



UNIVERSITAT POLITÈCNICA  
DE CATALUNYA  
BARCELONATECH



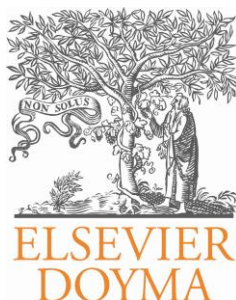
## Capítulo 2: Introducción a R: Primeros pasos

Jordi Cortés, José Antonio González  
Erik Cobo, Marta Vilaró, Rosario Peláez y Nerea Bielsa

Septiembre 2014

Departament d'Estadística  
i Investigació Operativa  
UNIVERSITAT POLITÈCNICA DE CATALUNYA

equator  
network



MEDICINA  
CLINICA

TRIALS  
TRIALS

# Introducción a R

	<b>Presentación</b> .....	3
<b>1.</b>	<b>Visión general</b> .....	<b>4</b>
	1.1. Instalación.....	5
	1.1.1. Instalación de R.....	5
	1.1.2. Instalación de RStudio .....	5
	1.2. Interfaz de RStudio .....	6
	1.3. Primeros pasos .....	8
	1.3.1. Instrucciones .....	8
	1.3.2. Objetos .....	9
	1.3.3. Funciones .....	10
	1.3.4. Instalar paquetes.....	11
	1.3.5. Ayuda.....	12
	1.3.6. Cierre de la sesión.....	13
<b>2.</b>	<b>Organizarla información</b> .....	<b>14</b>
<b>3.</b>	<b>Acceder y modificar datos</b> .....	<b>16</b>
<b>4.</b>	<b>Importar datos</b> .....	<b>18</b>
	4.1. Lectura.....	19
	4.2. Vista.....	20
	4.3. Descriptiva global y tipos de variables.....	20
	4.4. Datos ausentes: “missings”.....	22
	4.5. Validación.....	23
<b>5.</b>	<b>R-Comander</b> .....	<b>24</b>
<b>6.</b>	<b>Referencias</b> .....	<b>24</b>
	<b>Soluciones a los ejercicios</b> .....	25

### Presentación

Este capítulo le ayudará en sus primeros pasos con el nuevo líder de los paquetes estadísticos. El código de R es público: todo el mundo puede usarlo, revisarlo, criticarlo y mejorarlo. Así, con R, los resultados de su investigación son más transparentes. Nuestra Universidad apuesta, tan fuerte como puede, por programas libres.

Por supuesto, habituarse a un nuevo programa requiere paciencia. No se desespere, piense que profundizará en R a lo largo del curso.

Algunos consejos son: (1) siga las instrucciones, instale los programas y haga los ejercicios; (2) revise los vídeos “iniciáticos” de la página web del curso; (3) consulte a su tutor o cuelgue sus dudas en la web; (4) como con otros programas, intercambie experiencias con sus colegas; y (5) vaya aplicando las instrucciones a sus propios datos.

Nosotros podemos ayudarle a “entender”. Ayúdese Vd. a “retener”.

**Contribuciones:** (1) versión original de marzo 2013, JC, autor, y JAG, editor, con la colaboración de MV y RP; y (2) revisado en agosto de 2014 por NB y EC.

# 1. Visión general

R es un software libre para el análisis estadístico de datos.

**Lecturas:** Dicen Rius y Gonzalez en Medicina Clínica: “Que el software sea libre o privativo es una cuestión legal. Decimos que un determinado programa es libre si quien lo usa ostenta 4 derechos específicos sobre éste. A saber, el derecho a usar el programa con cualquier finalidad, el derecho a estudiar el programa, el derecho a compartir el programa y, finalmente, el derecho a mejorar el programa y distribuir la nueva versión.. Un software sobre el cual los usuarios no ostentan alguno de los derechos anteriores es un software privativo”.

**Nota:** Se basó en el programa comercial S+.

Nace en la segunda mitad de los años noventa y ha ganado popularidad ya que: 1) su adquisición es gratuita; 2) se pueden llevar a cabo los mismos análisis estadísticos que con S+; 3) estadísticos de todo el mundo contribuyen con paquetes que permiten realizar análisis cada vez más específicos y sofisticados; y 4) posee una versatilidad gráfica única destacando su variedad y facilidad de adaptación.

