

# Fastbook 01

## Programación en R

Introducción a R y RStudio



# 01. Introducción a R y RStudio

Bienvenidos a Programación en R.

En el presente fastbook vamos a realizar nuestro primer contacto con esta disciplina.

Empezaremos entendiendo el importante papel que R juega en el mundo del análisis de datos e instalaremos en nuestros equipos el software necesario para poder trabajar en el contenido práctico de la asignatura. Tras ello, pasaremos a la acción y presentaremos los elementos esenciales que nos permitirán iniciar nuestra andadura como usuarios de R y RStudio.

¡Comenzamos!

*Autor: Juan Jiménez García*

[Introducción a la asignatura](#)

[Instalando](#)

[Conociendo el entorno](#)

[Tipos de variables](#)

[Operadores](#)

[Conclusiones](#)

# Introducción a la asignatura

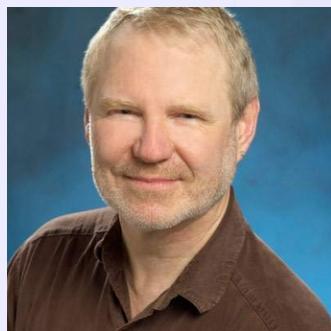
**X** Edix Educación

---

Si os encontráis aquí es porque habéis mostrado ilusión e interés por desarrollar vuestro conocimiento y carrera profesional en el apasionante mundo de analytics, en concreto, aquel que aplica al sector del marketing.

Dicho esto, es normal que puedan surgir muchas preguntas y dudas. **¿Para qué sirve la analítica de datos? ¿Qué utilidad tiene para las empresas? ¿Y para el sector del marketing?**

Para solventar dichas cuestiones, quiero empezar haciendo referencia a una famosa cita del escritor y experto **Stephen Few**.



“Numbers have an important story to tell. They rely on you to give them a voice. (Los números tienen una historia importante que contar. Ellos confían en ti para que les des una voz).”

- Stephen Few

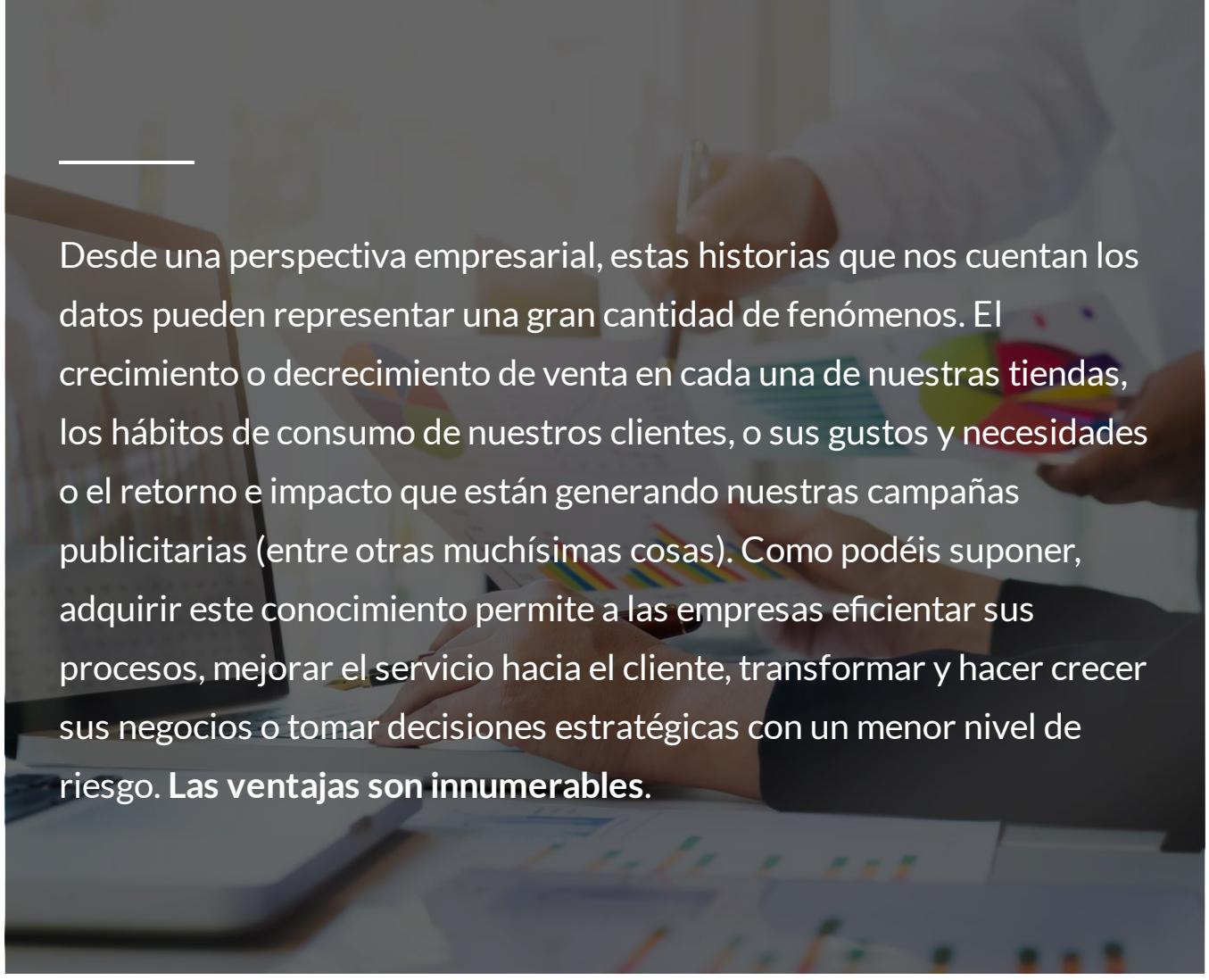
Esta frase me gusta, porque representa muy bien **cuál es la labor de un data scientist**.

