Curso PIB-M 4ª Ed. (fase práctica)

Estadística climatológica con R

Breve introducción a RStudio®

Belinda Lorenzo Mariño
(blorenzom@aemet.es)

RStudio

https://posit.co/downloads/

RStudio es un IDE (Integrated Development Environment) práctico e interesante para trabajar con R. Dispone de ediciones de código abierto (licencia AGPL v3) y comerciales y es multiplataforma.

En nuestro caso disponemos de dos maneras interesantes para utilizarlo: el RStudio Desktop y el RStudio Server.

RStudio Desktop

Es la modalidad escritorio para desarrollo en local.

En los servidores usuales Linux de Aemet está disponible. Requiere la carga de su módulo rstudio previo a la ejecución con el comando `rstudio`.

También dispone de versiones Linux o Windows instalables para nuestro uso en PC local.

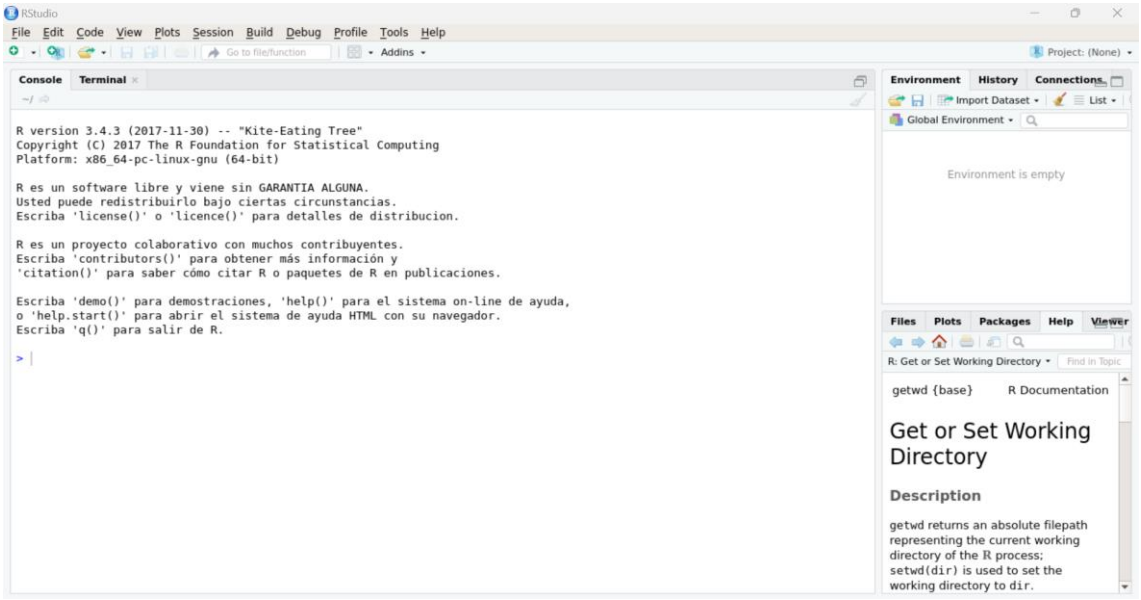


Figura 1. Apariencia inicial de RStudio Desktop

RStudio Server

Es la modalidad que se ha optado en este curso, debido a su comodidad y sencillez en el acceso.

El RStudio Server permite la ejecución de RStudio (alojado en un servidor), a través del navegador.

Para ejecutarlo:

- Abrimos un navegador, desde nuestro PC o portátil local. Por ej.: Firefox, Chrome.
- Nos dirigimos a la siguiente url: http://sur.aemet.es:8787/
- Se introducen las credenciales correspondientes a nuestro usuario del curso.

Una vez que se accede al RStudio Server, se nos presenta la aproximadamente la siguiente página. Está estructurada en diferentes zonas:

- Menú superior de herramientas y opciones. Menú rápido.
  Mediante estos menús se tiene acceso a la funcionalidad del IDE.
- Botones de desconexión.
- Paneles de la zona de trabajo.

