Guía de RStudio

PID_00265511

Jordi Mas Elias

Tiempo mínimo de dedicación recomendado: 2 horas

Universitat Oberta de Catalunya

Jordi Mas Elias

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Jordi Mas Elias (2019)

Primera edición: septiembre 2019

Avda. Tibidabo, 39-43, 08035 Barcelona
Realización editorial: FUOC

Índice

Introducción

El uso de los programas de gestión de datos es una tarea cada vez más extendida en los estudios internacionales. Si bien el conocimiento de este tipo de programas no es un requisito indispensable en el mundo laboral en el ámbito de las ciencias sociales, también es verdad que tener un dominio básico de ello se está convirtiendo en una plataforma muy útil para acceder a puestos de trabajo en este sector. Una primera aproximación del estudiante a la gestión de datos es, pues, importante en dos sentidos. En primer lugar, realizar los primeros pasos con estos programas puede ayudar a crear, en algunos estudiantes, el interés por el mundo de los datos y las ganas de continuar aprendiendo el lenguaje de R u otros programas como por ejemplo Python, Stata o SPSS. En segundo lugar, el aprendizaje inicial de un lenguaje como el de R dota al estudiante de unas habilidades que le pueden facilitar oportunidades laborales que requieran un conocimiento mínimo de la gestión y el análisis de datos.

Este módulo tiene como principal objetivo ayudar al estudiante a familiarizarse con el entorno de R. Es por eso que las siguientes páginas pretenden ser una guía práctica para iniciarse en los softwares R y RStudio y complementar otros módulos de UOC que utilizan R para sus propósitos docentes. Los otros objetivos de estos materiales son:

- Ayudar a descargar e instalar los programas R y RStudio.
- Facilitar el conocimiento de la interfaz de RStudio de forma que ayude al estudiante a familiarizarse con las principales ventanas del programa.
- Explicar cómo se empieza un proyecto con RStudio, cuáles son los primeros pasos y cómo se guarda un proyecto.
- Realizar una pequeña incursión en el lenguaje de R, mostrando algunos ejercicios prácticos que el estudiante puede hacer antes de empezar a trabajar los módulos.

Con todo, también hay que tener en cuenta dos consideraciones. En primer lugar, nos tenemos que tomar esta guía como un manual de ayuda que nos permitirá dar los primeros pasos con un software desconocido hasta ahora para nosotros como es R. Todo lo que se enseña en estas páginas va dirigido a poder seguir con más facilidad las indicaciones y actividades que encontraremos en otros módulos de esta asignatura. Aprenderemos, pues, a hacer una primera inmersión en el lenguaje de R y, muy importante, seremos capaces de resolver errores que nos puedan sobrevenir durante la realización de las actividades de la asignatura. La guía no es, por lo tanto, un manual que enseñe a programar o analizar datos con R. Quien esté interesado en adentrarse en el mundo del

(1)Para guías más avanzadas, probad Advanced R (http://adv-r.had.co.nz) o R for Data Science (https://r4ds.had.co.nz/).

análisis de datos puede consultar guías para aprender a hacer funcionar R y RStudio que encontrará en el seminario en línea y en la sección de aprendizaje en línea del propio sitio web de RStudio, o bien puede buscar diferentes guías, foros o tutoriales, fáciles de encontrar en internet (ved, por ejemplo, Stack Overflow). Estas guías permiten empezar a utilizar el lenguaje R desde un nivel inicial y aprenderlo de manera progresiva hasta obtener un nivel avanzado[1].

La segunda advertencia se refiere a la dificultad de iniciarse en los programas de gestión de datos. Este tipo de software acostumbra a tener una curva de aprendizaje muy larga. Para una persona que se ponga por primera vez delante de este tipo de programas, las primeras horas pueden ser difíciles y pesadas. Se aprende muy despacio, a partir de equivocarse y tropezarse muchas veces, y solo una formación continuada en el tiempo puede comportar grandes retornos en el uso de este tipo de programas. Por lo tanto, el estudiante tendrá que tener paciencia y seguir muy atentamente esta guía para poder superar las siempre difíciles horas iniciales. Hay que decir que las horas invertidas no son en vano, sino que son una ganancia en experiencia acumulada que ayudará en el futuro a incrementar la productividad en el uso del programa.

R y RStudio

La asignatura que trabajaréis requiere el uso de los programas R y RStudio. Son dos programas diferentes y los dos se tienen que instalar en el ordenador. Una vez instalados, todas las operaciones indicadas en este manual las haremos utilizando RStudio, pero también es necesario tener instalado el programa R, aunque no lo ejecutamos en ningún momento, porque es el motor que utiliza RStudio para funcionar. En el siguiente apartado, se indican los pasos que se tienen que seguir para instalar los dos programas.

A lo largo de esta guía utilizaremos indistintamente los términos R y RStudio. Esto no quiere decir que en los casos en que nos referimos a R tengamos que utilizar el programa R. Siempre tendremos que utilizar RStudio en todas nuestras operaciones, pero utilizaremos los dos términos en esta guía, entendiendo normalmente R como el motor que hace funcionar RStudio y procesa la información, y RStudio como la interfaz, entendido como el entorno de programación que nos permitirá utilizar las herramientas del programa R de una manera más visual y eficiente. Podemos utilizar R y RStudio desde un sistema operativo Windows, Mac OS X o Linux.

¿Qué haremos?

Esta guía inicial está estructurada de la manera siguiente. En primer lugar, veremos cómo descargar y ejecutar los programas R y RStudio. En segundo lugar, ejecutaremos el programa RStudio y explicaremos la interfaz, en especial las cuatro ventanas con las cuales trabajaremos. Esta guía contiene ejercicios prácticos y está pensada para ser utilizada en paralelo con RStudio. De este modo, el estudiante podrá aplicar directamente en el programa las indicacio-

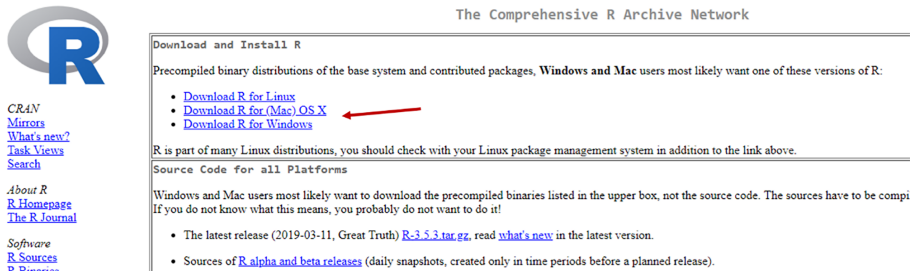
nes propuestas. También aprenderemos a utilizar los paquetes o librerías de R y a preparar los primeros códigos con el script. Finalmente, esta guía contiene un apartado de ayuda para resolver errores.