Funciona por comandos, lo que requiere introducir instrucciones que Vd. debe conocer previamente. Ello garantiza que cada uno hace lo que sabe y sabe lo que hace. Así, al inicio es algo farragoso, pero a larga garantiza mayor fiabilidad de los resultados.

La ayuda de R, con un “?” interrogante, explica detalladamente cada instrucción.

Existen interfaces que facilitan trabajar con R: RStudio abre y edita más códigos y más opciones que el R convencional. Por ejemplo, permite comprobar rápidamente si existe algún paréntesis sin cerrar; o ver el contenido de unos datos con un solo clic de ratón.

## 1.1. Instalación

### 1.1.1. Instalación de R

Ejecute ahora los siguientes pasos para instalar R.

Instalación de R
1. Abra la página web de R: <a href="http://www.r-project.org/">http://www.r-project.org/</a> .
2. Haga clic en ‘CRAN’ y, a continuación escoja uno de los servidores ( <i>mirrors</i> ) de CRAN (Comprehensive R Archive Network).
3. Según el sistema operativo, haga clic en Linux, MacOS X o Windows y siga las instrucciones correspondientes.
4. Si usa Windows, haga clic en ‘base’ y a continuación en ‘Download R 3.x.y for Windows’, en donde <i>x</i> e <i>y</i> indican la versión actual de R.
5. Guarde el fichero.
6. Ejecute el fichero desde la carpeta en la cual fue guardado y siga las instrucciones de instalación.

**NOTA:** De esta manera instala la versión básica de R con los paquetes básicos. Cuando sea preciso, explicaremos cómo instalar algún otro de los más de mil paquetes contribuidos.

### 1.1.2. Instalación de RStudio

Ejecute ahora los siguientes pasos para instalar RStudio.

Instalación de RStudio
1. Abra la página web de RStudio: <a href="http://www.rstudio.com/">http://www.rstudio.com/</a>
2. Clique en ‘Download now’ → ‘Download RStudio Desktop’
3. Clique en la versión recomendada: ‘ <i>RecommendedForYourSystem</i> ’
4. Guarde el fichero
5. Ejecute el fichero desde la carpeta en la cual fue guardado y siga las instrucciones de instalación