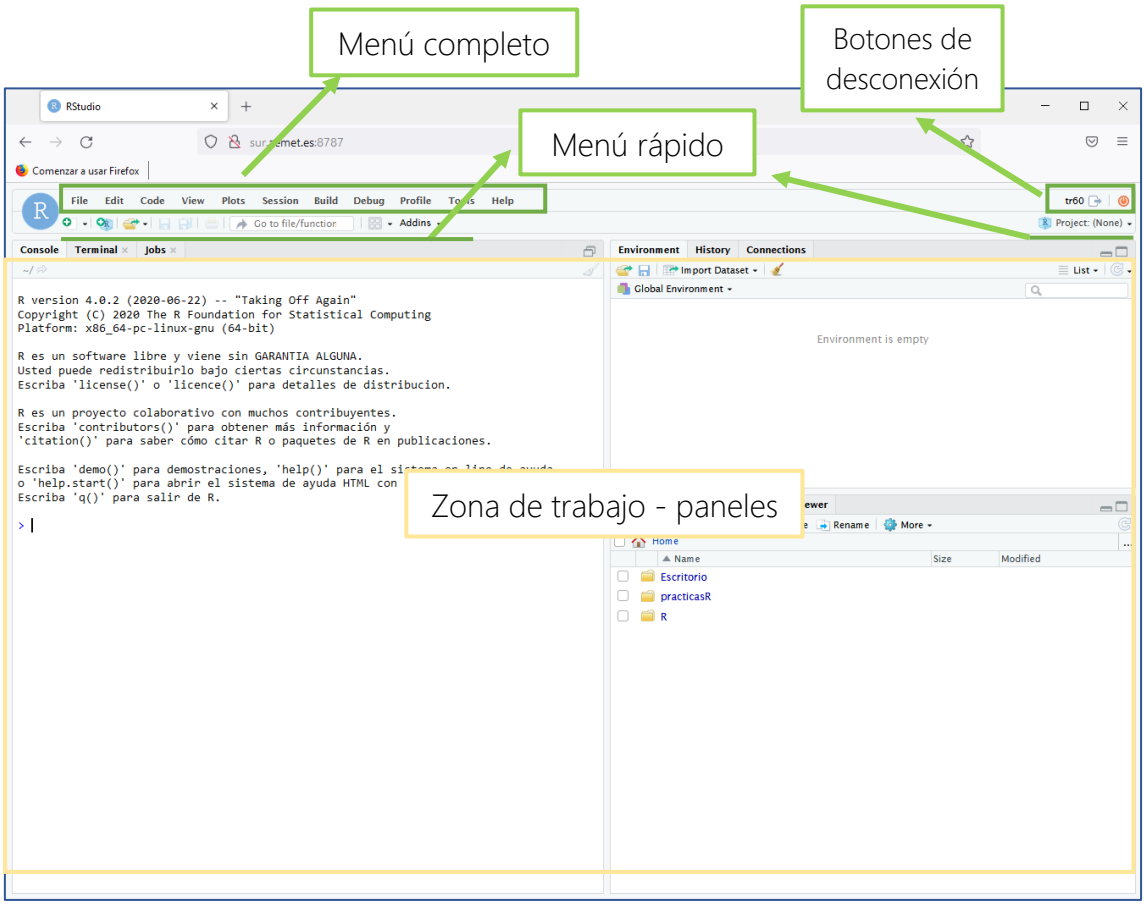


Figura 2. Apariencia inicial de RStudio Server

Dispone de cuatro paneles principales en la zona de trabajo. Para verlos todos hay que abrir algún fichero (File → Open File…).

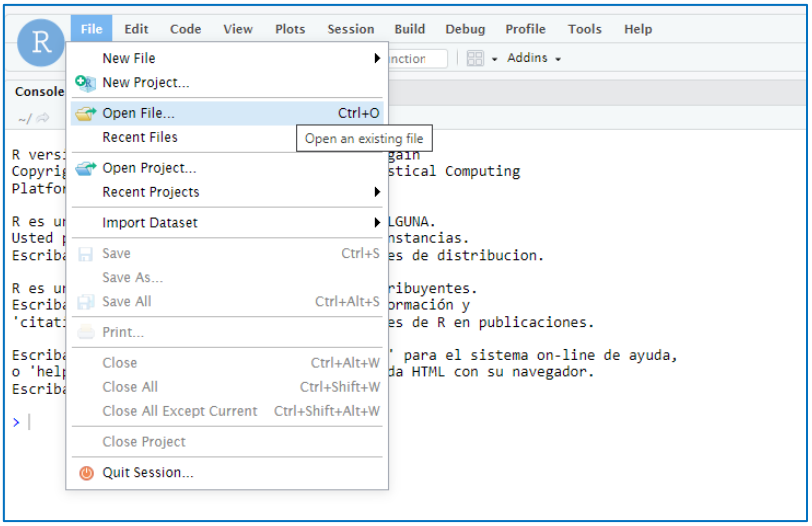


Figura 3. Abriendo un archivo desde menú.

Al abrir un fichero, aparece en la zona superior izquierda el panel que faltaba con el contenido o código del fichero:

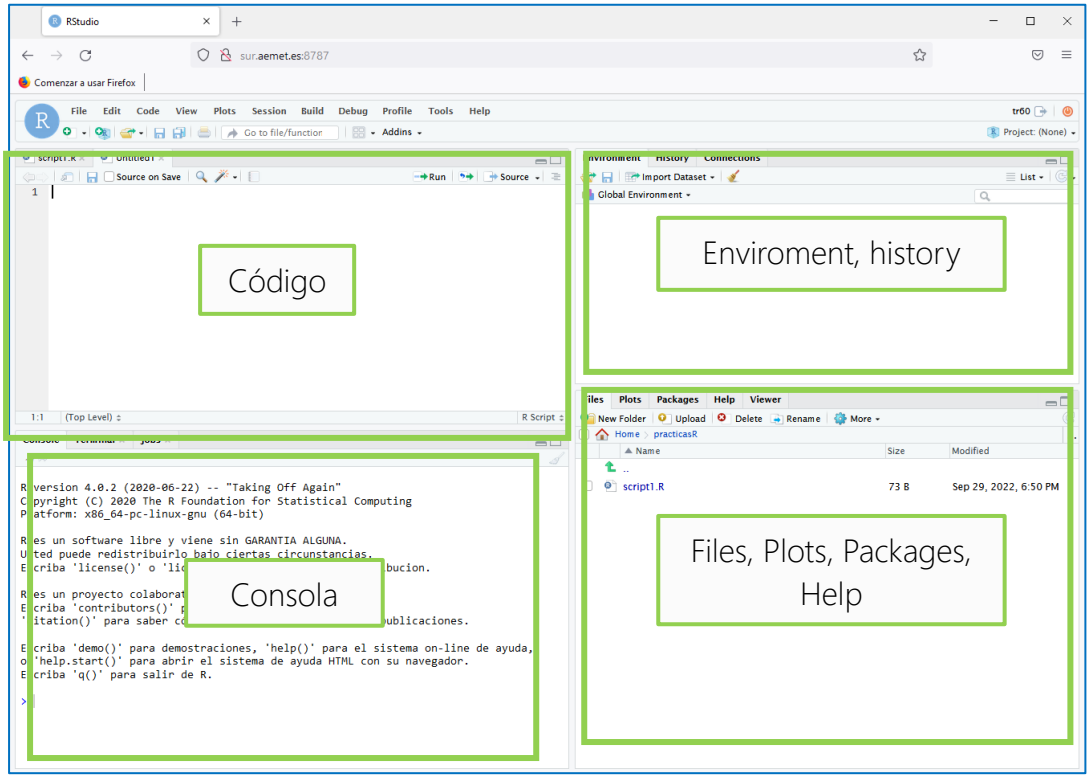


Figura 4. Los cuatro paneles del área de trabajo.

