Ciencia de Datos y Políticas Públicas

Una introducción a la exploración, análisis y visualización de datos

Antonio Vazquez Brust

2019-03-24

Contents

Placeholder

Antes de empezar

Chapter 1

¿Qué es la ciencia de datos?

Placeholder

1.1 ¿Qué significa hacer ciencia de datos?

Chapter 2

Una presentación a toda marcha de R

`R` es un lenguaje de programación especializado en análisis y visualización de datos. Es un producto de código abierto, lo cual significa que cualquier persona puede usarlo y modificarlo sin pagar licencias ni costos de adquisición de ningún tipo.

Expertos de todo el mundo colaboran en forma activa con el proyecto, no sólo desarrollando el lenguaje en sí (llamado "R base"), sino también extendiéndolo con nuevas habilidades que pueden ser incorporadas por los usuarios finales en forma de "paquetes" instalables.

La calidad del lenguaje en sí, de los paquetes instalables que le agregan un sinfín de funciones (desde algoritmos de inteligencia artificial hasta mapas interactivos) y de la comunidad de usuarios que comparte información en foros y blogs, ha hecho de R uno de los lenguajes de programación más populares del mundo. En el campo del análisis de datos, es la herramienta por excelencia en muchas universidades, empresas de tecnología, y redacciones de periodismo de datos.

2.1 Nuestro primer proyecto en R

A continuación reproduciremos un ejercicio paso a paso, para ilustrar la potencia de una herramienta de análisis como R. Que nadie se preocupe si algunas de las operaciones parecen no tener sentido, o resultan arbitrarias. ¡Es normal! Nadie aprende un lenguaje en 10 minutos, sea R o esperanto. La idea es tener exposición temprana a un caso de uso interesante, usando datos reales. Y que nos sirva como motivación para practicar luego ejercicios básicos que son muy necesarios pero, a veces, no tan emocionantes.

2.1.1 Crear un proyecto en RStudio

El primer paso es ejecutar RStudio, que ya deberíamos tener disponible en nuestro sistema.

Una vez abierta la interfaz gráfica, creamos un proyecto nuevo, cliqueando en `File -> New Project... -> New Directory -> New Project`. En la ventana que surge, elegir un nombre para el proyecto (por ejemplo, "Practicando R") y finalizar la operación cliqueando en `Create project`.

Utilizar proyectos nos permite continuar otro día desde donde dejamos la tarea al terminar una sesión. Es sólo cuestión de recuperar el proyecto deseado la próxima vez que abrimos RStudio, cliqueando en `File -> Recent Projects -> "nombre de mi proyecto"`.

Por ahora, sigamos trabajando. Vamos a crear un "script". Un script, como su nombre en inglés lo indica, es un guión; una serie de pasos que escribimos para que nuestra computadora ejecute en secuencia. Cliqueamos en `File -> New File -> R Script`. De inmediato se abre una ventana con un editor de texto. ¡Ahora empieza la acción!