## 1.2. Interfaz de RStudio

La Figura 1.1 muestra la estructura básica, por defecto, de RStudio

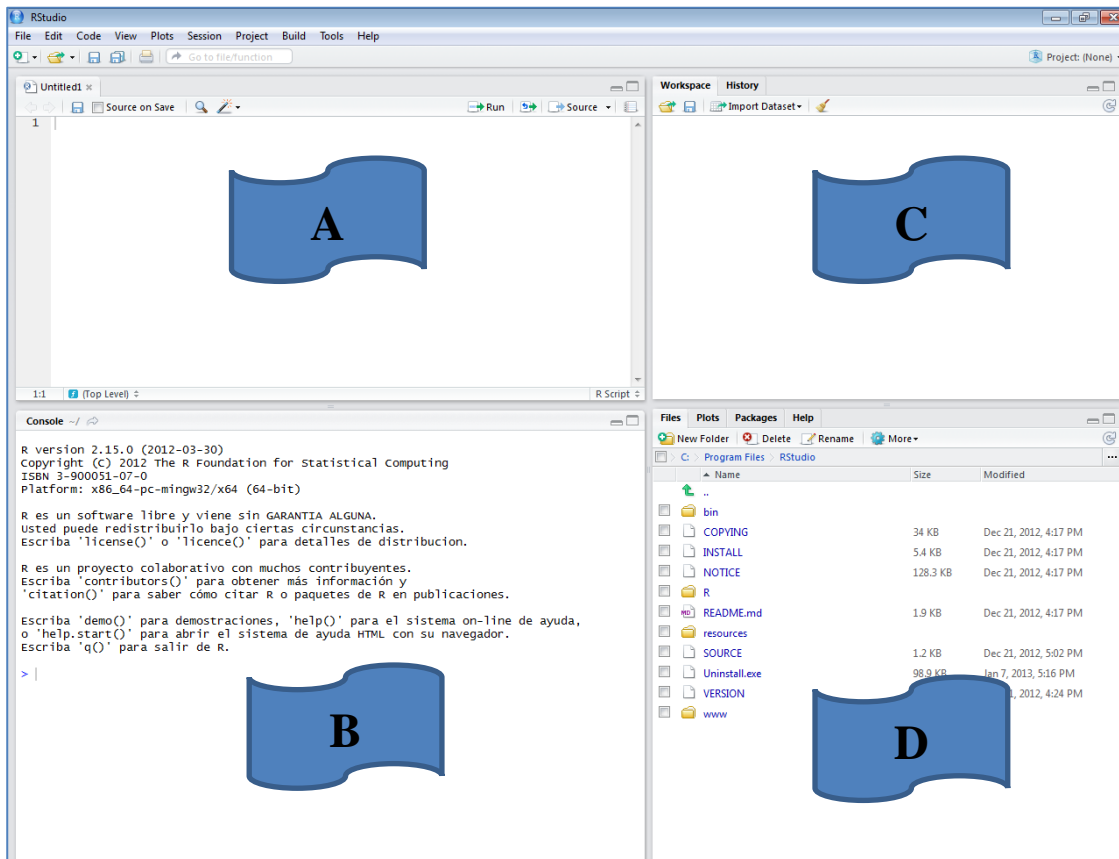



Figura 1.1. Interfaz por defecto del editor RStudio.

La interfaz consta de las siguientes 4 ventanas:

- A) **Script.** Para escribir el código de análisis. No está presente la primera vez: cree uno nuevo con *File* → *New* → *R Script*.
- B) **Consola.** Donde se envían las instrucciones del script para ser ejecutadas y donde aparecen los resultados.
- C) Se subdivide en 2 subventanas:
  - a. **Workspace** (Área de trabajo). Lista de todos los objetos (variables, datos, funciones...) de la sesión actual.
  - b. **History** (Historia). Lista de todas las instrucciones ejecutadas desde el inicio de la sesión.
- D) Se subdivide en 4 subventanas:
  - a. **Files** (Archivos). Lista de los ficheros en el directorio de trabajo (similar al explorador de windows).

- b. **Plots** (Gráficos). Contiene todos los gráficos realizados durante la sesión. Con las flechas  puede ir adelante y atrás en la búsqueda de gráficos.
- c. **Packages** (Paquetes). Ventana destinada a la instalación de paquetes.
- d. **Help** (Ayuda). Ventana donde aparece la ayuda de R cuando se solicita.

**NOTA:** Esta interfaz puede modificarse de forma sencilla a través de Tools → Options → PaneLayout. Permite redistribuir o eliminar las ventanas según sus preferencias.

Al ver esta interfaz diferente de la de otros paquetes estadísticos pueden surgir algunas dudas:

### ¿Porqué conviene utilizar un script?

Si trabaja por menús, los análisis realizados no siempre se almacenan. Al guardar todo el proceso de análisis en un fichero de texto con extensión *.R* podrá reproducir y documentar en todo momento el análisis realizado.

Más importante es conseguir una investigación transparente y reproducible: programarlo con antelación y documentarlo en un Plan de Análisis Estadístico garantiza que los resultados no guían el análisis.

### ¿Qué aporta el workspace (área de trabajo)?

En ocasiones, conviene disponer de variables o funciones creadas por uno mismo; o de de datos complementarios. En el workspace aparecen enumerados todos estos ítems y se puede acceder a su contenido clicando sobre ellos.

### ¿Qué aporta el history (historial)?

En la subventana de historial se guardan todos los comandos ejecutados en la sesión. Es útil para ver cómo se ha llegado hasta cierto punto. En la consola también se dispone del historial de instrucciones, pero mezclado con los resultados obtenidos.

### ¿Es la ventana gráfica pequeña?

Si se requiere ver el gráfico en un tamaño mayor del que ofrece la subventana de gráficos, la opción *windows* ( ) permite abrir una nueva ventana del tamaño deseado.

## 1.3. Primeros pasos

### 1.3.1. Instrucciones

Puede ejecutar las instrucciones directamente de la consola o a través de un *script* o programa.

**Consola.** El indicador o *prompt* del sistema es el signo `>`. A continuación del mismo, se escriben las instrucciones seguidas de un *Enter*. En ese momento, el programa examinará la sentencia y: (1) si es correcta, la ejecutará; (2) si no lo es, mostrará un mensaje de error; y (3) si es incompleta, mostrará el signo `+`, indicando que espera que complete la orden en la línea siguiente.