- **Panel de código**: es la zona dedicada a visualización y edición de ficheros de código (scripts, markdowns), así como la de visualización de información de objetos o ficheros (tablas, dataframes, etc.), funciones, etc.

Aquí se puede seleccionar el código a ejecutar de diversas maneras, y éste se ejecutará en la consola.

- **Panel de "enviroment"**: en esta zona obtendremos información sobre los enviroments (variables, funciones, etc. cargadas), así como un historial de los comandos y conexiones.
- **Panel de consola**: en este panel se accede a las consolas, a las terminales del sistema, jobs,...
- **Panel de "files, plots, etc."**: en este panel se pueden explorar los ficheros del sistema, se representarán los plots y views. Se presenta también la pestaña de paquetes, donde se pueden cargar, instalar, etc. También en este panel tenemos la pestaña de ayuda, donde se puede consultar la ayuda de funciones y paquetes instalados.

Todos los paneles tienen cierta conexión entre sí.

La disposición de paneles y las pestañas asociadas se puede configurar en la vista y las opciones de RStudio.

Veamos a continuación algunas curiosidades.

Panel del Ficheros, Plots, etc

Files

En la pestaña "Files" tenemos acceso a los ficheros de nuestro usuario en el servidor.

Se puede navegar, manipular ficheros, etc. pero quizás lo más interesante es que permite subir ficheros al servidor, y también descargarlos a nuestro local.

Se da el caso de que si subimos ficheros comprimidos, automáticamente se descomprimen en el directorio donde se suben.

Subimos a modo de ejemplo el comprimido ejemplos_rstudio.zip (Figura 5).

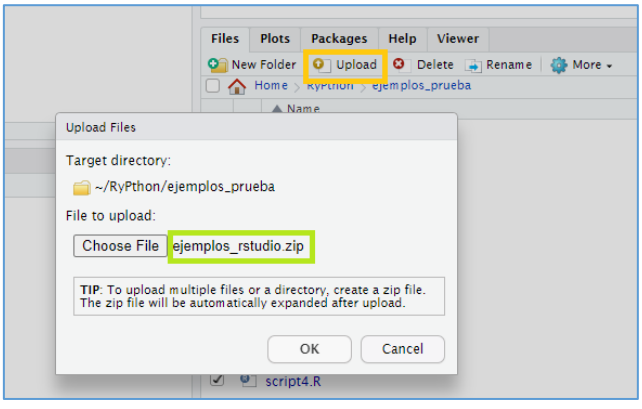


Figura 5. Subiendo múltiples ficheros al servidor.

Se puede observar que en el directorio del servidor aparece el contenido del zip descomprimido.

Plots

Es una pestaña dedicada a la visualización gráfica.

Permite exportar los gráficos en diferentes formatos (Figura 6).

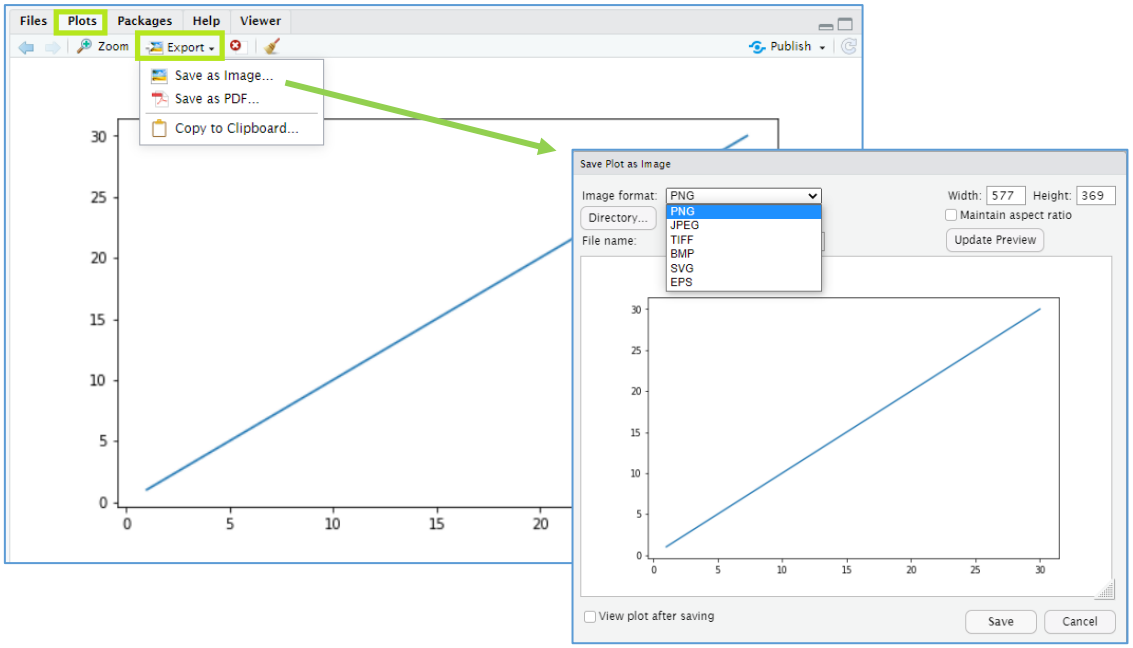


Figura 6. Exportando un gráfico dese la pestaña "Plots"

Packages

Es una pestaña dedicada a la gestión de los paquetes.