1. Primer paso: descargar R y RStudio

R y RStudio son dos programas diferentes que se tienen que bajar desde páginas web diferentes. Necesitaremos, con todo, tener los dos programas instalados para trabajar. R es el motor y tiene los engranajes suficientes para funcionar de manera autónoma, pero su visualización es austera y poco práctica para un usuario que se inicia en lenguajes de programación. RStudio, en cambio, no es autónomo, necesita R para funcionar, pero incorpora muchas características que facilitan su manejo, como por ejemplo ventanas, elementos visuales y diferentes opciones que nos ayudarán a trabajar con R. Por lo tanto, nosotros trabajaremos con RStudio pero necesitaremos tener los dos programas instalados, puesto que RStudio no puede funcionar si no tenemos previamente instalado R.

Encontraremos los dos programas en sus respectivas webs. Recordad elegir la opción correcta en función de si vuestro sistema operativo es Linux, Mac o Windows. Para instalar R, el primer paso es acceder a su web y seleccionar el sistema operativo. Tened en cuenta que la versión del programa puede ser diferente a la que veis en las siguientes figuras. En este manual vemos la versión 3.5.3, pero es muy posible que cuando os descargáis el programa ya tengáis a vuestra disposición una versión más nueva.

Figura 1. Bajar R: primer paso, seleccionar el sistema operativo.



En caso de utilizar el sistema Mac, en la siguiente ventana encontraréis el enlace para bajar el archivo *.pkg*. En caso de Windows, hay que seleccionar 'Install R for the first time' y en la siguiente ventana podréis bajar la última versión.

Figura 2. Bajar R: bajar el archivo .pkg en caso de trabajar con sistema operativo Mac.

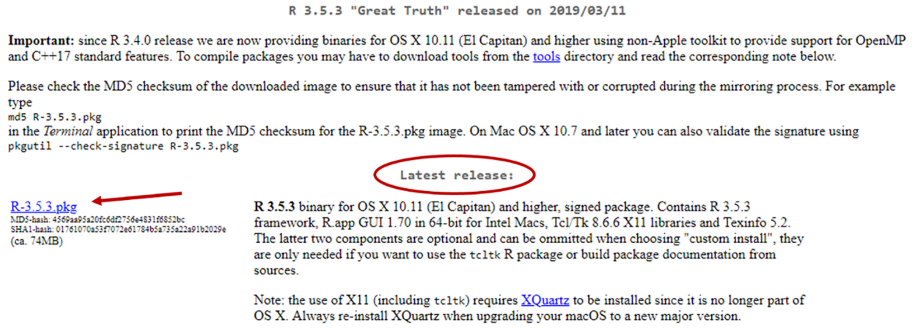


Figura 3. Bajar R: seleccionar 'Install R for the first time' en caso de trabajar con sistema operativo Windows.

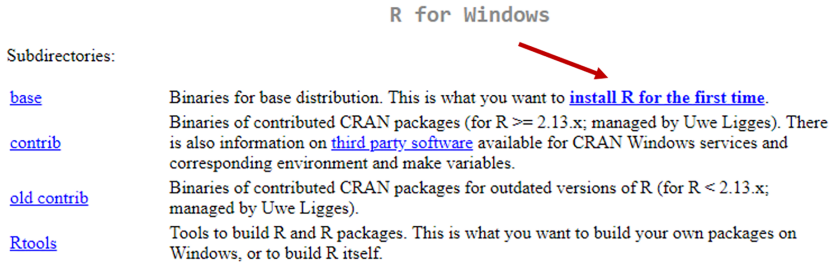
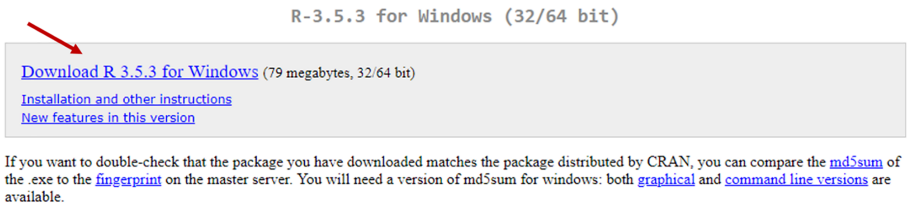
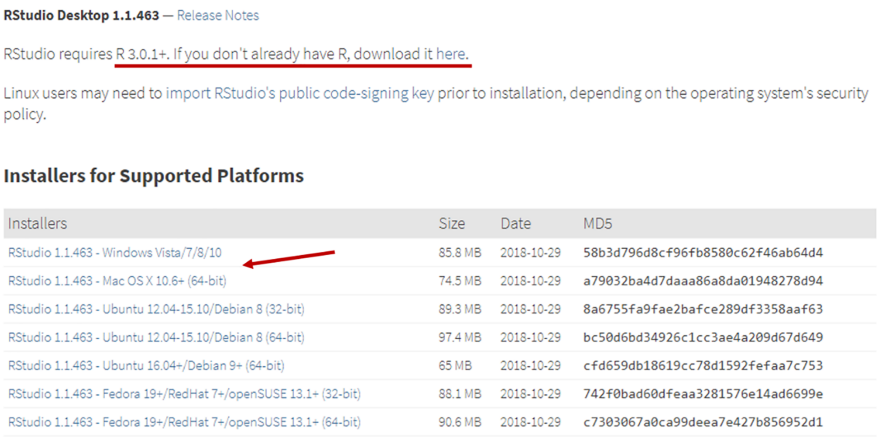


Figura 4. Bajar R: bajar la última versión en caso de trabajar con sistema operativo Windows.



Una vez descargado R, pasaremos a bajar RStudio. Tenemos que acceder a la web de descarga del programa y seleccionar el sistema operativo correspondiente. Fijaos bien en que, en la parte superior, nos deja claro que RStudio no puede trabajar sin tener R instalado.

Figura 5. Bajar RStudio: escoger el sistema operativo.