#### Ejercicio 1.1

Ejecute los siguientes comandos en la consola y describa que ocurre:

```
> 2+3
> 2 + "a"
> 2 +
```

**NOTA:** Es posible navegar entre los comandos ejecutados previamente mediante las teclas `↑` y `↓`. La tecla `Esc` permite reiniciar la actual línea en edición y la combinación `'Ctrl+C'` interrumpe la edición o ejecución en curso.

**Script.** Es más práctico y eficiente crear un código que contenga las instrucciones. Se abre un script nuevo desde la barra de herramientas mediante: `File → New → R script`. Diferentes comandos irán en distintas líneas o bien separados por `;`. Para ejecutarlos, se tienen que seleccionar y clicar en el botón  (o bien con la combinación de teclas `'Ctrl+R'` o `'Ctrl+Enter'`). Los resultados aparecerán en la consola.



#### Ejercicio 1.2

Cree el siguiente script y ejecute línea por línea con `'Ctrl+Enter'`

```
pi
5*3 ; 6/2
3 + 4 # debería dar 7
```

**NOTA:** El signo `#` indica la introducción de un comentario que puede ser útil para clarificar algún comando.



### 1.3.2. Objetos

¿Qué son? En R, todo es un objeto: un conjunto de datos, una variable, un valor, una función. Sobre estos objetos se aplican operaciones.

**NOTA:** La “[programación orientada a objetos](#)” tiene muchas ventajas, por ejemplo, que se pueden ‘heredar’ operaciones entre diferentes objetos. Vea esta entrada en Wikipedia.

**Nombre.** El nombre de un objeto de R puede ser cualquier cadena alfanumérica formada por letras (R distingue entre mayúsculas y minúsculas), dígitos del 0 al 9 (nunca en primera posición del nombre) y los signos "." y "\_" (punto y guion bajo). Por ejemplo, Exp1289 o muestra.ini son nombres válidos.

**NOTA:** mejor no usar ciertas palabras que R se reserva, como los nombres de las instrucciones de su lenguaje de programación (*break*, *for*, *function*, *if*, *in*, *next*, *repeat*, *return*, *while*) o los de las funciones incorporadas.

**Asignaciones.** Puede dar valor a un objeto con los signos "=", "<-", y "->".



#### Ejercicio 1.3

Realice las siguientes asignaciones:

```
> n <- 5*2 + sqrt(144)
> m = 4^0.5
> n + m -> p
```

Escriba el nombre de un objeto para ver el contenido.



#### Ejemplo R

```
# Ejemplo para ver distintos objetos
> n
[1] 22
> m ; p
[1] 2
[1] 24
> log
function (x, base = exp(1)) .Primitive("log")
```

Observe que si el objeto es una función ('log'), R muestra lo que ejecuta.

**NOTA:** para algunos objetos, también puede clicar sobre su nombre en la ventana *Workspace*

**Lista de objetos.** El comando `ls` proporciona el listado de objetos presentes en la sesión de trabajo actual.



**Ejemplo R**

```
# Lista los objetos en memoria
> ls()
[1] "n" "m" "p"
```

**1.3.3. Funciones**

Las funciones son instrucciones que realizan operaciones sobre objetos.



**Ejemplo R**

```
# logaritmo natural de n=22; log(22)=3.091
> log(n)
[1] 3.091042
```

**NOTA:** ‘log’ se refiere al logaritmo natural, con base  $e=2.71$ , no al decimal, con base 10.

**Nota técnica:** los objetos que necesita una función para ejecutarse se denominan *parámetros* o *argumentos* de entrada. En el caso de la función `log` tiene un parámetro obligatorio (el número del cual se desea calcular el logaritmo) y uno opcional (la base en la que se calcula, que si no se especifica, se sobrentiende que es el logaritmo natural de base  $e$ )

**Sintaxis.** Se escribe el nombre de la función seguida de un paréntesis que contiene los parámetros (separados por comas) con la información necesaria para que se ejecute.



**Ejemplo R**

```
# Cálculo del valor máximo de n y p
> max(n,p)
[1] 24
```

**Creación.** Puede crear funciones propias con la instrucción *function*: introduzca entre paréntesis los parámetros de entrada (objetos necesarios para que se ejecute) y a continuación, entre llaves, los comandos a realizar.