Desde ahí se destaca que se pueden activar o desactivar (library()), actualizar e instalar y acceder a las ayudas específicas de cada paquete (estas se visualizarían en la pestaña "Help").

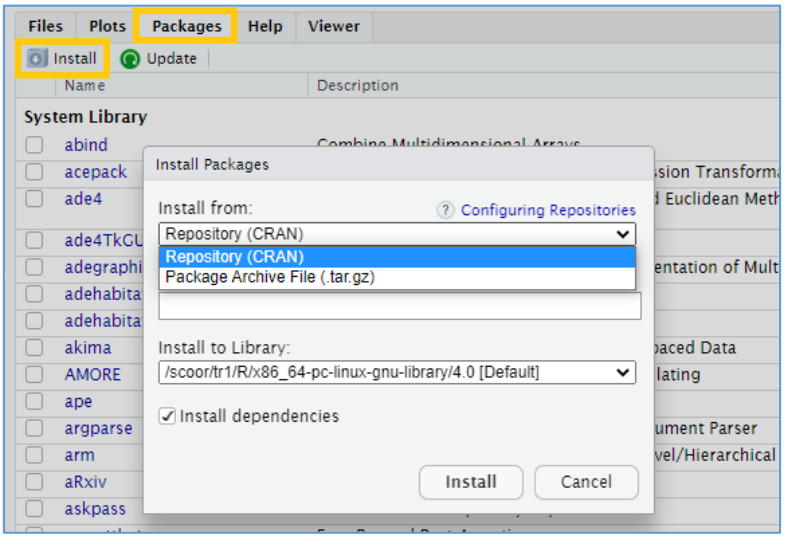


Figura 7. Ventana de instalación de paquetes.

Help

Es la pestaña donde se muestra la ayuda, documentación de paquetes y funciones. Hay múltiples formas de activar la ayuda desde RStudio, mostrándose en esta pantalla.

El funcionamiento es similar a un navegador, permitiendo ir a páginas anteriores, posteriores, volver al inicio o incluso imprimir la ayuda. Su motor de búsqueda es de gran ayuda a la hora de encontrar la que necesitemos.

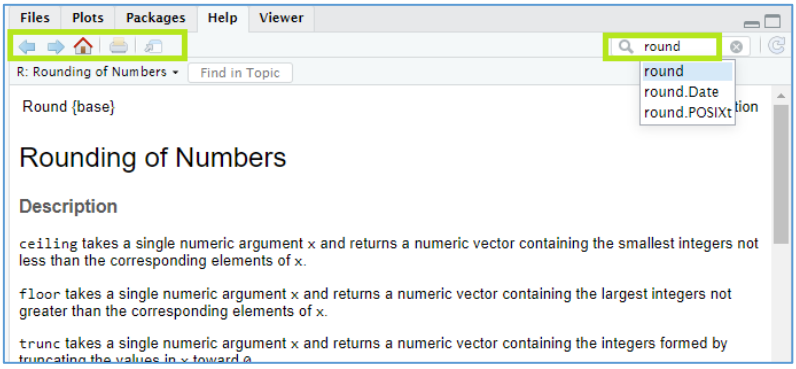


Figura 1. Página de ayuda de la función round().

Panel del código y consola

Se ha visto que se pueden abrir o crear diferentes scripts y documentos de código, y que se visualiza en el panel de arriba a la izquierda por defecto.

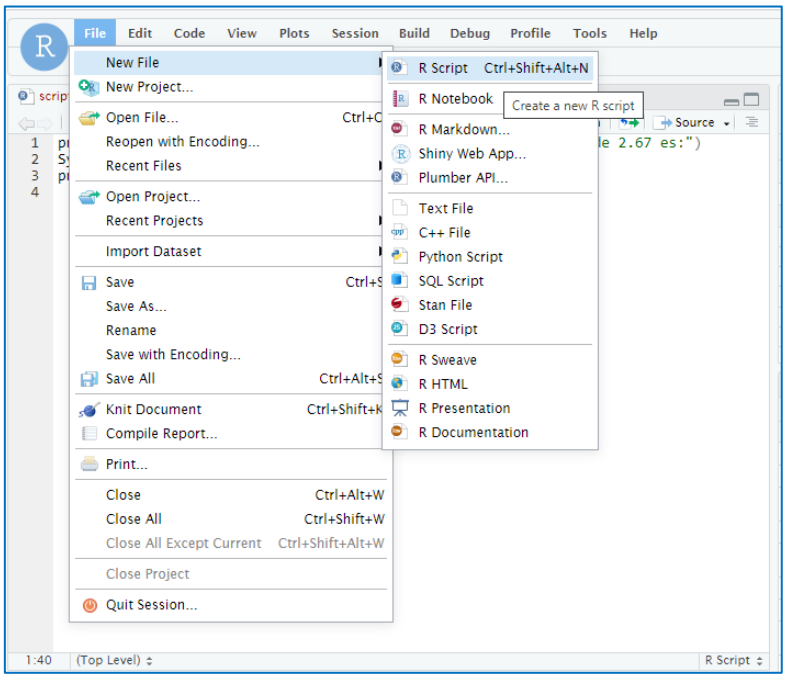


Figura 9. Creando nuevo R script desde menú.