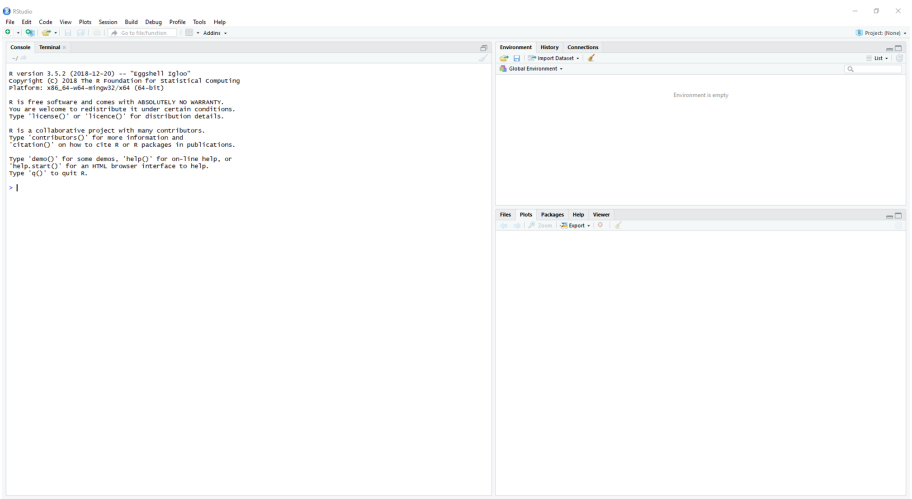


Recordad que, una vez bajados, tenéis que instalar los dos programas. Será necesario que el programa R esté instalado en nuestro ordenador, pero no será necesario ejecutarlo. Sí que instalaremos y ejecutaremos RStudio, que será el programa con el cual haremos todas las operaciones.

2. Segundo paso: situarnos

Cuando entremos en RStudio por primera vez, veremos un escritorio formado por tres ventanas principales. La interfaz del programa es muy simple, pero si no estamos acostumbrados a este tipo de entornos de programación nos costará un tiempo habituarnos a él. A la izquierda vemos una ventana más grande que las otras, con las pestañas 'Console' y 'Terminal', mientras que a la derecha vemos dos más pequeñas: la de encima tiene las pestañas 'Environment', 'History' y 'Connections', mientras que la de debajo tiene las pestañas 'Files', 'Plots', 'Packages', 'Help' y 'Viewer'.

Figura 6. Pantalla inicial de RStudio.



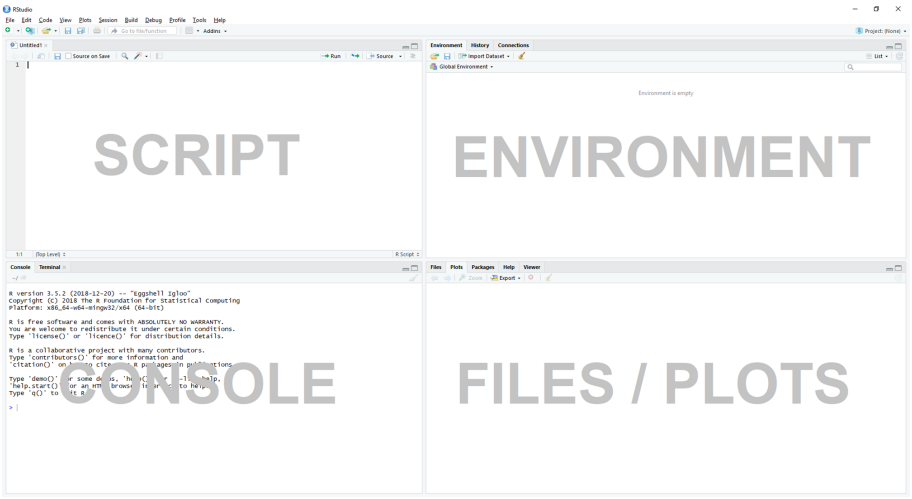
Lo primero que haremos es añadir una cuarta ventana al escritorio de RStudio:

- En la barra superior de la pantalla iremos a: File -> New File -> **R Script** (alternativamente podéis pulsar Ctrl+Shift+N). Normalmente la tecla equivalente al Ctrl en PC será la tecla Command en Mac.
- R acaba de crear una cuarta ventana con el nombre '**Untitled1**' y la ha ubicado en la parte superior izquierda del escritorio. Acabamos de crear un script, una de las herramientas fundamentales para utilizar R. Lo primero que haremos es guardar este documento en blanco (podemos guardarlo clicando en el icono del disquete, pulsando Ctrl+S o bien yendo a File -> Save As).
- Cuando intentemos guardar el documento, lo más probable es que por defecto nos sugiera guardarlo en la carpeta "Documents" (o "Documentos") de nuestro ordenador. Lo más recomendable es crear una carpeta dentro de la carpeta de la asignatura titulada 'R Scripts'. tendremos que crear unos cuantos y bajarlos durante el curso, por lo tanto tener una carpeta donde guardarlos nos será útil.

- Este primero script lo guardaremos con el nombre, por ejemplo '**R_Script_prueba**'.

En la siguiente imagen vemos las cuatro ventanas de RStudio. Por comodidad, a la hora de trabajar, haremos las ventanas de la izquierda más anchas que las de la derecha, de forma que nos quede más o menos como la imagen.

Figura 7. Las cuatro ventanas de RStudio.



En los próximos apartados repasamos las cuatro ventanas (Console, Script, Environment y Files), una a una:

2.1. Console

La *console* o consola es el instrumento que utilizamos para comunicarnos con RStudio. Lo que nosotros le indicamos aquí, R lo leerá y realizará las operaciones necesarias. En esta ventana mantendremos la visualización de la pestaña 'Console' y tendremos escondida la de 'Terminal'. En la consola, fijémonos en el símbolo > que aparece al final de todo del texto inicial. Se le llama operador. En el operador es donde podemos introducir nuestras primeras indicaciones para comunicarnos con R. Teclead los códigos que aparecen en la tabla 1 y a continuación pulsad la tecla Enter.