**Ejemplo R**

```
# Cálculo del máximo y el mínimo de una variable

max.min <- function(x) {
  print(max(x))
  print(min(x))
}

edad <- c(20,21,20,22,23,20,25,26,20,21)

max.min(x=edad)
```

**NOTA:** Si desea que proporcione (“retorne”) un resultado, finalice con la instrucción `return()` y el resultado entre paréntesis.

**Ejercicio 1.4**


Construya una función llamada IMC que calcule el 'Índice de Masa Corporal' a partir del peso (en Kg) y la altura (en m).

$$IMC = \frac{\textit{peso}}{\textit{altura}^2}$$

La función debe tener dos parámetros de entrada.

**1.3.4. Instalar paquetes**

Un paquete es un conjunto de funciones sobre un tema concreto. Para análisis específicos o sofisticados, se deben instalar paquetes adicionales.

Para usar un paquete se hacen 2 pasos: 1) Instalarlo (desde CRAN) y 2) Cargarlo (ponerlo) en memoria. La pestaña *Packages* (Paquetes) de la ventana D contiene la lista y una breve descripción de todos los paquetes instalados. El símbolo ✓ indica que, además, está cargado. Si desea cargar un paquete ya instalado, marque con un ✓ el paquete en cuestión. Clicando en el icono  **Install Packages**, se instalan otros paquetes especificando el nombre del mismo. Por ejemplo, el paquete *survival* contiene funciones para el análisis de supervivencia.

**NOTA:** En el menú *Packages* de la página <http://cran.r-project.org/> están todos los paquetes disponibles.

**NOTA:** La primera vez que instale un paquete, R le preguntará el país desde dónde desea descargarlo (aunque el tiempo de descarga no difiere en exceso). Una vez instalado un paquete en un ordenador, no se necesitará instalarlo más, pero sí que se deberá cargar clicando ✓ en el paquete. Una alternativa para instalar y cargar los paquetes por comandos es con las instrucciones *install.packages* y *library*.



### Ejemplo R

```
# Instalación de un paquete  
> install.packages ('sudoku') # instalar paquete  
> library (sudoku)           # cargar paquete  
> windows ( )                # ventana del tablero  
> playSudoku ( )             # ya se puede jugar!
```

### 1.3.5. Ayuda

**Videos.** Encontrará muchos en la red. Los de nuestra página están pensados para Vd.

**Manuales.** R dispone de manuales a los cuales se accede vía la barra de herramientas: *Help* → *R Help*. En la subventana de ayuda aparecerá, entre otras cosas, una lista de manuales.

**Instrucciones.** El comando *help* y '?' dan información específica sobre funciones



### Ejemplo R

```
> help(log)  
> ?ls
```

**Paquetes.** El comando *library()* abre una ventana con información sobre los paquetes instalados en R. Para obtener más información sobre estos paquetes, use las funciones *library* y *help* conjuntamente.



### Ejemplo R

```
> library(help="foreign")
```

**NOTA:** Otra posibilidad para obtener esta información es accediendo a ella desde la barra de herramientas *Help* → *R Help* y después, en la página que se abre, hacer clic en '*Packages*' y en el paquete correspondiente.

**Temas.** La función *help.search* busca ayuda sobre un tema concreto entre todos los paquetes instalados.



### Ejemplo R

```
> help.search("logistic regression")  
> help.search("R help")
```

**Foros.** La función *RSiteSearch* busca las palabras de interés entre todos los mensajes enviados a las listas de ayuda de correo electrónico de R; por ejemplo, para hallar información sobre la prueba de *Hosmer Lemeshow*.



### Ejemplo R





```
> RSiteSearch("Hosmer Lemeshow test")
```





### Ejercicio 1.5

(1) Instale el paquete *survival*, (2) busque la ayuda sobre la instrucción *plot.survfit* y (3) ejecute las instrucciones que aparecen en el ejemplo (al final de la ayuda)

### 1.3.6. Cierre de la sesión.

**Guardar/Cargar histórico.** Puede guardar los comandos ejecutados hasta el momento clicando en el icono de disco () en la pestaña *History* de la ventana C. Es posible cargar el histórico de otra sesión mediante el icono de carpeta () en la misma pestaña. De esta manera pasará los comandos de la sesión anterior a un script () o a la consola directamente () .