---

El verdadero valor de los datos se encuentra en esa historia que tienen que contarnos, en el conocimiento que pueden aportarnos.

---

Para eso, no nos basta con adquirirlos o poseerlos, necesitamos transformarlos, modelarlos y visualizarlos con inteligencia y un objetivo claro.



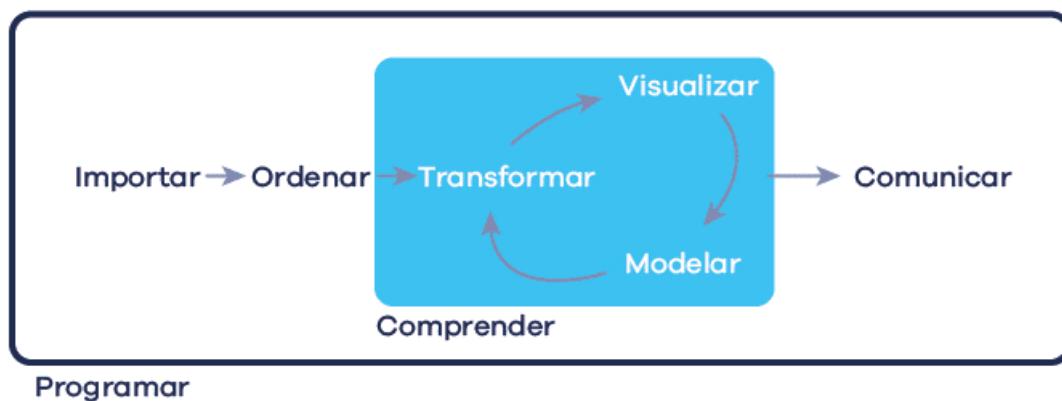
Desde una perspectiva empresarial, estas historias que nos cuentan los datos pueden representar una gran cantidad de fenómenos. El crecimiento o decrecimiento de venta en cada una de nuestras tiendas, los hábitos de consumo de nuestros clientes, o sus gustos y necesidades o el retorno e impacto que están generando nuestras campañas publicitarias (entre otras muchísimas cosas). Como podéis suponer, adquirir este conocimiento permite a las empresas eficientar sus procesos, mejorar el servicio hacia el cliente, transformar y hacer crecer sus negocios o tomar decisiones estratégicas con un menor nivel de riesgo. **Las ventajas son innumerables.**

---

En tu formación como experto en Marketing Data Analytics, nuestro objetivo es el de brindaros el conocimiento y las herramientas necesarias para que seáis capaces de llegar a este tipo de resultados en los que transformamos el dato de marketing en soluciones de negocio.