Tenemos la posibilidad de ejecutar parte del código escrito o incluso cargar/ejecutar todo un fichero mediante las opciones de la parte superior derecha del editor de código (Figura 10).

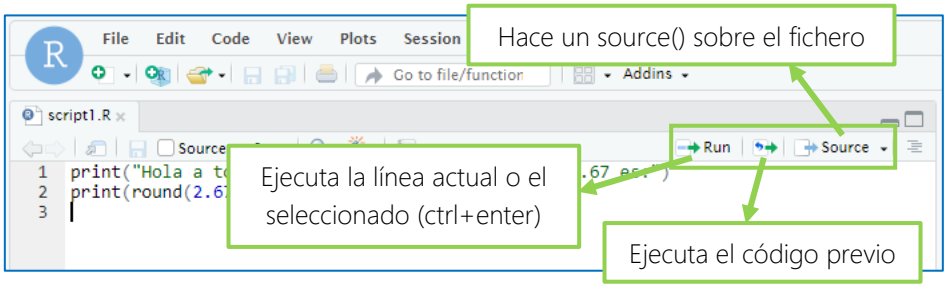


Figura 2. Botones de ejecución de código.

La ejecución se realiza en la consola de R, que suele estar en el panel inferior izquierdo.

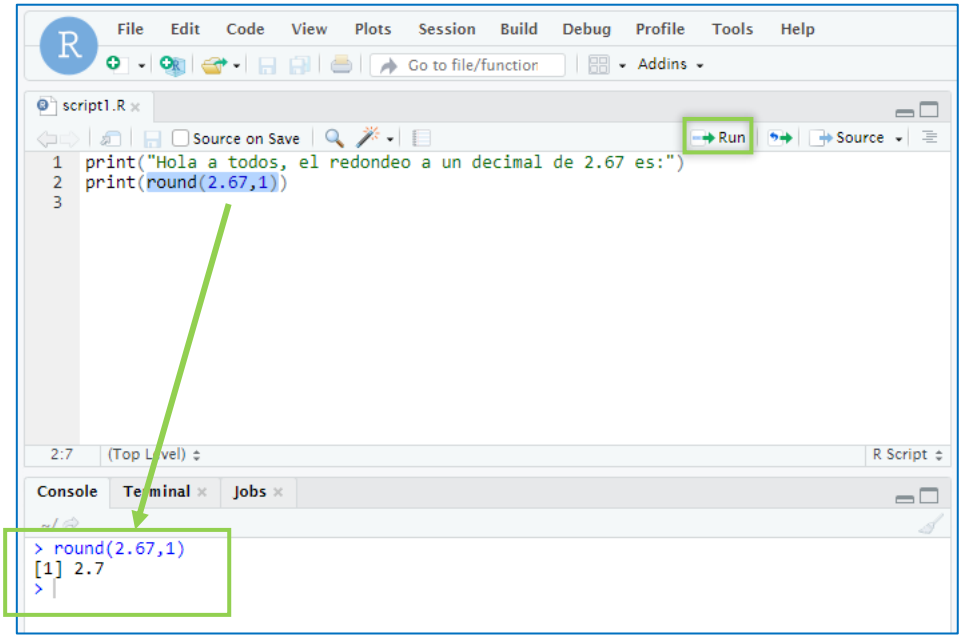


Figura 3. Ejecución de código seleccionado.

En el ejemplo de la anterior imagen (Figura 11), al seleccionar un trozo de código y ejecutarlo, automáticamente este se ejecuta en la consola de R.

Cabe señalar las funciones de **autocompletado**, activas en ambos paneles.

Mientras se escriben funciones, objetos y demás suele aparecer un cuadro emergente con las posibilidades coincidentes en la sesión. Podemos intentar autocompletar pulsando asimismo la tecla <Tab>.

En el caso de funciones es especialmente interesante, ya que además va seleccionando los parámetros que se pueden introducir. Y en el caso de estar documentadas, aparecen cuadros emergentes amarillos con detalles de la ayuda, a la que se puede acceder de forma completa pulsando <F1>.

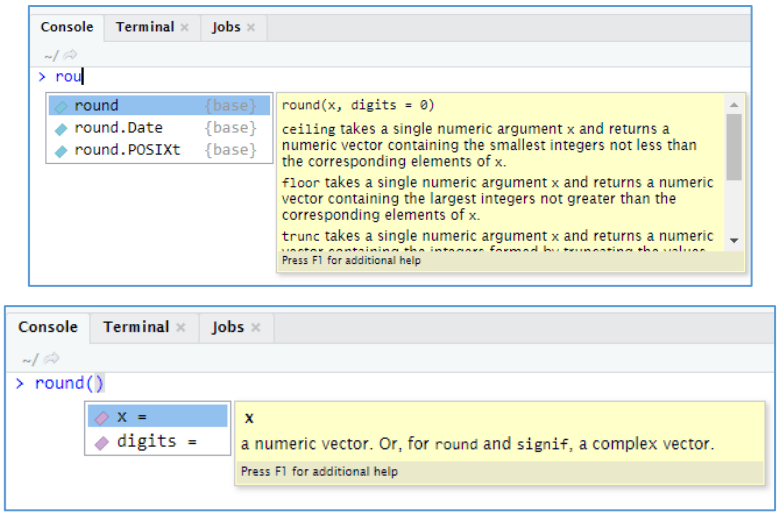


Figura 4. Autocompletado de código con ayuda.

