Introducción a R

Florencia Hnilo

Marzo de 2019

Introducción a R

Objetivos del curso

- Entender y aplicar los conocimientos vistos en las clases teóricas
- Este no es un curso de R, solo se presentarán algunos comandos básicos del programa necesarios para realizar los trabajos prácticos
- A modo de ejemplo, una posible resolución del primer trabajo práctico está disponible en la página del curso
- Si surgen dudas, primero intentar solucionarlas investigando en Internet, ¡es la mejor manera de aprender! Si no logran resolverla, escribir a `fhnilo@udesa.edu.ar` con el asunto *Duda Econometría Exactas*.

Antes que nada, ¿qué es R?

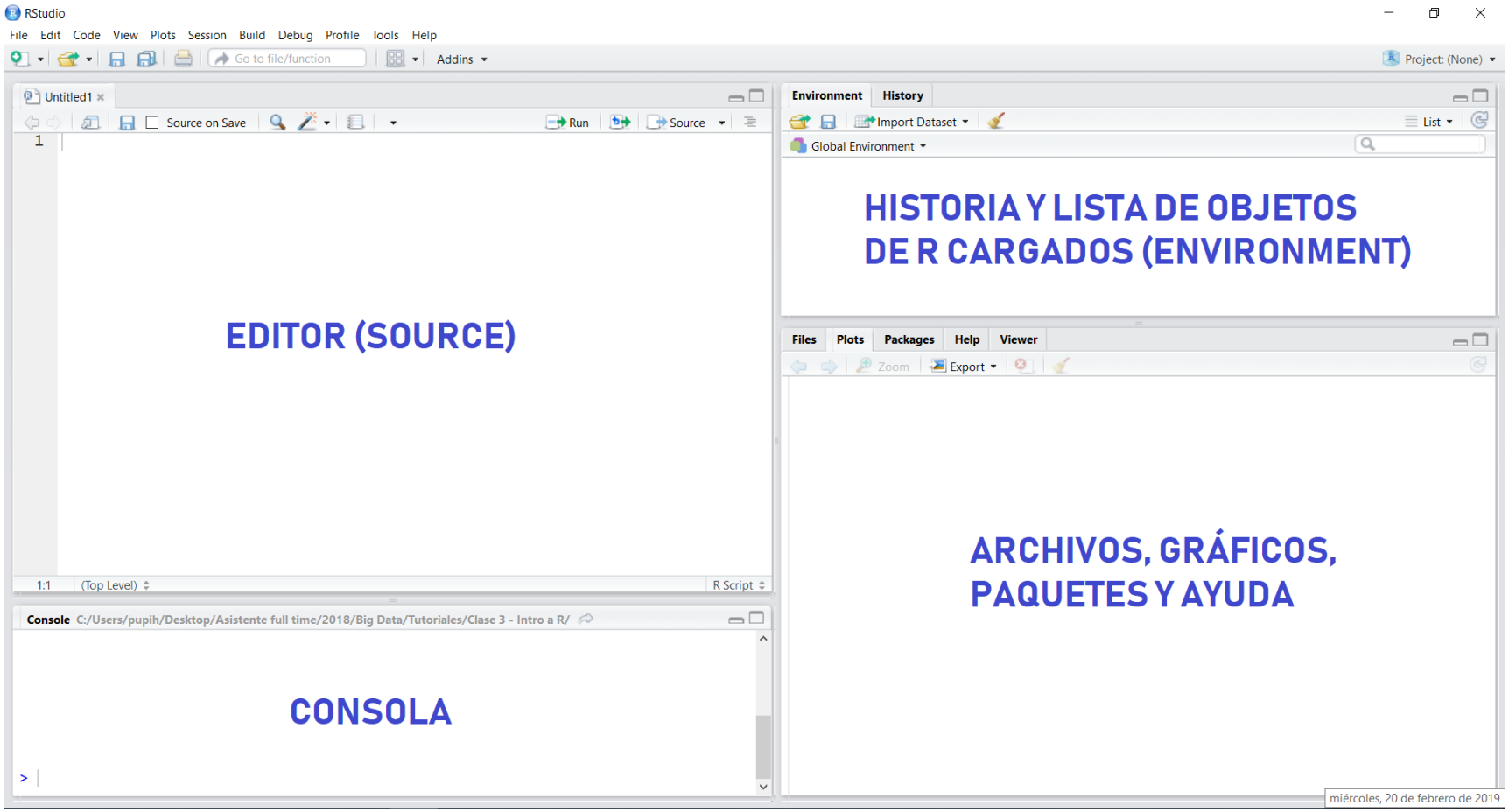
- Programa de computación estadístico
- Surgió como una simplificación del programa S-PLUS
- ¡Gratis! Gracias a eso, mucha información dando vuelta en la red. Ver por ejemplo
  1. `https://stackoverflow.com`
  2. `https://renbaires.github.io/`
  3. `https://cran.r-project.org/web/views/`