Una de esas herramientas es el lenguaje de programación R y su entorno de desarrollo RStudio, objeto principal y único de esta asignatura. Con su uso, seremos capaces de importar, ordenar, transformar, modelar y visualizar la información en aras de obtener conocimiento y comunicarlo de una forma efectiva y elegante.

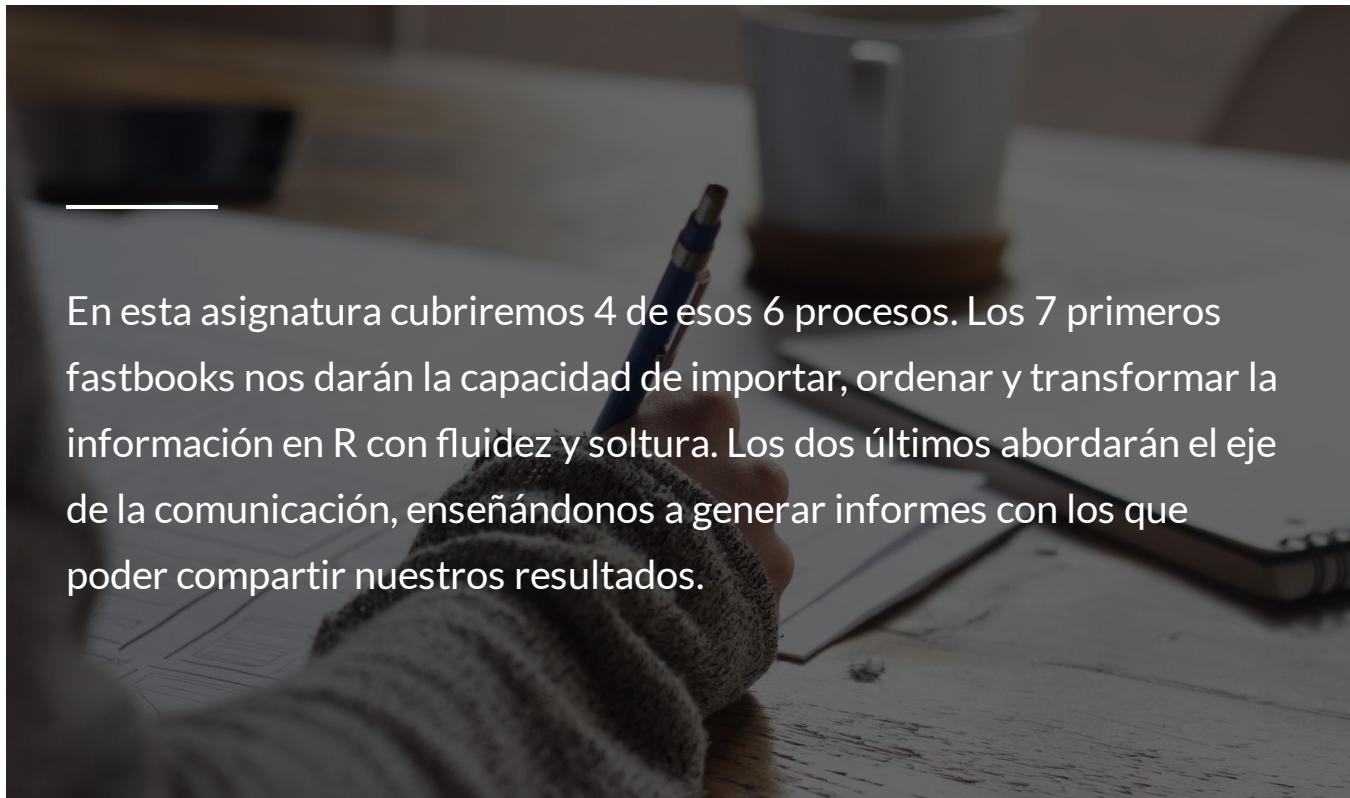
La siguiente imagen muestra el proceso habitual del análisis de datos:



Fuente: Hadley Wickham y Garrett Grolemund, *R for Data Science* (2016).

Como podéis observar, la etapa de comprensión es un ciclo cerrado de transformación, visualización y modelado de datos.

