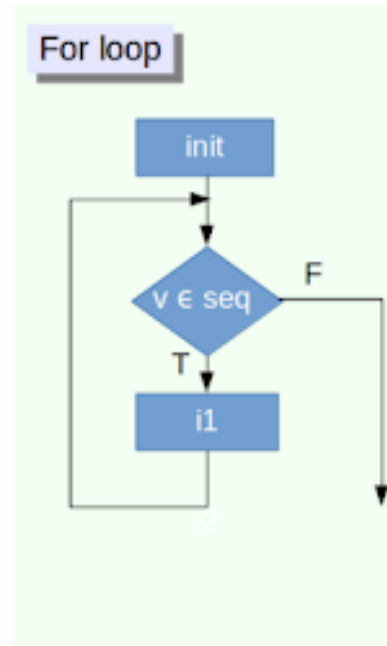
08

Estructuras de control

8.- Estructuras de control

8.1.- Estructuras de control

- Las estructuras de control en R, pueden dividirse en los siguientes bloques.
 - Condicionales (`if - else`)
 - Bucles (`for`, `while` y `repeat`)



8.– Estructuras de control

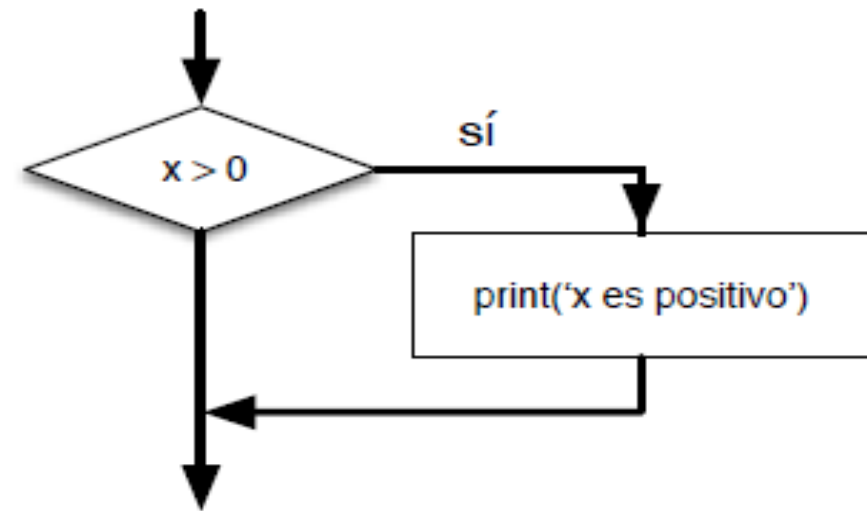
8.2.– Condicionales If

- Se utilizan para controlar la ejecución de un programa a través de expresiones booleanas (True, False).
- En R distinguiremos tres tipos de condicionales
 - Condicional **simple**
 - Condicional **anidado**.
 - Condicional **encadenado**.

8.3.- Condicionales If – simple

- Es el tipo más básico se estructura como
- **if** (expresión booleana se cumple)
 - Realizar una acción
- **else** (si no).
 - Realizar otra acción
- No tiene porqué aparecer obligatoriamente el bloque else.
- La sintaxis de un condicional en R es:

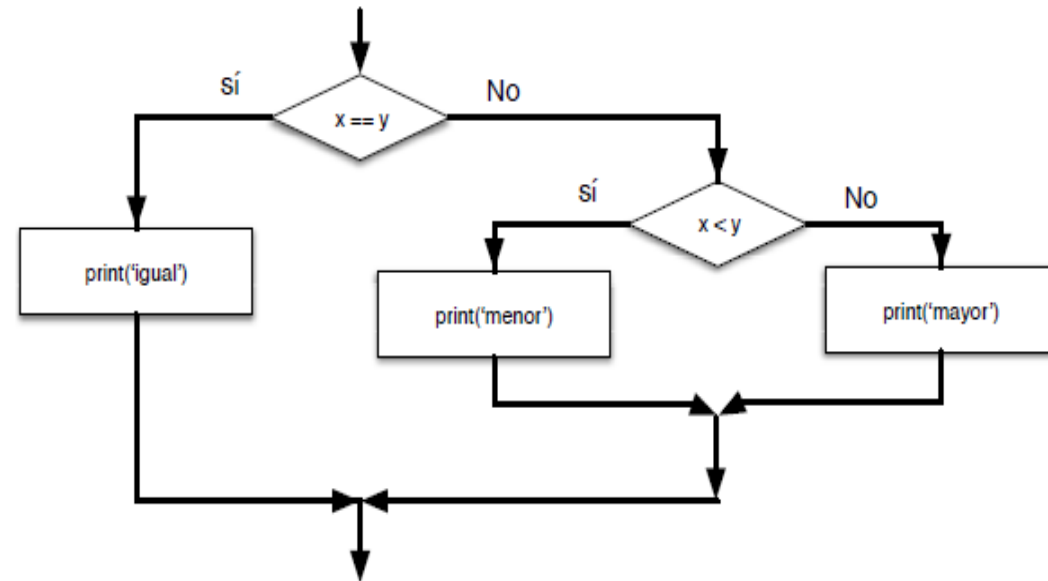
```
if (condición booleana) {  
  ...  
} else {  
  
}
```



8.4.– Condicionales If – anidado

- Puede entenderse como tener un condicional dentro de un condicional. Por ejemplo.

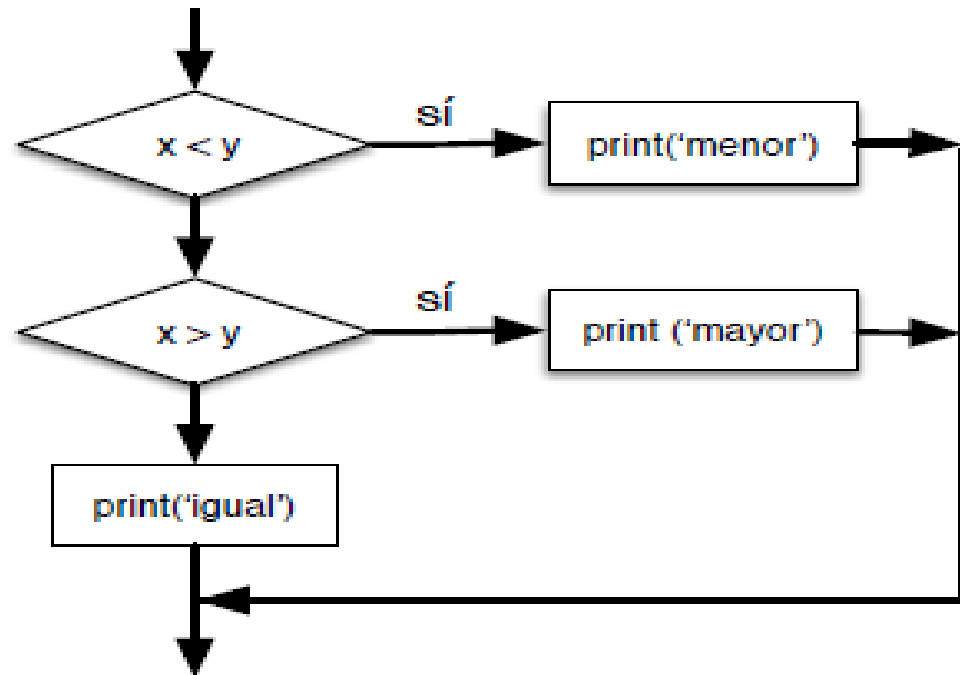
- **if** (condición booleana) {
 - **if** (condición booleana) {
 - Realizar una acción
 - } **else** {
 - Realizar otra acción }
- **else** (si no).
 - Realizar otra acción



8.5.– Condicionales If – Encadenado

- Al término de un condicional (simple o anidado), aparece otro condicional (simple o anidado). Por ejemplo:

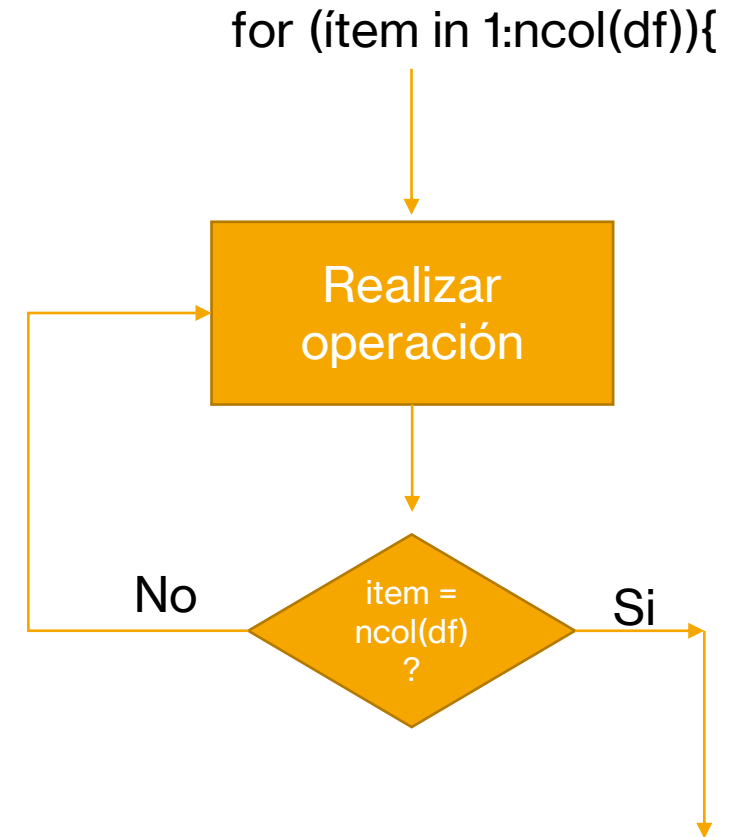
- `if (condición booleana) {`
 - Realizar una acción
- `} else if (condición booleana) {`
 - Realizar una acción
- `} else {`
 - Realizar otra acción }



8.6.– Bucles – For

- Realiza de forma secuencial una misma instrucción, hasta llegar al fin de la secuencia.
- La sintaxis de un bucle **for** es.

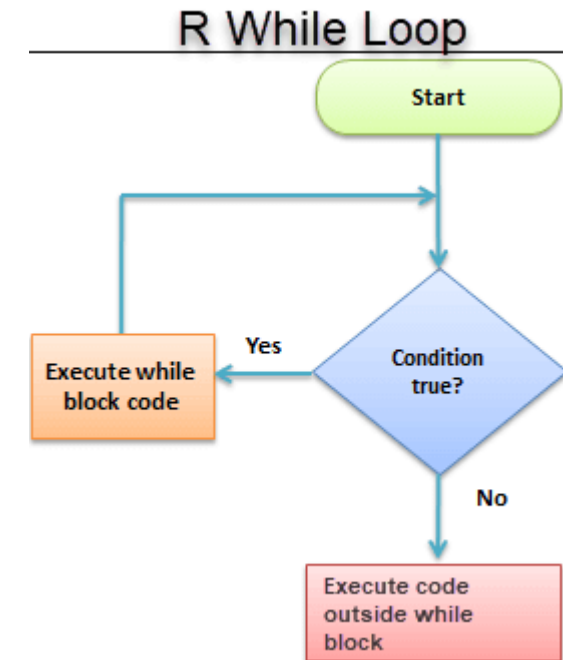
```
for (variable in secuencia) {  
...  
}
```



8.7.– Bucles – While

- Se realiza una misma acción hasta que se cumpla una instrucción de parada, dada por una condición booleana.
- La sintaxis de un bucle **while** en R es:

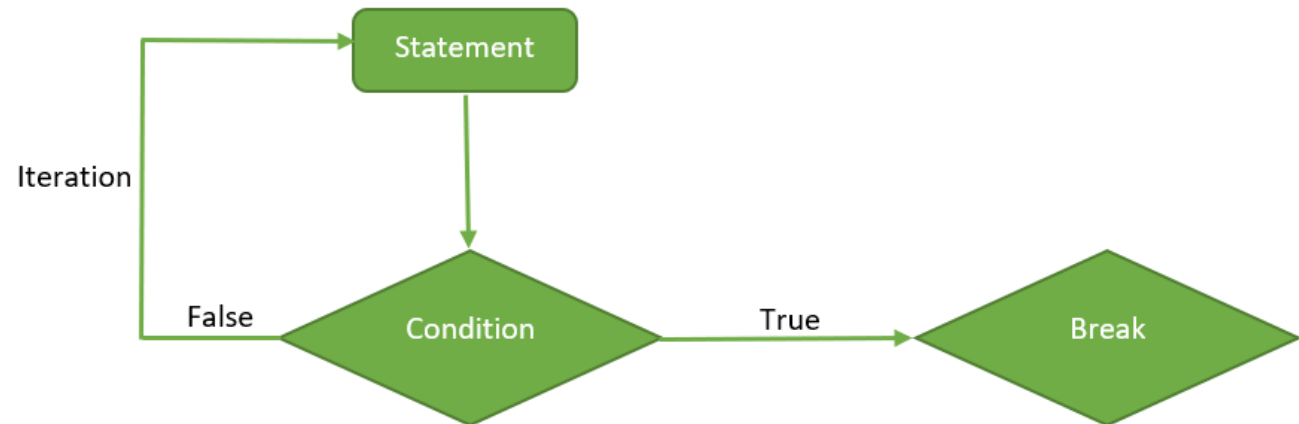
```
while (condición booleana) {  
  ...  
  ¿se cumple ahora la condición? – Instrucción de parada  
}
```




8.8.– Bucles – Repeat

- Se parte desde el valor de una variable, se realiza de forma repetitiva la misma operación hasta que el resultado cambie, por lo tanto, dentro del bucle **repeat** hay una instrucción condicional.
- Para salir de un bucle **repeat**, hay que emplear una condición para romper el ciclo, **break**.
- La sintaxis de un bucle **repeat** es:

```
repeat
{
    ...
    if( condicion booleana )
    {
        break
    }
}
```





09

Funciones

9.1.– Funciones

- Para aislar cualquier parte del código, hacerlo modular y reproducible se emplean funciones.
- Una función en R se denota con la palabra reservada **function**, a la cuál se le pasan los parámetros de entrada (o no).
- A través de la sentencia **return**, se devuelve el parámetro de salida.
- Para devolver más de un parámetro de salida, lo más aconsejable es emplear una lista para devolver los n parámetros.
- La estructura de una función en R es:

```
function nombre.funcion(parametro1, parametro2, ... , parametro n) {  
...  
return (variable de salida)  
}
```

9.1.– Funciones – Apply

- Existen una serie de funciones, que forman parte de la familia de funciones **apply**, que sirven para aplicar a una estructura de datos, ya sea por sus filas o columnas una misma función.
 - **apply**: Realiza la misma operación sobre un eje (1 filas, 2 columnas)
 - **lapply**: Mismo funcionamiento que apply pero, optimizado para listas, el resultado lo devuelve en forma de lista.
 - **sapply**: Recibe una lista y, devuelve un vector.
 - **tapply**: Realiza una operación sobre un vector en función de un vector de categorías, muy recomendado en dataframes.
 - **mapply**: Opera entre matrices o vectores, devuelve el resultado en forma de vector o, de lista si devuelve más de un resultado.



10

Anexo Operadores

3.– Anexo operadores

Operación	Operador
Negación	!(TRUE)
AND 1	TRUE & TRUE
AND 2	TRUE & FALSE
OR	TRUE FALSE

Seguimiento práctico del contenido

A partir de aquí, vamos a ver la parte fundamental de R con.

4_1_Primer_script.R

4_2_Primer_notebook.RMD

4_3_Sintaxis_básica_R.RMD

4_4_Estructuras_datos_1.RMD

4_4_Estructuras_datos_2.RMD

4_5_Herramientas_de_control.RMD

4_6_Funciones.RMD

4_7_Ejercicios_complementarios.RMD

IMF

Smart Education