Ejecución de scripts a través de "jobs".

La ejecución a través de jobs es interesante, ya que permite tener varias ejecuciones corriéndose en la sesión, a la vez que se sigue teniendo disponible la consola de trabajo en R.

Hay varias formas de proceder para ejecutar un script a través de un job. Podemos acceder al botón "Source" de la zona superior derecha en la ventana del script, y seleccionar "Source as Local job":

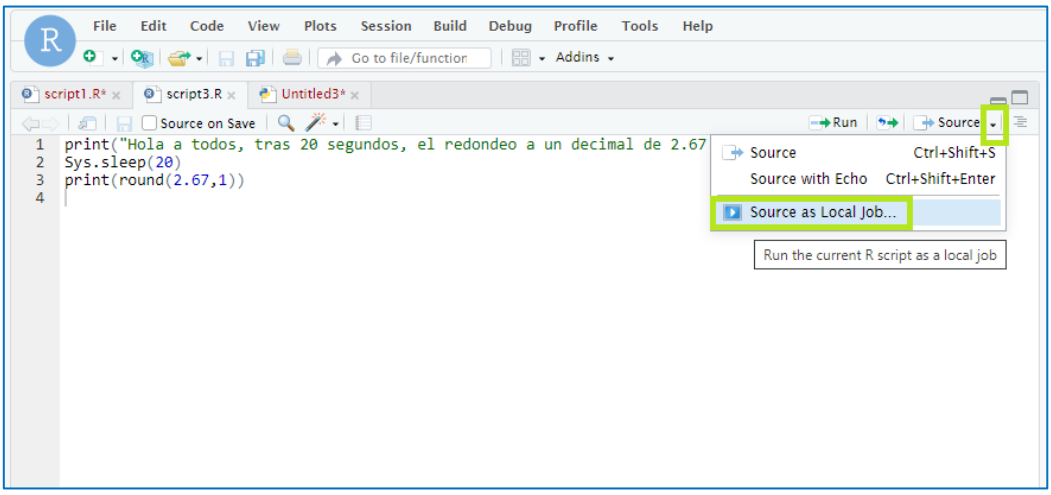


Figura 13. Iniciando un Job local desde botón Source..

O por ejemplo, desde el panel de la consola, seleccionar la pestaña de jobs, y pinchar en "Start Local job" (Figura 14).

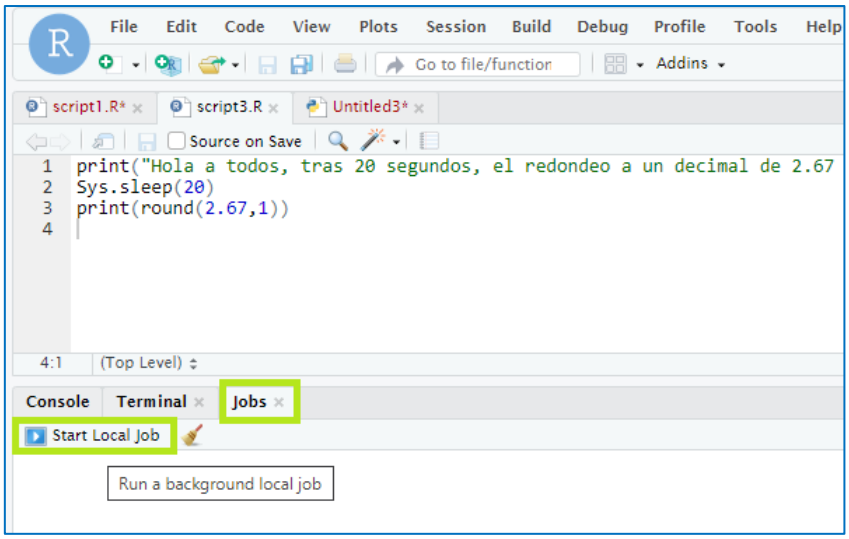


Figura 5. Iniciando Job local desde pestaña Jobs

En ambos casos saldrá una ventana emergente (Figura 17) donde se puede seleccionar el fichero a ejecutar y el directorio de trabajo. Por defecto selecciona el fichero cargado en el momento.

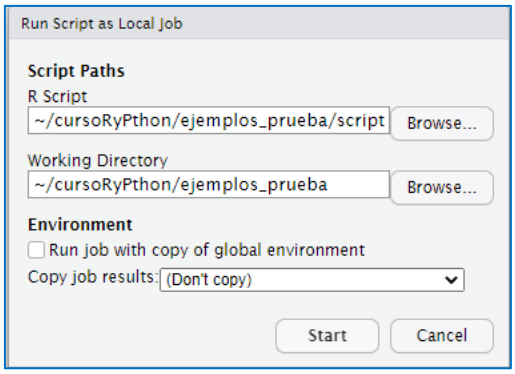


Figura 6. Ventana de selección de Job.

Al pinchar en Start, el proceso se queda ejecutando de fondo, en background.

En la pestaña Jobs se puede examinar la evolución de los Jobs activos.

Pinchando en cada job, se ve la ejecución del script correspondiente. Y si navegamos hacia atrás pinchando en la zona de la flecha hacia atrás, iríamos a la ventana principal de la pestaña "Jobs" donde se mostraría en una lista el estado de los procesos en ejecución o ya terminados.

Este procedimiento resulta interesante para el caso de procesos simultáneos de cierta duración. Hay que tener en cuenta que además la consola de R queda libre y mientras suceden las ejecuciones se puede trabajar de manera habitual, sin necesidad de establecer nuevas sesiones de R.