Esto es debido a que, en la gran mayoría de casos, cuando nos encontramos en una de esas tres etapas, los datos nos desvelan nuevos horizontes y nuevas preguntas que deben ser respondidas, para lo que necesitamos volver a transformar, visualizar y modelar. Son **tres tareas que se retroalimentan y comunican entre sí**, por lo que deben trabajar de forma unida con la finalidad de alcanzar un mismo objetivo.



En esta asignatura cubriremos 4 de esos 6 procesos. Los 7 primeros fastbooks nos darán la capacidad de importar, ordenar y transformar la información en R con fluidez y soltura. Los dos últimos abordarán el eje de la comunicación, enseñándonos a generar informes con los que poder compartir nuestros resultados.

Es importante dejar claro que **el conocimiento y las capacidades de programación se desarrollan con la práctica, tras horas y horas de prueba y error**. En ese sentido, os recomiendo que le deis mucha importancia a la práctica de los ejercicios que se van a ir realizando a lo largo de vuestra formación en R. Replicar y desarrollar las cosas por vosotros mismos os permitirá adquirir dominio en el contenido que vayamos abordando.

---

**Por último, solo me queda desearos un gran disfrute y éxito durante este aprendizaje, así como en vuestro futuro como analistas de datos y programadores en R.**

# Instalando

X Edix Educación

---

Es importante conocer las **diferencias entre R y RStudio**.

- R es el lenguaje de programación que vamos a desarrollar. Mediante código, indica a la máquina qué instrucciones debe ejecutar.
  - RStudio es el software que usamos para crear, editar, visualizar y ejecutar dicho código, es nuestro IDE (Integrated Development Environment).
- 

A continuación, vamos a proceder con la instalación de R y RStudio en nuestros equipos. Para ello, hemos preparado un tutorial que os ayudará en este proceso.



[Programación en R, fb. 01: instalación de R y RStudio](#)

# Conociendo el entorno

 Edix Educación

---

Tras haber finalizado con la instalación, vamos a definir los **elementos básicos que forman parte de R y RStudio** con el objetivo de ir ganando entendimiento y fluidez.

---

**Algunos de estos conceptos pueden resultar complejos cuando se estudian por primera vez. No os preocupéis por esto. Tras una visión teórica, pasaremos a un vídeo en el que los abordaremos desde una perspectiva mucho más práctica y visual.**

1

Ejecución de código R

Cuando trabajamos con lenguajes de programación, como R, Python o Java, generamos archivos de código que están compuestos por instrucciones que el ordenador debe ir ejecutando. Sin embargo, nuestros procesadores no son capaces de entender estos lenguajes de forma directa. Para ello, se debe realizar una traducción a lo que se conoce como **código máquina** o **lenguaje de máquina**.

Dicha traducción se puede realizar de dos formas:

- A través de un **compilador** que traduce de golpe todo el archivo de código a lenguaje de máquina y posteriormente pasa a ejecutarlo.
- A través de un **intérprete** que traduce el archivo de código línea por línea durante su ejecución. Es decir, traduce la línea 1 y la ejecuta, a continuación la 2 y la ejecuta, luego la 3... y así sucesivamente.

RStudio incorpora el intérprete de R a través de su consola. En ella, introducimos las instrucciones que son **traducidas y ejecutadas de forma inmediata**. Es un proceso totalmente transparente para nosotros.

Es interesante saber que también podemos interpretar y ejecutar nuestro código R desde el terminal de nuestro sistema operativo, aunque, en general, es una forma de trabajo poco común debido a su mayor complejidad.

2

## Script

Denominamos así a los **ficheros o archivos que contienen código de programación**. Al igual que en Excel los ficheros tienen extensión .xlsx, en programación R los scripts tienen extensión .R.

3

## Variables

Las variables son entidades que forman parte del código de programación y que **sirven para almacenar información**. Esta puede ser de cualquier tipo, desde un texto a una tabla hasta un vector numérico o una imagen.

4

## Operadores

Los operadores son símbolos que forman parte del código y que, usando las variables, nos permiten realizar acciones.

## 5

## Comentarios

Los comentarios son aquellas **partes de código que no son ejecutadas por el ordenador** y que nos sirven para explicar y ordenar nuestros scripts. Para comentar una línea de código R usamos el símbolo #.

## 6

## Funciones

Son fragmentos de código a los que les hemos asignado un nombre que les representa y que están compuestos por una serie de **operaciones que realizan una o más tareas**. Además, si resulta necesario, pueden recibir argumentos de entrada y devolver los correspondientes resultados.