Tabla 1. Operaciones iniciales con la consola

Código	Retorno
`1+1`	`[1] 2`

`1:50`	`[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20`
	`[21] 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40`
	`[41] 41 42 43 44 45 46 47 48 49 50`

`"Hola"`	`[1] "Hola"`

En cada código, vemos que R nos devuelve primero un número entre corchetes `[1]`. Este número, que encontraremos al comienzo de cada línea, nos indica en qué posición se encuentra el primer valor de cada fila de toda la cadena de valores que nos está mostrando. Como vemos, el primer valor de la primera fila es el valor 1. Cuando llega a la segunda fila, ya nos ha mostrado 20 valores y, por lo tanto, el primero de la segunda fila es el 21 en este ejemplo (en vuestro ordenador puede aparecer un valor diferente). El primero de la tercera es el 41. En este caso, el número entre corchetes nos coincide con el primer número de cada fila, pero no tiene que ser así necesariamente. Si pedimos a R que nos muestre los mismos números de una manera desordenada, veremos que ya no coinciden. Por ejemplo, teclead `sample(50)` y pulsad Enter.

A continuación, teniendo la consola seleccionada, pulsad en vuestro teclado la flecha hacia arriba. Veréis que aparecen los códigos que hemos entrado previamente. La consola memoriza todos los comandos que hemos pedido en la sesión actual que tenemos iniciada en RStudio. Con las flechas arriba y abajo del teclado podemos recuperar códigos anteriores y modificarlos. Seguimos, en la tabla 2, con una última prueba.

Tabla 2. Errores en R

Código	Retorno
`R`	`Error: object 'R' not found`
`$`	`Error: unexpected '$' in "$"`

Si entramos los dos códigos de la tabla anterior veremos que nos da error. La consola, en una anotación en rojo, nos indica en el primer caso que no ha encontrado el objeto 'R'. En el segundo caso, nos indica que ha encontrado un símbolo que no ha entendido. En nuestro proceso de aprendizaje con RStudio toparemos con decenas de errores en la consola. Tenemos que considerarlo un hecho habitual, que con paciencia solucionaremos, y saber que el primer paso para llegar a la solución requiere leer muy atentamente lo que comenta R en estos mensajes en rojo.

Por ejemplo, cuando hemos introducido `R`, el programa no nos ha encontrado ningún objeto con el nombre R. Efectivamente, no recordamos haber creado ninguno. Dad ahora la siguiente indicación al operador: `R <- c(1:2, 5, 7:10)` y pulsad Enter. A continuación volved a teclear `R` y pulsad Enter. Como veis, lo que hemos hecho primero es crear un objeto con el nombre `R` por medio del comando `<-` (símbolo 'menor que' seguido del símbolo 'menos')[2], que sirve para crear objetos. Después, hemos pedido a R que nos imprima este objeto llamando a su nombre. Como lo hemos creado previamente, nos lo ha mostrado. La función `c()` que hemos utilizado significa *concatenado* y lo tendremos que utilizar de manera muy habitual cuando trabajemos con objetos

[2]Un atajo para crear este símbolo es con el teclado introduciendo Alt +- (signo menos).

Llamar e imprimir

Los términos llamar e imprimir serán habituales en este módulo. Cuando llamemos a un objeto de R quiere decir que lo pedimos a la consola. Para hacerlo, tenemos que teclear el nombre del objeto en la consola. Cuando R nos muestra este objeto en la consola, muchas veces diremos que nos lo imprima.

que tienen más de un valor. Para crear estas cadenas de valores introduciremos la letra *c*, seguida de paréntesis. Dentro de los paréntesis, separados por comas, pondremos los valores que queramos.

El símbolo : en R

Cuando queramos pedir una cadena larga de valores consecutivos no hace falta que introduzcamos cada uno de estos valores. El símbolo : es un atajo que nos permite introducir solo el inicio y el final. Si queremos imprimir los valores del 1 al 100, solo hace falta que tecleamos `1:100`.

La consola nos muestra los resultados de todas las operaciones que hacemos con R. En algunos casos, sin embargo, veremos que no nos devuelve ningún resultado. Esto pasa principalmente cuando creamos un objeto. Cuando anteriormente hemos creado el objeto `R`, no hemos recibido ninguna indicación en la consola conforme lo hemos creado satisfactoriamente. Esto, aunque pueda parecer lo contrario, es buena señal. Cuando creamos objetos, RStudio nos almacena el objeto en memoria, pero no nos lo muestra en pantalla. Para verlo impreso en la consola, tendremos que llamarlo una vez creado, o bien podemos poner todo el código entre paréntesis en el momento de crearlo para que nos lo guarde y nos lo imprima en el mismo momento. Haced la prueba escribiendo, por ejemplo, `(S <- c(1:100))`.

También hay que remarcar que R no guardará nada de lo que hagamos si no se lo indicamos explícitamente con el comando <-. Por ejemplo, cuando hemos efectuado la operación `sample(50)`, R nos ha mostrado el resultado en la consola, pero no nos ha guardado el resultado en forma de objeto. Para guardarlo, tenemos que introducir primero el nombre que tendrá el objeto, después el comando <- y finalmente teclear el contenido que queramos que tenga el objeto. Haced el siguiente ejercicio: cread e imprimid al mismo tiempo un objeto llamado `muestra` que contenga `sample(50)`.

Cada orden completa que escribimos en la consola se llama línea de comando. Nos referiremos indistintamente tanto por este nombre como por el de código. Para que R nos lea cualquier código o línea de comando en la consola, siempre tendremos que pulsar la tecla Enter al final. R lo leerá, lo procesará y nos devolverá los resultados.

2.2. Script

El guion o script es un documento con el cual podemos anotar y transmitir a R nuestras indicaciones. A partir de ahora, en este módulo dejaremos de dar las indicaciones por medio de la consola y lo haremos con el script, puesto que es como lo haremos habitualmente cuando trabajemos con R. A pesar de que las funciones del script y la consola son parecidas, en el script podemos registrar todas nuestras operaciones y guardar líneas y líneas de anotaciones. Tenemos que considerar el script como un documento de texto que podemos usar de borrador para apuntar y modificar tantas veces como haga falta nuestros códigos de R. Apuntarlo en el script nos permitirá preparar, modificar, guardar y compartir códigos más fácilmente.

Veréis cómo todas las líneas del script están numeradas en la parte izquierda. A continuación tenéis, en el siguiente cuadro de color gris, varias líneas de comandos. Copiadlas todas y pegadlas a la primera línea numerada de código de vuestro script. Una vez pegadas, situad el cursor en esta primera línea de código. A diferencia de la consola, para ejecutar un código o línea de comando en el script tenemos que pulsar Ctrl+Enter en los PC y Command+Enter en los Mac [3]. Cada vez que ejecutáis una línea de comando en el script, el cursor saltará al inicio de la línea siguiente. Ejecutad todas las líneas del script hasta llegar a la última.