Creación de documentos

Veamos ahora unas pequeñas pinceladas de cómo crear documentos markdown desde RStudio.

Mediante R markdown podemos crear documentos escritos en código markdown, y ese mismo código/documento renderizarlo en diferentes formatos de salida: html, pdf, documentos Word, diferentes formatos de presentaciones, etc. (formatos)

Estos documentos permiten inserción y ejecución de código, son dinámicos y automatizables.

Podemos abrir o crear documentos R markdown fácilmente desde el menú "File" (Figura 16). Son ficheros con extensión Rmd.

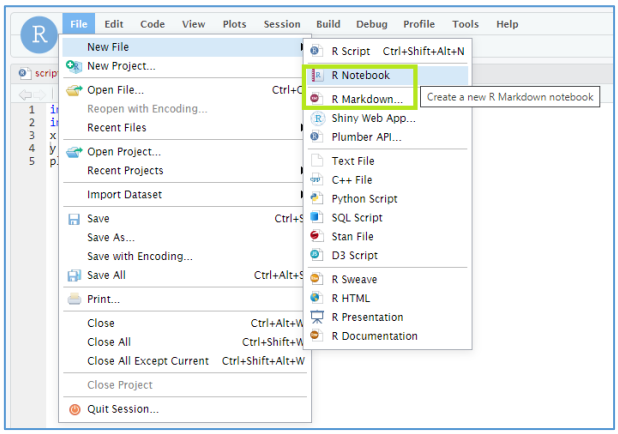


Figura 16. Creando un nuevo documento R markdown desde menú.

En la primera parte del documento se establecen los metadatos y parámetros de renderizado.

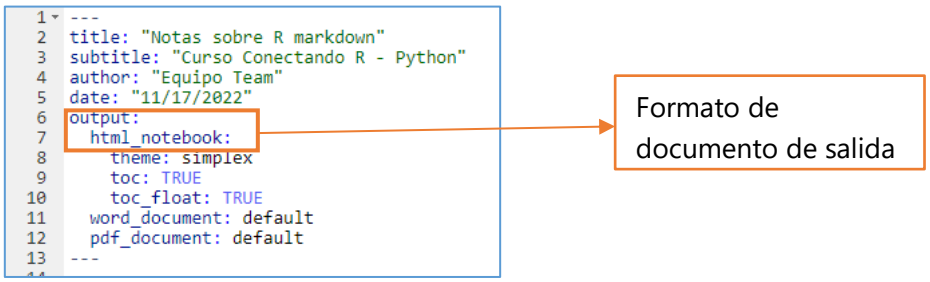


Figura 17. Parte inicial del documento.

En el menú "Knit" de la parte superior se pueden seleccionar ciertos tipos de renderizado.

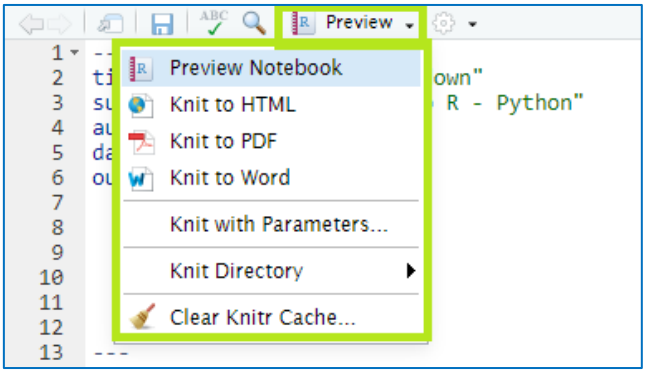


Figura 18. Menú "Knit".

Texto

A la hora de introducir texto, se sigue la sintaxis markdown básica, donde por ejemplo tenemos:

Títulos

```
# Este es un título 1
## Este es un título 2
### Este es un título 3
#### Este es un título 4
##### Este es un título 5
###### Este es un título 6
```

Este es un título 1

Este es un título 2

Este es un título 3

Este es un título 4

Este es un título 5

Este es un título 6

Texto

```
Esto es texto en **negrita**.
Esto es texto en *cursiva*.
Este _también_.
Así metemos superíndice^100^
Así subíndice~100~
Así tachamos ~~texto~~
```

Esto es texto en **negrita**.
Esto es texto en *cursiva*.
Este *también*.
Así metemos superíndice^{100}
Así subíndice$_{100}$
Así tachamos ~~texto~~

Listas

```
## Lista sin numerar
Así insertamos una lista sin numerar:

* Elemento 1
* Elemento 2
* Elemento 3

o así:

- Elemento 1
- Elemento 2
- Elemento 3
```

Lista sin numerar

Así insertamos una lista sin numerar:

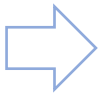
- Elemento 1
- Elemento 2
- Elemento 3

o así:

- Elemento 1
- Elemento 2
- Elemento 3

```
## Lista numerada
Así insertamos una lista numerada:

1. Elemento 1
2. Elemento 2
3. Elemento 3
```



Lista numerada

Así insertamos una lista numerada:

1. Elemento 1
2. Elemento 2
3. Elemento 3

Imágenes

```
## Así puedo insertar una imagen

![Foto otoño](autumn_cut.jpg)
```

Así puedo insertar una imagen

Foto otoño

Enlaces

```
## Así podemos insertar enlaces

[AEMet](http://www.aemet.es)
```

Así podemos insertar enlaces

AEMet

Código

Se añade una gran potencialidad a la hora de combinar texto con inserción de código dinámicamente.

Los bloques de código (chunks) se pueden insertar de diversas maneras. Un atajo corto a código en R es <ctrl>+<atl>+<i>.