Las reconocemos por la siguiente notación: *nombre\_de\_la\_función(argumentos)*.

Por ejemplo, la función *mean()* recibe un conjunto de números y devuelve su media.

Las funciones de R tienen tres **tipos de origen** que debemos diferenciar:

- Pueden venir incluidas con la instalación de R, lo que denominamos **R base**.
- Pueden estar contenidas en paquetes externos que tenemos que descargar e instalar.
- Pueden ser creadas, definidas y programadas por nosotros mismos.

7

## Sesión

Cuando iniciamos R, estamos creando una **instancia del entorno computacional de este lenguaje de programación**. ¿Esto qué significa? Por un lado, estaremos activando el intérprete de R para que pueda traducir las instrucciones de código al lenguaje entendido por la máquina. Por el otro, estaremos reservando un espacio de nuestra memoria RAM en el que alojaremos las variables y funciones que vayamos definiendo.

A esas instancias las llamamos sesiones de R.

Es posible tener más de una sesión en el mismo ordenador, pero las variables y funciones no se comparten, pertenecen a cada sesión de forma independiente.

8

## Paquetes

Los paquetes son **colecciones de funciones** desarrolladas por la comunidad de programadores y que se ofrecen a todo el público de forma gratuita. Para ello, deben pasar un riguroso control que les permite estar disponibles para el uso generalizado. Se encuentran alojados en los **servidores de CRAN** (*comprehensive R archive network*), de donde pueden ser descargados.

Para su instalación desde R usamos la función `install.packages()` con el nombre del paquete entre comillas.

Cuando queremos usar un paquete no basta con instalarlo, también tenemos que importarlo. Esto hace que las funciones definidas en él pasen a formar parte de nuestra sesión. Para ello, usamos la función `library()` con el nombre del paquete sin comillas.

9

## Directorio de trabajo

El directorio de trabajo es el **lugar o carpeta de nuestro ordenador** en el que se encuentran alojados los ficheros de datos, archivos de código y documentos que estamos utilizando.

Para saber cuál es el directorio de trabajo (*working directory*) que R está considerando, debemos ejecutar la función `getwd()`.

Si deseamos modificarlo, debemos usar la función `setwd()` con la ruta indicada entre comillas.