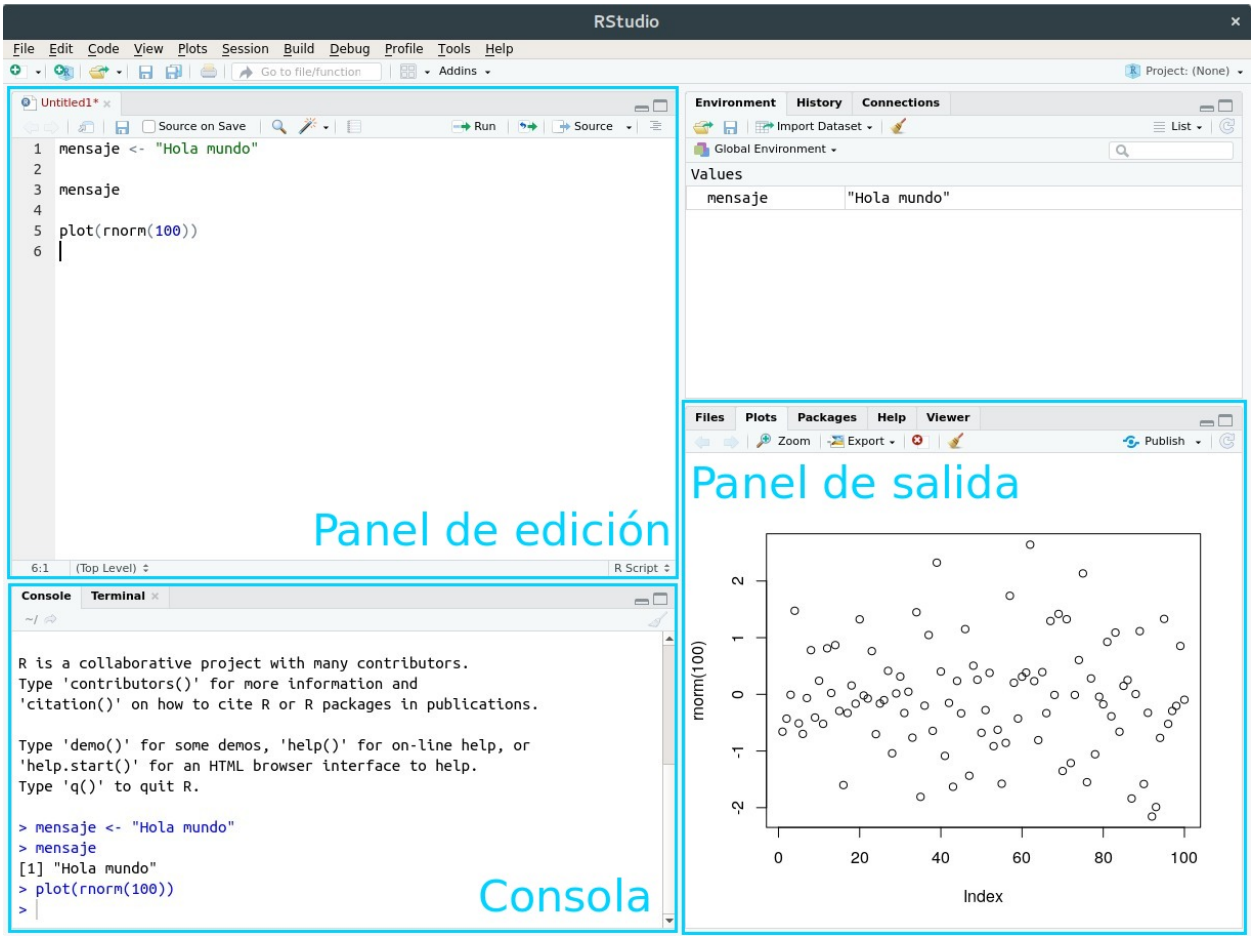

Figure 2.1: La interfaz de RStudio

2.1.2 Escribiendo un script

Aprovechemos para dar un nombre a los áreas que vemos en RStudio:

Vamos a escribir nuestro código (las instrucciones que `R` entiende) en el panel de edición. Los resultados van a aparecer en la consola (cuando se trate de texto) o en el panel de salida (cuando produzcamos gráficos)

Por ejemplo, podemos escribir el panel de edición la instrucción para mostrar el resultado de una operación matemático:

```
sqrt(144)
```

`sqrt()` es una *función*. En el mundo de la programación, las funciones son secuencias de código ya listas para usar, que realizan tareas útiles. Por ejemplo, mostrar algo en pantalla. En nuestro caso, completamos la función con algo más: un *parámetro*, pues así se le llama a los valores que una función espera de parte del usuario para saber que hacer. La función print espera que le demos un número para el cual calcular su raíz cuadrada (*square root* en inglés), y eso hicimos: le pasamos cómo parámetro `144`, un número. Los parámetros siempre se escriben entre paréntesis, a continuación del nombre de la función.

Ahora vamos a aprender la combinación de teclas más importante al usar RStudio: `Ctrl` + `Enter`. Presionar `Ctrl` + `Enter` al terminar de escribir una instrucción hace que RStudio la ejecute de inmediato, y espere en la siguiente instrucción, si la hubiera.

Cambien podemos buscar una línea que deseemos ejecutar, posicionando el cursor de texto (que luce como

una barra vertical que titila, en el panel de edición) sobre ella. Si a continuación pulsamos `Ctrl` + `Enter`, la línea será ejecutada y el cursor se moverá sólo hasta la siguiente línea, listo para repetir el proceso.

La modalidad de ejecución línea por línea es muy útil para lo que se llama "análisis interactivo". Uno ejecuta un comando, observa el resultado, y en base a eso decide su próxima acción: cambiar parámetros e intentarlo de nuevo, dar por buenos los resultados y usarlos para una tarea subsiguiente… etc.

Por ejemplo, si escribimos las siguientes líneas:

```
sqrt(144)

mensaje <- "Hola mundo"

mensaje
```

…y posicionamos el cursor en cualquier posición de la primera línea, para luego pulsar `Ctrl` + `Enter` tres veces, veremos que las instrucciones son ejecutadas línea a línea.

```
sqrt(144)
```

```
## [1] 12
```

```
mensaje <- "Hola mundo"
```

```
mensaje
```

```
## [1] "Hola mundo"
```

Dos de ellas (la primera y la última) mostraron una salida en pantalla, y la del medio, no. Esto es porque algunas funciones entregan algo como resultado algo -un número, un texto, un gráfico, u otros tipos de salida que ya veremos- mientras que otras hacen su tarea silenciosamente sin expresar nada. En este caso, la función silenciosa fue la de asignación: `mensaje <- "Hola mundo"` es una instrucción que le pide a R que cree una variable llamada "mensaje" (o que la encuentre si ya existe) y que le asigne como valor el texto "Hola mundo". ¿Cómo sabemos que la instrucción se llevó a cabo, a pesar de no producir una salida? En general, es un tema de confianza. Si una instrucción no genera un mensaje de error, si es silenciosa, se asume que pudo cumplir su cometido. En este caso, además lo hemos verificado. La línea final, `mensaje` pide a R que busque la variable, y muestre en pantalla su contenido (esa es una característica muy práctica del lenguaje: para saber el contenido de una variable, basta con escribirla y ejecutar la línea). Y al hacerlo, comprobamos que la variable contiene precisamente lo que hemos tipeado.

De paso, hay que mencionar que la creación y manipulación de variables es un concepto clave en programación. Trabajar con variables nos permite almacenar valores para usarlos después, además de hacer nuestro código más fácil de leer y compartir con otros, en especial cuando usamos nombre de variable auto-explicativos. Como ejemplo de ésto ultimo comparemos

```
x <- 8 * 6
x
```

```
## [1] 48
```

… con

```
ancho_habitacion_m <- 8
profundiad_habitacion_m <- 6
superficie_habitacion_m2 <- ancho_habitacion_m * profundiad_habitacion_m

superficie_habitacion_m2
```

```
## [1] 48
```

En su resultado ambas expresiones son iguales, dado que producen lo mismo. Pero la segunda esta escrita de