[3]Si queremos ordenar a R que lea todo el script, de arriba abajo, tendremos que pulsar Ctrl+Shift+Enter en los PC y Ctrl+Shift+Command en los Mac.

```
#Base de datos Mtcars
mtcars
hist(mtcars$mpg) #Histograma de la variable mpg
plot(mtcars$disp, mtcars$wt)
plot(mtcars$   disp,
   mtcars$qsec)
```

En este simple proceso hemos realizado varias cosas que tenemos que tener en cuenta para saber cómo funcionan las órdenes que damos al script. Lo primero que tenemos que saber es que el símbolo `#` (almohadilla) desactiva todo el resto de información que haya en la misma línea de comando. Esto sucede tanto en la primera línea con `#Base de datos Mtcars` como la tercera línea, donde a partir de la aparición de la almohadilla, R entenderá que tiene que dejar de leer el código en esta línea y pasar a la línea siguiente. La almohadilla, pues, nos será muy útil para hacer anotaciones. Podemos utilizarlo, por ejemplo, como título para indicar las operaciones que vendrán a continuación, y también podemos utilizarlo al final de una operación para dar algún detalle sobre la línea en cuestión.

La segunda cosa que hemos podido observar en el ejemplo anterior es cómo funcionan los espacios. Fijaos en que en la cuarta línea del código anterior no hay espacio entre `$` y disp , mientras que en la quinta línea estos dos términos sí que están separados por varios espacios. Tenemos que saber que R no lee espacios, de forma que tanto en una línea como en la otra R nos ha leído el mismo objeto `mtcars$disp`. Los espacios nos son útiles para poder leer e interpretar los códigos con más facilidad, pero R no los tiene en cuenta cuando lee un código. Por ejemplo, para R será exactamente lo mismo el código `v<-c(3,2)` que el código `v < - c (3 ,2)`.

Finalmente, hay que señalar que cuando pedimos a R que lea una línea, el programa nos saltará automáticamente a la siguiente si interpreta que el código no está finalizado. Los códigos de R no acaban nunca con una coma. Como la última línea del código anterior acaba con una coma, R ha pasado a leer automáticamente la línea siguiente. Hay otros símbolos, como un paréntesis abierto (`(`), una comilla abierta (`"`) o un símbolo de suma (`+`) que requieren

código a su derecha. R interpretará que tiene que continuar leyendo y/o saltar de línea si es necesario. Los saltos de línea nos son útiles para organizarnos el código y facilitar nuestra lectura.

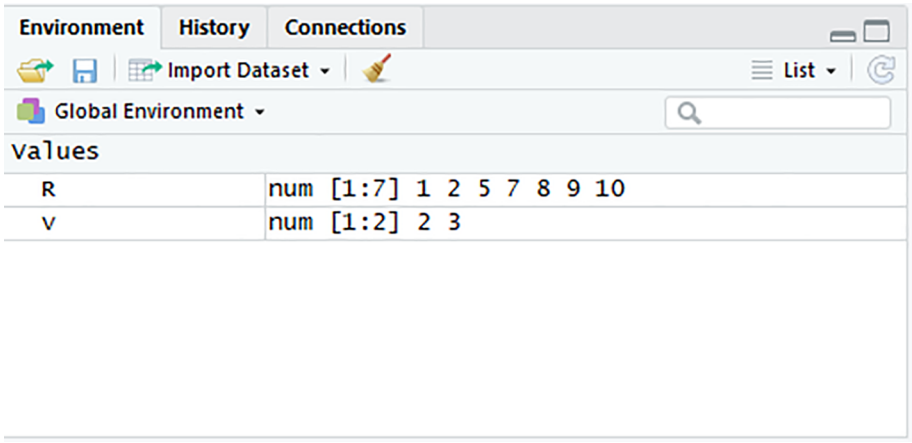
También es importante saber que si el cursor está situado en cualquier posición en una línea de comando del script, R leerá toda la frase entera (y esto puede comportar que R también nos lea líneas superiores o inferiores). R irá a buscar donde está el principio y el final del código completo y nos lo leerá entero. Si ve que está incompleto, nos devolverá error. Si en lugar de situar el cursor en una parte del código, seleccionamos una parte de la línea de comandos iluminándola con el cursor, R solo nos leerá esta parte. Probad, del código anterior, de seleccionar solo `mtcars$disp`. Después haced lo mismo con `mtcars$qsec`. R nos leerá únicamente la selección (siempre que tenga sentido gramatical para R).

Utilizar el script para apuntar todos los pasos que vamos siguiendo cuando creamos y desarrollamos códigos es muy útil tanto para nosotros como para terceras personas. Nosotros nos aseguramos de conservar en detalle todo el procedimiento que hemos seguido para generar unos datos determinados. Así, podemos repetir el procedimiento en cualquier momento, cambiarlo o bien examinarlo para detectar posibles errores. Anotar los códigos en el script también ayuda a nuestro rol de analista de datos de cara a terceras personas. Si queremos compartir nuestro trabajo, no hará falta que guardemos y enviemos varias tablas y gráficos. A veces tampoco será necesario que guardemos una base de datos. En un simple script, que ocupa poquísimo en nuestro ordenador, tenemos detallado todo el procedimiento, de forma que la otra persona solo tendrá que reproducir en su ordenador los códigos que hemos preparado.

2.3. Environment

En la ventana Environment o entorno podemos consultar todo lo que R tiene almacenado en memoria. En la ventana principal encontramos los objetos que ha creado. Si hemos seguido el ejercicio anterior de crear los objetos `R` y `v`, R nos mostrará estos dos objetos en la ventana junto con una breve descripción. Como veis en la figura 8, `R` es un objeto numérico de siete caracteres, mientras que `v` es un objeto también numérico pero de dos caracteres. Desde la consola o el script podemos obtener información de los objetos creados con las funciones `ls()` u `objects()`.

Figura 8. Global Environment.



Los objetos también se pueden eliminar. Para eliminar uno concreto, introducid el nombre del objeto dentro de la función `rm()` o `remove()`. Se pueden eliminar todos los objetos con la función `rm(list=ls())`.