10

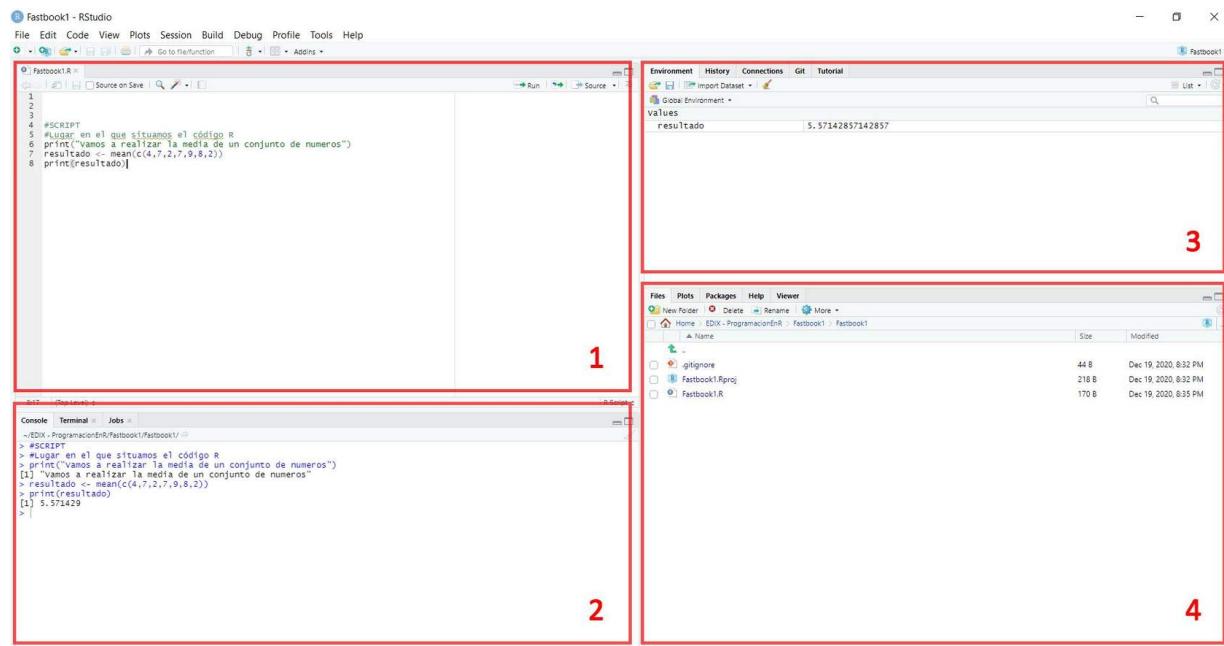
## Proyecto de RStudio

Con el objetivo de trabajar de una forma ordenada y lógica, RStudio contempla la creación de proyectos. Estos se sitúan en el directorio que elijamos, y los archivos contenidos en él pasan a formar parte del correspondiente proyecto.

11

## Entorno de RStudio

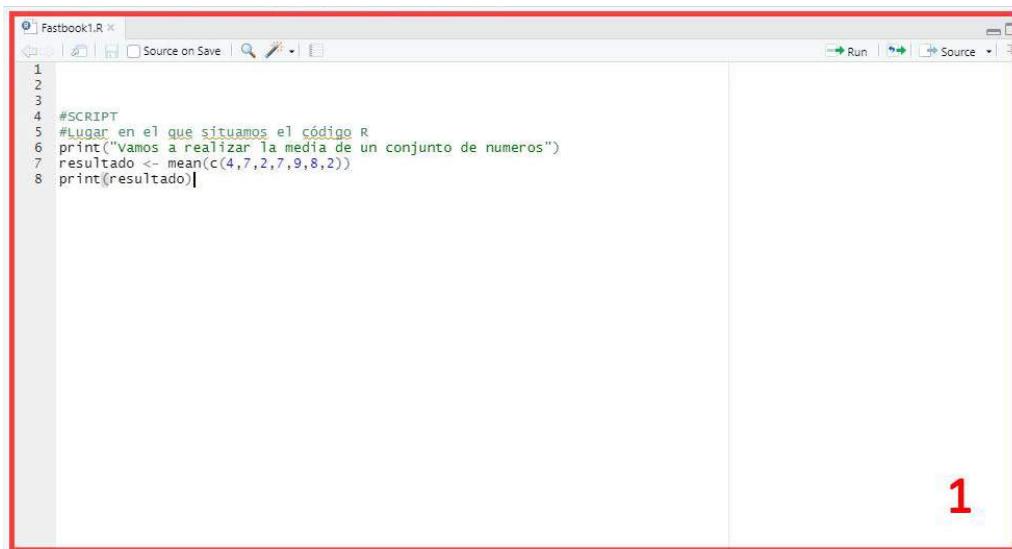
A continuación, vamos a visualizar la interfaz de RStudio con el objetivo de identificar sus componentes más importantes.



Como veis, su estructura estándar está dividida en 4 bloques:

### Bloque 1

Es el lugar en el que visualizamos y editamos los **scripts** de código.



The screenshot shows the RStudio interface with a code editor window titled "Fastbook1.R". The window contains the following R code:

```
1
2
3
4 #SCRIPT
5 #Lugar en el que situamos el código R
6 print("vamos a realizar la media de un conjunto de numeros")
7 resultado <- mean(c(4,7,2,7,9,8,2))
8 print(resultado)
```

A large red rectangle surrounds the entire code editor area, and a red number "1" is positioned in the bottom right corner of this rectangle.

### Bloque 2

Contiene la **consola**. Como ya hemos mencionado, es el lugar en el que se interpreta y ejecuta el código R, visualizando los correspondientes resultados.