  Para libros sobre R, ver:
  1. `https://www.r-project.org/doc/bib/R-books.html`
  2. `https://bookdown.org/`

Descarga

- Descargar la versión 3.5.2 desde `http://mirror.fcaglp.unlp.edu.ar/CRAN/` según el sistema operativo que tengan.
- Si bien es posible trabajar directamente con R, la aplicación RStudio facilita la tarea. Descargar la versión gratuita desde `https://www.rstudio.com/` (elijan la versión recomendada para su sistema).

Consola de RStudio

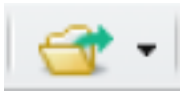


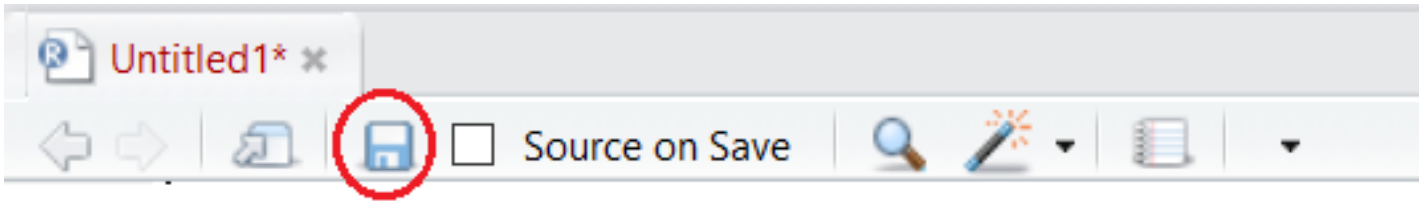
Consola de RStudio

- R es un programa que funciona a través de comandos. Requiere que se escriban comandos después del símbolo ">" en la consola y se ejecuten oprimiendo "Enter".
- Sin embargo, normalmente escribiremos los comandos en el editor pues tiene la ventaja de conservar a mano el listado de comandos. Para ejecutar un comando desde el editor, seleccionarlo y oprimir 

.
- Además, es posible tener varios editores abiertos de muchos proyectos distintos, lo que evita volver a escribir comandos que ya están en otro proyecto. Para abrir un proyecto nuevo, optimir Ctrl + Shift + N o bien ir a  y elegir "R Script".

Consola de RStudio

- Para abrir un proyecto guardado, oprimir  y buscar el archivo.
- Para guardar un script, oprimir el disquete



Tips

- La mayor parte de los errores se deben a escribir mal los comandos o los nombres de las variables. No es lo mismo escribir `Resultado` que `resultado`.
- Los comandos de R también se llaman funciones, y su estructura básica es:
  `objeto <- funcion(dataset, TRUE)`
- La flecha para arriba del teclado sirve para llamar comandos anteriores.
- El símbolo $ se usa para llamar a una columna particular de un dataset (ejemplo: dataset$columna_i).
- Suele ser útil escribir al lado o arriba de los comandos lo que queremos hacer. Para que R no lea esas líneas, insertar el símbolo # antes de ellas.

Directorio

Antes de empezar a trabajar en R, es necesario avisarle al programa cuál es nuestro directorio para que sepa dónde guardar los archivos o gráficos que resulten de nuestro trabajo. Para eso utilizaremos la función `setwd`, escribiendo entre paréntesis la dirección de nuestra carpeta de trabajo:

```
setwd("C:/workspace2") #o bien setwd("C:\\workspace2")
```

ATENCIÓN: fíjense que utilicé "/" o "\\". R no reconoce una única barra invertida "\".

Para chequear que indicamos bien el directorio, correr:

```
getwd()
```

R como una calculadora

R puede ser utilizado como una calculadora. Escribir en la consola los siguientes comandos oprimiendo "Enter" después de cada uno y observar el resultado.

- `5+3`
- `5*(7/3)`
- `sqrt(4)`

También es posible guardar los resultados de una operación en el Environment para utilizar más tarde:

```
resultado <- 5+3
print(resultado)
```

¡Cuidado! Si más tarde llamamos `resultado` a otro valor, perdemos el valor anterior sin previo aviso.