Aparte de objetos, R también puede almacenar otros tipos de datos. Si clicáis en el apartado de 'Global Environment', os aparecerá un desplegable donde podréis comprobar los paquetes instalados en el entorno de R. Un paquete o librería se refiere al mismo concepto, un grupo de funciones y objetos, y nos referiremos a ello en este módulo de las dos maneras (ved el apartado Paso 3: Paquetes). Dentro del desplegable de 'Global Environment' veréis las siete librerías con que R viene por defecto: *stats*, *graphics*, *grDevices*, *utils*, *datasets*, *methods* y *base*. Más adelante añadiremos otras librerías en este desplegable. También podemos consultar las librerías cargadas con la función `search()`.

Utilizar archivos de Excel, Stata y SPSS

Las librerías más comunes para bajar bases de datos de Excel son *readxl* y *gdata*, mientras que para Stata y SPSS utilizaremos los paquetes *haven* o *foreign*.

Encima del apartado 'Global Environment' encontramos 'Import Dataset'. Con este apartado podremos importar diferentes bases de datos a R. Lo más sencillo es bajar ficheros con extensión *.csv* o *.txt* . Si queréis importar ficheros en formato Excel, Stata o SPSS, R os pedirá que bajéis la librería necesaria para poderlos importar en cada caso. Lo ideal es importar datos usando código, puesto que nos permite replicar un comando en otros ordenadores, pero también es útil saber que podemos importar manualmente bases de datos desde la ventana Environment.

Las otras pestañas de esta ventana no las usaremos. 'History' hace una recopilación de los procedimientos que ha hecho RStudio en la sesión actual. Podéis echar un vistazo para ver el historial de operaciones que habéis hecho hasta ahora.

2.4. Files

Como veis, esta ventana tiene varias pestañas: 'Files', 'Plots', 'Packages', 'Help' y 'Viewer'. En la pestaña 'Files' nos tendría que aparecer la carpeta de nuestro ordenador donde R irá por defecto a buscar los archivos, que acostumbra a ser la carpeta 'Documents' o 'Documentos'. R necesita una carpeta de nuestro

ordenador para operar. Será donde nos guardará las bases de datos que exportamos y el primer lugar donde irá a buscar las bases de datos que queramos importar. Esta carpeta la denominaremos directorio de trabajo. Podemos consultar el contenido del directorio de trabajo por medio de `dir()` y su ubicación dentro de nuestro ordenador con `getwd()`.

Si tenemos una carpeta especial donde queramos tener todos los archivos que vamos generando con R, será conveniente indicarla como directorio de trabajo para facilitarnos el trabajo. Podemos, por ejemplo, ir a la carpeta de la asignatura que tengamos creada en nuestro ordenador y crear dentro una nueva carpeta llamada 'R Docs'. Una vez estemos en esta nueva carpeta, tenemos que especificar a R que queremos fijarla como directorio de trabajo. Para cambiar manualmente la ubicación del directorio de trabajo, lo podemos hacer mediante los tres puntos suspensivos (...) que encontramos en el extremo derecho de la ventana 'Files'. Una vez hayamos seleccionado el nuevo directorio de trabajo, podemos ir al engranaje azul de la ventana 'Files' y marcar 'Set As Working Directory'. Este proceso también se puede hacer por medio del script con la función `setwd()`, en que dentro de los paréntesis especificaremos la dirección de la carpeta. Por ejemplo, ponemos poner `setwd("C:/RRII/Indicadores/R")` en PC y `setwd("H:\\myfolder\\data")` con Mac. De manera semiautomática tenemos la opción `setwd(choose.decir())`.

Ya hemos visto la utilidad de la segunda pestaña, 'Plots'. Cuando R crea un gráfico, lo visualizaremos automáticamente en esta ventana.

La pestaña 'Packages' es extremadamente importante en nuestros primeros pasos con R para familiarizarnos con el sistema de paquetes (aunque a medida que ganamos habilidad con el programa la iremos olvidando progresivamente). En este apartado podemos visualizar todos los paquetes instalados en RStudio de manera parecida a como hemos hecho cuando visitábamos 'Global Environment'. La lista que vemos son los paquetes que R tiene instalados, y encontramos el nombre del paquete, una breve descripción y la versión. También podemos visualizarlos en la consola con `installed.packages()`. Que un paquete esté en esta lista no quiere decir que R lo esté usando. R puede tener un paquete instalado, pero no cargado. Por lo tanto, tenemos que diferenciar dos procesos:

- *Paquetes instalados:* los paquetes que se encuentran en la lista que aparece en la ventana son los que RStudio tiene descargados. Esto no quiere decir que los esté utilizando.
  - De manera manual, podemos instalar un paquete nuevo clicando en el apartado 'Install'. Nos aparecerá una ventana donde podemos introducir el nombre del paquete que queremos que instale. Para instalar los paquetes básicos que nos piden en la guía de la asignatura, buscaremos *dplyr*, *ggplot2* y *tidyr* en la lista y, si no los encontramos, iremos al apartado 'Install' para instalarlos.

  - En el script, podemos utilizar la función `install.packages()` para instalar paquetes. El nombre del paquete tiene que ir siempre entre comillas. Si queremos instalar a la vez más de un paquete, lo tendremos que hacer con el concatenado. Para instalar los paquetes que nos piden en la guía de la asignatura, utilizaremos: `install.packages(c("dplyr", "ggplot2", "tidyr"))`.

- *Paquetes cargados:* para que R pueda trabajar con el paquete, tiene que estar cargado. Si nos encontramos con problemas cuando hagamos una determinada operación con R, es muy posible que el paquete no esté cargado. Lo podemos cargar de dos maneras:
  - Manualmente, activando el paquete que nos interese de la lista con el cuadradito que aparece a su lado. R nos lo cargará automáticamente. Si no aparece en la lista, lo tendremos que instalar (ved el apartado anterior).

  - Una segunda opción para instalar y cargar los paquetes es desde la consola o el script. Para hacerlo, utilizaremos la función `library()`. No podemos cargar varios paquetes a la vez. Los tendremos que cargar individualmente. Por ejemplo: `library(dplyr)`, `library(ggplot2)`, `library(tidyr)`.

El mejor amigo de un usuario de R es la ventana 'Help'. En este apartado podremos pedir información sobre un paquete, una función o un objeto determinado. Solo hace falta que pongamos el símbolo `?` antes del nombre del cual queremos pedir información. Por ejemplo, `?sum`, `?dplyr` o `?datasets`. En esta fase todavía no nos será de gran utilidad, pero a medida que nos familiaricemos con R descubriremos el gran valor que tiene.