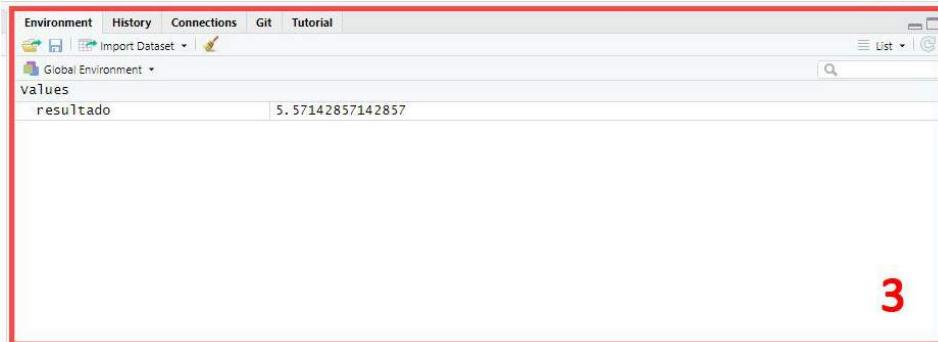
The screenshot shows the RStudio interface with a terminal window titled "R Script". The window displays the following R session history:

```
R> ~/EDIX - ProgramacionEnR/Fastbook1/Fastbook1/ ↵
> #SCRIPT
> #Lugar en el que situamos el código R
> print("vamos a realizar la media de un conjunto de numeros")
[1] "Vamos a realizar la media de un conjunto de numeros"
> resultado <- mean(c(4,7,2,7,9,8,2))
> print(resultado)
[1] 5.571429
> |
```

A large red rectangle surrounds the terminal window, and a red number "2" is positioned in the bottom right corner of this rectangle.

## Bloque 3

Aquí nos centraremos únicamente en el **environment**. En él se representan todas las variables y funciones que están definidas en la sesión actual.



## Bloque 4

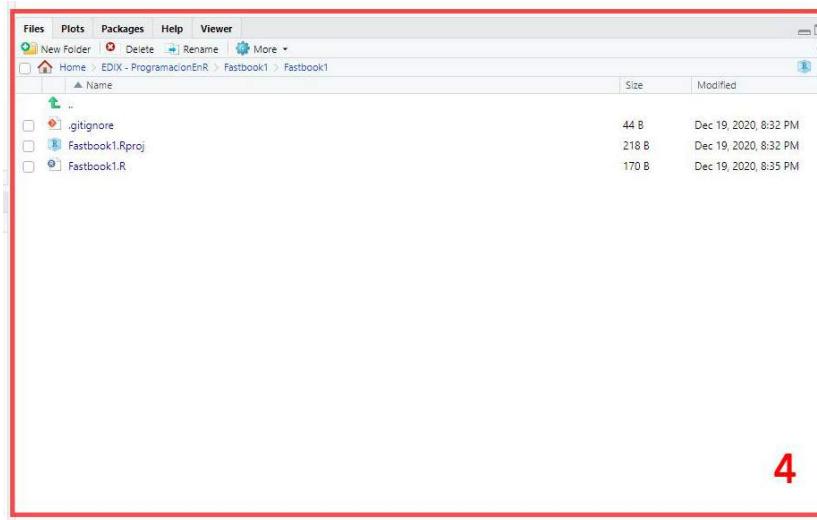
En la pestaña **Files** visualizamos los archivos y carpetas del directorio. Podemos abrirlos y navegar con total libertad.

La pestaña **Plots** muestra los gráficos cuando estamos trabajando con visualización.

En **Packages** tenemos el listado de todos los paquetes que se encuentran instalados en nuestro equipo, y se indica cuáles han sido importados en la sesión.

El último elemento de interés es la pestaña **Help**. Allí podemos encontrar la documentación de todas las funciones y paquetes de R, donde se explica en detalle su modo de empleo. Es una

herramienta realmente valiosa y útil.



---

A continuación, pasamos a visualizar un vídeo en el que vamos a recorrer y afianzar todos estos conceptos desde su perspectiva más práctica.



[Programación en R. fb. 01: Entorno RStudio](#)

# Tipos de variables

**X** Edix Educación

---

Las variables almacenan los datos y la información con la que estamos trabajando.

Existen **10 tipos de datos** que nos interesa conocer:

## Integer —

Números enteros (0, 3, -5, 2, -10).

## Numeric —

Números con decimales (2,5; -6,723; 3,03).

## Logical —

Booleanos, es decir, tienen dos posibles valores: *true* o *false*.

**Character** —

Información textual.

**Factor** —

Información categórica, es decir, cuando existe un rango de posibles valores. (Por ejemplo, el resultado de un examen: aprobado o suspendido).

**Date** —

Fecha (día, mes y año).