También se puede insertar a través del botón de la parte superior derecha:

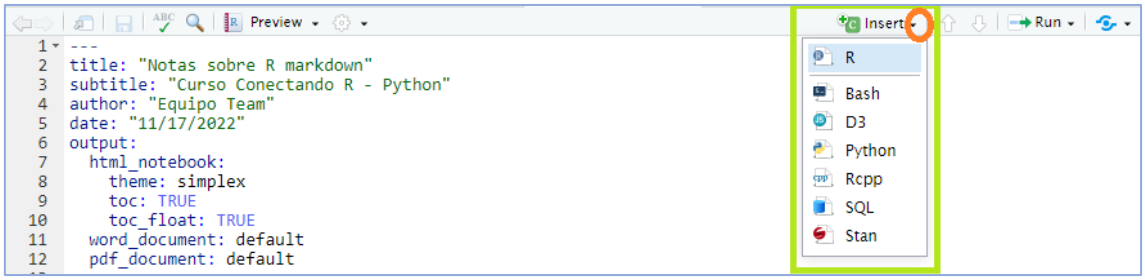


Figura 19. Botón de inserción de código.

Esencialmente los bloques de código son aquellos comprendidos entre las 3 comillas inversas **```{r} código ```**.

Podemos atribuir etiquetas, modificar parámetros de display, etc. en las opciones de los chunks.

Veamos algunos parámetros interesantes (ejemplo de algunos parámetros en Anexo I):

OPTION	DEFAULT	EFFECTS
echo	TRUE	display code in output document
error	FALSE	TRUE (display error messages in doc) FALSE (stop render when error occurs)
eval	TRUE	run code in chunk
include	TRUE	include chunk in doc after running
message	TRUE	display code messages in document
warning	TRUE	display code warnings in document
results	"markup"	"asis" (passthrough results) "hide" (don't display results) "hold" (put all results below all code)
fig.align	"default"	"left", "right", or "center"
fig.alt	NULL	alt text for a figure
fig.cap	NULL	figure caption as a character string
fig.path	"figure/"	prefix for generating figure file paths
fig.width & fig.height	7	plot dimensions in inches
out.width		rescales output width, e.g. "75%", "300px"
collapse	FALSE	collapse all sources & output into a single block
comment	"##"	prefix for each line of results
child	NULL	files(s) to knit and then include
purl	TRUE	include or exclude a code chunk when extracting source code with knitr::purl()

See more options and defaults by running **str(knitr::opts_chunk$get())**

Figura 7. Parámetros de chunks y valores por defecto.

Se puede establecer un chunk de configuración, con los parámetros que se aplicarán a todos los bloques de código del documento, mediante por ejemplo:

```{r setup} knitr::opts_chunk$set() ```

```
```{r setup, include = FALSE}
knitr::opts_chunk$set(
  warning = FALSE,
  message = FALSE,
  echo = FALSE,
  fig.path = "alison-figs/"
)
```
```

Temas

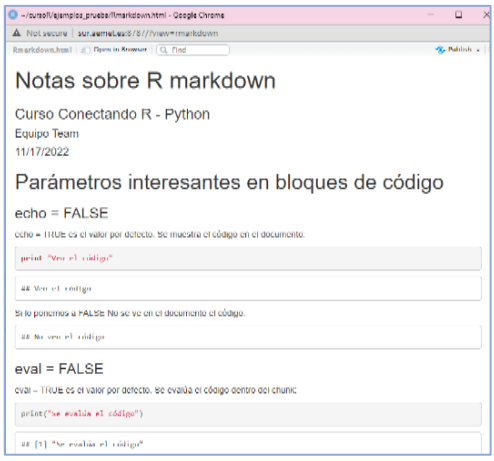
Simplemente como curiosidad se disponen de una serie de temas estéticos sobre la visualización, que simplemente podemos variar incluyéndolo en la zona de metadatos del documento.

Algunos ejemplos:

Básico:

```
---
title: "Notas sobre R markdown"
author: "Equipo Team"
date: "11/17/2022"
output:
  html_document:

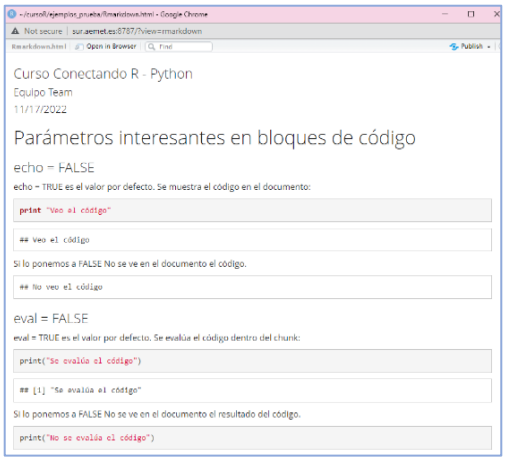
  pdf_document: default
  word_document: default
subtitle: Curso Conectando R - Python
---
```



yeti:

```
---
title: "Notas sobre R markdown"
author: "Equipo Team"
date: "11/17/2022"
output:
  html_document:
    theme: yeti
    # toc: yes
    # toc_float: yes

  pdf_document: default
  word_document: default
subtitle: Curso Conectando R - Python
---
```



Simplex con toc:

```
---
title: "Notas sobre R markdown"
author: "Equipo Team"
date: "11/17/2022"
output:
  html_document:
    theme: simplex
    toc: yes
    toc_float: yes

  pdf_document: default
  word_document: default
subtitle: Curso Conectando R - Python
---
```