Finalmente, tenemos la ventana 'Viewer', que por ahora no utilizaremos.

3. Tercer paso: los paquetes

Las librerías o paquetes son los manuales que consulta R para poder trabajar. Para hacer un símil con los estantes de libros que tenemos en casa, diremos que después de instalar R en nuestro ordenador, el programa tiene en su estudio una estantería gigante con la mayoría de estantes vacíos. Solo tiene unos cuantos libros básicos que ya vienen incorporados en el programa y que R ya se sabe de memoria. Estos libros son los que podemos ver inicialmente en la pestaña de Global Environment o bien si tecleamos `search()`. Con lo que ha aprendido de estos libros, R sabe hacer operaciones básicas como por ejemplo sumar y restar, crear gráficos sencillos y algunas cosas más.

Cuando damos la orden `install.packages()`, lo que hacemos es decir a R que vaya a buscar nuevos libros y los coloque en los estantes del estudio. A medida que instalamos más libros o paquetes, la librería se nos va llenando de material, aunque R solo los tiene colocados decorativamente. Ocupan espacio en los estantes (en nuestro disco duro), pero ¡cuidado, tener muchos libros no quiere decir que R los haya leído! R no los leerá hasta que se lo indiquemos.

El paso siguiente es decir a R que nos lea de arriba abajo algunos de los libros con la función `library()`. Los libros son como manuales de instrucciones. Cuando decimos a R `library(jardineria)`, el programa lee el manual de jardinería con décimas de segundo. De repente, R ya sabe absolutamente todo lo que se tiene que hacer para ser un buen jardinero. Si le decimos `library(carnetB)`, R se sacará el carnet B de conducir en pocas décimas de segundo y estará preparado para pilotar el coche que le indiquemos.

La cuestión es, pues, llenar la librería de libros que necesitemos para tenerlos fácilmente al alcance e indicar a R que lea los manuales que utilizaremos para hacer las operaciones que necesitaremos. De base, R ya tiene algunos conocimientos. Sabe sumar, restar, hacer varias operaciones, crear objetos, leer bases de datos, dibujar algunos gráficos..., pero nosotros le pediremos mucho más. Principalmente, le diremos que aprenda a hacer tres tareas. La más importante es la de manipular bases de datos. R aprenderá a crear y eliminar columnas y filas, ordenar filas o hacer operaciones estadísticas más sofisticadas. Esto nos lo permitirá un libro cuyo lomo dice *dplyr*. Una segunda tarea que también nos será muy útil es la de dibujar gráficos complejos. Esto nos lo permitirá hacer el manual *ggplot2*. Finalmente, también utilizaremos *tidyr*, un paquete que manipula las filas y las columnas de las bases de datos. Hay más paquetes que utilizaremos, como por ejemplo *readxl*, *stringr* o *lubridate*, pero ya hablaremos de ellos más adelante.

El CRAN es nuestra biblioteca central. Cuando le decimos `install.packages()`, R va a la biblioteca, coge los libros que le decimos y los deja en el estante del estudio. Cuando le decimos `library()`, R va al estante y lee el manual que le decimos. En el apartado anterior ya hemos aprendido a diferenciar entre estos dos procesos y también hemos conocido las dos maneras que hay para bajar paquetes: por medio de la ventana 'Packages' o por código mediante el script o la consola. Es posible que primero nos sintamos más cómodos usando la ventana, pero más adelante a buen seguro que nos sentiremos más cómodos con el código.

Diferencia clave entre instalar y cargar

Es muy importante saber que, cuando instalamos un paquete, R se lo guardará en memoria y no necesitaremos volverlo a instalar cada vez que iniciemos sesión en R (salvo que aparezca una nueva versión que queramos utilizar, caso en que sí que lo tendremos que instalar de nuevo). En cambio, cuando salgamos de R los paquetes dejarán de estar cargados. La próxima vez que volvamos a ejecutar el programa, tendremos que volver a cargar los paquetes. Por lo tanto, tened en cuenta, de manera sistemática, que hay que poner estos tres códigos siempre que entréis en R.

```
library(dplyr)
library(tidyr)
library(ggplot2)
```

4. Cuarto paso: script

Ahora que ya conocemos las partes esenciales de la interfaz de R, ya estamos preparados para empezar a trabajar con los materiales de la asignatura. En este apartado utilizaremos, como ejemplo, los scripts que figuran en el módulo inicial de la asignatura *Fuentes de información e indicadores en estudios internacionales*, aunque podrían valer los de cualquier asignatura. Lo primero que podemos hacer es organizarnos el script a nuestro gusto y empezar a entrar los códigos que tenemos en los materiales docentes. En la figura 9 encontraréis una propuesta de cómo empezar.

Figura 9. Organización del script.

En primer lugar, hemos puesto un título de encabezamiento precedido por la almohadilla (#). Como sabéis, cuando RStudio ve que hay una almohadilla delante de una línea, entiende que no tiene que leer nada a partir del símbolo que haya en la misma línea. A nosotros, en cambio, nos irá bien para hacer anotaciones en el script. En `#INSTALL` instalamos los tres paquetes básicos que utilizaremos. Recordad que los paquetes se instalan con comillas. Mucho cuidado si copiamos y pegamos códigos con comillas desde programas como por ejemplo Word. Normalmente, estos programas redondean o inclinan las comillas[4]. Si enganchamos estas comillas modificadas a RStudio, el programa no entenderá qué símbolo son y nos dará error.

[4]Fijaos en la diferencia entre escribir `"dplyr"` o `“dplyr”`. R reconocerá el primer término pero nos dará error en el segundo caso.

Para instalar los paquetes tenemos que estar conectados a internet, puesto que R va a buscarlos en la biblioteca CRAN. Veremos que la instalación de algunos paquetes puede tardar un rato, en especial la de *ggplot2*. Paciencia. Sabréis si R todavía está procesando si hay una señal roja de stop en el margen superior izquierdo de la ventana de la consola.