**POSIX** —

Fecha, hora y uso horario.

**NA** —

Valor perdido o no disponible.

**NULL** —

Elemento vacío.

**Inf** —

Número infinito.

**Una variable básica almacena un solo dato.** Por ejemplo, si creamos una variable que contiene el valor 2, será una variable básica de tipo integer.

Normalmente, nos interesa que nuestras variables puedan almacenar más de un dato. Aquí es donde aparecen las **variables estructurales**:

- **Vector:** es una colección unidimensional de datos del mismo tipo.
- **Matriz:** es una colección bidimensional de datos del mismo tipo.
- **Lista:** es una colección unidimensional que admite datos de distinto tipo.
- **DataFrame:** es una colección bidimensional que admite datos de distinto tipo.

---

**En el siguiente vídeo vamos a trabajar en RStudio para entender a fondo todos estos conceptos.**



[Programación en R, fb. 01: tipos de variables](#)

# Operadores

X Edix Educación

---

Haciendo uso de las variables, los operadores nos permiten llevar a cabo ciertas **tareas básicas**.

Diferenciaremos entre 4 tipos:

1

## Operadores de asignación

Operador	Definición	Ejemplo
<code>&lt;-</code>	Asigna un valor a una variable	<code>var &lt;- 10</code>
<code>=</code>	Asigna un valor a una variable, pero es recomendable usarlo únicamente en los argumentos de las funciones	<code>mean(x = c(2,7,6,4))</code>

Esta forma de trabajo es parte del estándar de R desde su creación. La flecha es usada cuando queremos introducir un contenido dentro de nuestra variable. El signo de igual se utiliza cuando queremos indicar a la función el valor de sus argumentos de entrada.

## 2

### Operadores matemáticos

Operador	Definición	Ejemplo	Resultado
+	Suma	$4 + 4$	8
-	Resta	$14 - 3$	11
*	Producto	$3 * 3$	9
/	División	$12 / 4$	3
$\wedge$	Potencia	$3 \wedge 3$	27
$\% \%$	Devuelve el resto de dividir un número por otro	$10 \% \% 4$	2

## 3

### Operadores comparativos

Los operadores comparativos nos sirven para comprobar si las variables cumplen o no ciertas condiciones que definamos.

Operador	Definición	Ejemplo	Resultado
<	Es menor que	$3 < 4$	TRUE
$\leq$	Es menor o igual que	$5 \leq 5$	TRUE

Operador	Definición	Ejemplo	Resultado
>	Es mayor que	10 > 12	FALSE
>=	Es mayor o igual que	4.5 >= 4.6	FALSE
==	Es igual que	7 == 9	FALSE
!=	No es igual que	7 != 9	TRUE

## 4

### Operadores lógicos

Los operadores lógicos son aquellos que trabajan con datos booleanos (TRUE o FALSE).

Operador	Definición	Ejemplo	Resultado
&	Sentencia AND. Para que devuelva TRUE, ambos elementos deben ser TRUE.	TRUE & FALSE	FALSE
	Sentencia OR. Para que devuelva TRUE, al menos un elemento debe ser TRUE.	TRUE   FALSE	TRUE
!	Invierte el dato booleano.	!TRUE	FALSE

---

Vamos a pasar a RStudio para poner en práctica lo aprendido en esta sección.



[Programación en R, fb. 01: operadores en R](#)

# Conclusiones

X Edix Educación

---

Este primer fastbook nos ha servido como aterrizaje en la asignatura.

Tras una breve introducción, hemos instalado R y RStudio en nuestros equipos para que puedas abordar los ejercicios prácticos. Posteriormente, se han presentado los elementos básicos, su entorno, variables y operadores. **Nuestro camino con R solo acaba de empezar.**

En el fastbook 2 aparecen conceptos más complejos que se centran en el tratamiento y la transformación de todos los tipos de variables que ya hemos presentado. No os preocupéis si, tras su visionado, vuestras ideas no están del todo claras; el lab y su enfoque totalmente práctico os ayudarán.

---

**En el campus os compartimos los scripts de R que se han construido durante los vídeos. Os recomiendo que los descarguéis para que pasen a formar parte de vuestro material de estudio y los podáis ejecutar.**

¡Enhорabuena! Fastbook superado

edix

Creamos Digital Workers