Tipos de datos

Podemos guardar otros objetos además de números:

```
resultado <- "Hola, me llamo Florencia"
class(resultado) #función que me permite saber el tipo de objeto con el que estoy tratando.
class(sqrt)
```

También podemos guardar vectores y combinarlos:

```
vector1 <- c(1,2,3)
vector2 <- c(1,4,77)
vector3 <- c(vector1, vector2)
```

Para acceder a un elemento específico de un vector, usar corchetes:

```
vector3[4]
```

Funciones útiles

- `length(vector3)`: para saber la cantidad de elementos del vector
- `min(vector3)`: valor mínimo del vector
- `max(vector3)`: valor máximo del vector
- `range(vector3)`: rango del vector
- `mean(vector3)`: media del vector
- `sd(vector3)`: desvío estándar del vector
- `sum(vector3)/length(vector3)`: otra forma de calcular la media del vector
- `rm(list = ls(all = TRUE))`: borrar todos los objetos guardados
- `rm(objeto)`: borrar un objeto particular

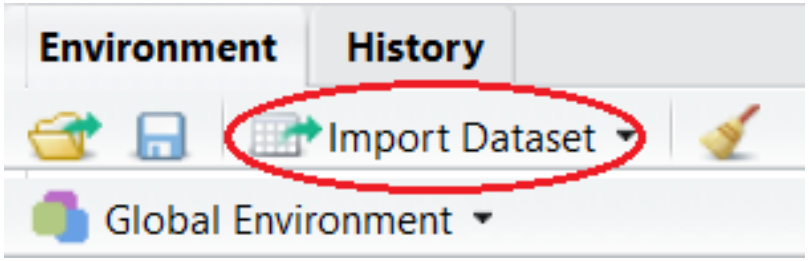
Importar bases de datos

Usualmente se trabaja con bases de datos que guardamos en formatos distintos a R. Dependiendo del tipo de archivo, el comando que deberemos usar para abrir la base.

- csv: `data <- read.csv("archivo.csv")` (si los datos están separados por punto y coma, usar la función `read.csv2`)
- txt: `data <- read.table("archivo.txt", header = TRUE)`
- xlsx: para abrir archivos de Excel, primero es necesario instalar el programa `readxl`. Correr `install.package("readxl")` y luego llamar al programa corriendo `library(readxl)`. Una vez hecho esto, correr `data <- read.excel("archivo.xlsx")`

Importar bases de datos

También es posible importar bases sin necesidad de escribir los comandos oprimiendo sobre



Elegir entonces el tipo de archivo que se quiere abrir.

Exportar bases de datos

Si quieren guardar una base de datos, lo mejor es hacerlo en formato csv (Excel tiene la desventaja de cambiar el formato de los datos en algunas ocasiones). Como ya definieron el directorio, el archivo se guardará en la carpeta elegida previamente:

```
write.csv(data, file = "data.csv")
```

En otros formatos:

- txt: `write.table(data, "data.txt", sep="")`
- xlsx:
```
library(xlsx)
write.xlsx(data, "data.xlsx")
```

Gráficos

El paquete más sencillo para hacer gráficos es `plot`. Para guardar un gráfico, es posible hacerlo desde la pestaña "Plots" de la derecha, o bien a través de comandos:

```
install.packages("ISLR") # instalo una base de datos para ejemplificar
library(ISLR) # cargo la base

png(file = "nombre.png") # nombro al archivo, que se guardará en el directorio especificado
plot(Auto$cylinders) # corro el comando con el gráfico
dev.off() # indico que hasta acá tiene que guardarse la información en el archivo nombrado más arriba
```

Gráficos

Usualmente se utiliza un paquete más avanzado para realizar gráficos: `ggplot2`. Para gráficos sencillos con este paquete, puede usarse el comando `qplot()`:

```
qplot(x, y, data=, color=, shape=, size=, alpha=,
geom=, method=, formula=, facets=, xlim=, ylim=,
xlab=, ylab=, main=, sub=)
```

Para más información sobre `qplot()`, ver `https://www.statmethods.net/advgraphs/ggplot2.html`, y para un buen resumen de todo lo que se puede hacer con este paquete, ver: `https://www.rstudio.com/wp-content/uploads/2016/11/ggplot2-cheatsheet-2.1.pdf`

¡A trabajar!