Una vez instalados los paquetes, los tendremos que cargar o activar. Podemos hacer la misma operación que antes y marcar con una almohadilla # el inicio de otro proceso. En este caso, hemos puesto `#PACKAGES`. Los paquetes se tienen que cargar por separado: `library(tidyr)`, `library(dplyr)` y `library(ggplot2)`. Normalmente solo necesitaremos instalar una sola vez

los paquetes. Después ya se quedan guardados en nuestro ordenador. Lo que sí que tendremos que hacer es cargarlos cada vez que iniciemos una nueva sesión.

La mejor manera de comprobar que R nos ha instalado correctamente el paquete será utilizándolo. En la tabla 3 se indica cómo comprobar que los paquetes *ggplot2* y *dplyr* funcionan con dos operaciones hechas mediante funciones de estos paquetes: crearemos un histograma con *ggplot2* y filtraremos unos datos con *dplyr*.

Tabla 3. Prueba de los paquetes *ggplot2* y *dplyr.*

Prueba ggplot2 `qplot(mtcars$mpg)`
Prueba dplyr `filter(mtcars, mpg > 25)`

Si en el primer código nos aparece un gráfico significará que tenemos *ggplot2* funcionando correctamente. Si en el segundo código nos aparece en la consola una tabla de seis filas, también significará que el paquete *dplyr* funciona correctamente.

Con las indicaciones dadas hasta aquí ya podéis repetir las instrucciones del manual de la asignatura en vuestro ordenador. Personalizad el script a vuestra manera e id introduciendo las líneas de comandos.

5. Quinto paso: posibles errores

No hay programador o analista de datos, por más que sepa, que no cometa decenas de errores cuando utiliza R. Las horas invertidas en el programa y nuestra familiarización con el entorno, manuales de ayuda o foros determinarán nuestra capacidad de poder resolver estos errores. Por lo tanto, al principio tendremos que tener mucha paciencia, puesto que todavía tendremos pocos recursos para poder resolver las situaciones que nos sobrevengan.

En la tabla 4 se ofrece una guía de los problemas más habituales con que el usuario se puede encontrar en las primeras horas de R. Obviamente, no es una lista exhaustiva, pero nos puede ayudar a identificar y resolver más de un error.

Tabla 4. Errores principales que os podéis encontrar con R.

1	No haber instalado RStudio ni R. Trabajaremos con RStudio, pero también tendremos que tener instalado R para que RStudio funcione adecuadamente.
2	No reconoce parámetros como `%>%`, `group_by()` o `summarize()`. No está instalado el paquete *dplyr*.
3	Algunos errores se dan porque no tenemos las librerías correspondientes instaladas o bajadas. Comprobad que tengáis todas las librerías necesarias instaladas y también cargadas para hacer el ejercicio. Podéis comprobar en la ventana inferior derecha 'Packages' cuáles son las librerías instaladas (las que figuran en la lista de la ventana) y cuáles son las que están cargadas (las que tienen un signo √ al lado).
4	Si copiamos los datos primero en un Word y después en RStudio nos podemos encontrar un problema de formato. Por ejemplo, Word modifica las comillas `"_"` de los códigos para hacerlas más redondeadas o inclinadas, y en este formato RStudio no entenderá el significado.
5	Es un error muy común copiar los datos de manera inexacta. Fijaos bien en la orientación de las barras, puesto que no es lo mismo `\` que `/`. Haced un recuento del número de paréntesis que abrís en una línea de código, puesto que por cada paréntesis (que abrimos tendrá que haber otro) que cierre.
6	Cuando en la consola nos sale + es que R espera instrucciones, que no hemos acabado bien el código y que posiblemente falta cerrar un paréntesis o unas comillas.
7	No pongáis nunca un `#` ante una línea de código. Las líneas de código que R tiene que interpretar tienen que estar libres delante.
8	R es sensible a las mayúsculas. `"SPAIN"` no es lo mismo que `"Spain"`. Teclead `"SPAIN" == "Spain"` y observad el resultado.

A medida que vamos ganando más experiencia con R, cada vez seremos más capaces de entender cuál es el error si leemos el texto que R nos devuelve en la consola. Por ejemplo, con "`unexpected symbol`" o "`unexpected end of input`" es muy posible que haya algún problema de tecleo en el código y que tengamos que revisar que todos los paréntesis estén cerrados, los caracteres

entre comillas y los nombres de los objetos y las funciones entrados correctamente. Si en la consola aparece un "+" es porque R espera que añadamos algún comando más para completar el código.

6. Glosario

Glosario de atajos con el teclado (PC)

`Alt+-`	Atajo por <-
`Ctrl+Enter`	Ejecuta una línea de código en el script
`Ctrl+S`	Guarda el script
`Ctrl+Shift+Enter`	Ejecuta todo el código del script
`Ctrl+Shift+N`	Crea un script
`Enter`	Ejecuta una línea de código en la consola

Glosario de funciones y operadores del módulo

`<-`	Crea un objeto
`?`	Abre el cuadro de ayuda de una función u objeto
`??`	Hace una búsqueda por palabra clave
`$`	Establece la separación entre el marco de datos y la variable.
`c()`	Crea un vector
`dir()`	Muestra los archivos del directorio de trabajo
`getwd()`	Muestra la ubicación del directorio de trabajo
`help()`	Abre el cuadro de ayuda de una función u objeto
`installed.packages()`	Muestra las librerías instaladas
`install.packages()`	Instala una o más librerías
`library()`	Muestra los paquetes instalados
`ls()`	Muestra los objetos creados
`objects()`	Muestra los objetos creados
`remove()`	Elimina un objeto creado
`rm()`	Elimina un objeto creado
`rm(list=ls())`	Elimina todos los objetos que hemos creado
`sample()`	Devuelve una muestra aleatoria
`search()`	Muestra los paquetes cargados en R.
`setwd()`	Establece la ubicación del directorio de trabajo

Resumen

En este módulo hemos visto cómo introducirnos en RStudio a través de cinco pasos principales. Los contenidos que hemos estudiado en las páginas anteriores no pretenden ser una guía exhaustiva sobre el uso del programa, sino que tienen como objetivo principal que el estudiante se pueda mover por RStudio con suficiente habilidad para poder trabajar los materiales y las actividades de una asignatura que utilice el entorno de R como herramienta docente.

Es importante completar este aprendizaje con otros materiales, principalmente con vídeos que encontraréis disponibles en el aula y también con los recursos que creáis necesarios. Tened en cuenta que R es un software libre con una comunidad de usuarios muy grande y que hay muchísimos sitios en la red donde podéis encontrar tutoriales o guías de ayuda.