**NOTA:** Otra posibilidad es guardar y cargar el historial con las instrucciones *savehistory* y *loadhistory*, respectivamente.

**Guardar/Cargar área de trabajo.** Si quiere volver a utilizar los objetos de R en uso, guarde el contenido de la sesión clicando en el icono del disco () en la pestaña *Workspace* de la ventana C. Y cárguelos con el icono de la carpeta () .

**NOTA:** Otra posibilidad es guardar y cargar el área de trabajo con las instrucciones *save.image* y *load.image*, respectivamente.

**NOTA:** Durante una sesión se pueden cargar diferentes áreas de trabajo.

**NOTA:** Si desea guardar solamente algunos de los elementos, por ejemplo los objetos *x* e *y*, tiene 2 opciones: o eliminar primero los demás objetos con la función *rm()* y después usar la función *save.image()*; o usar la función *save*:

```
> save(x, y, file="nombredearchivo.RData")
```

**Salir del programa.** Con la orden *q()* abandona R. Antes de cerrarse, R pregunta al usuario si quiere guardar el actual espacio de trabajo en el fichero *.RData* en la carpeta de trabajo actual – conjuntamente con el histórico de la sesión.

**Indicar el directorio.** Para hacer referencia a algún fichero de disco debe utilizar la dirección entre comillas con la barra / o las barras \\ entre subcarpetas.



### Ejemplo R

```
> save.image  
("C:/Archivos de programa/R/nombre.RData")
```

## 2. Organizarla información

Los datos son la materia prima de la Estadística. Este punto muestra estructuras para almacenar datos, vectores y *data.frames* en R.

**Vectores.** Se usan para almacenar el contenido de una variable. Es un conjunto de elementos del mismo tipo (numérico o carácter). Se crean con la instrucción *c ( )* poniendo en el interior del paréntesis todos sus elementos separados por comas.



### Ejemplo R

```
# Creación de la variable edad de 8 individuos  
> edad <- c(20,21,20,22,23,20,25,26)  
> edad  
[1] 20 21 20 22 23 20 25 26  
  
# Creación de la variable genero de 8 individuos  
> genero <- c("h","h","h","h","h","m","m","m")  
> genero  
[1] "h" "h" "h" "h" "h" "m" "m" "m"
```

**NOTA:** El [1] que aparece al principio de la salida indica la posición (orden) que ocupa el primer elemento de la fila. Es útil cuando la variable es muy larga. Para crear variables de caracteres, se deben poner los valores entre comillas simples (') o dobles (").

**Data.frames.** Son los conjuntos de datos habituales que constan de varias variables, sean numéricas o categóricas. Normalmente, las filas representan los individuos y las columnas, las variables. Es el tipo por defecto cuando se lee un fichero de datos. También es posible crearlo con la instrucción *data.frame*.



### Ejemplo R

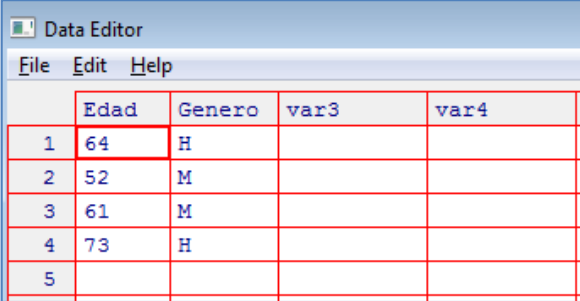
```
# Creación y vista de un data.frame
> Edad <- c(64,52,61,73)
> Genero <- c('H','M','M','H')
> df <- data.frame(Edad,Genero)
> df
  Edad Genero
1   64      H
2   52      M
3   61      M
4   73      H
```

La instrucción *edit* permite introducir directamente los datos en las celdas de un *data.frame*. Puede asignar el resultado al mismo objeto o a uno nuevo.



### Ejemplo R

```
# Editar los datos
> df <- edit(df)
```



	Edad	Genero	var3	var4
1	64	H		
2	52	M		
3	61	M		
4	73	H		
5				

**NOTA:** Existen otras formas de almacenar los datos: (1) Matrices (*matrix*), similares al *data.frame*, pero sólo con variables numéricas; (2) Arrays (*array*), útiles para datos con más de dos dimensiones (por ejemplo, repeticiones de variables); y (3) Listas (*list*), que contienen una combinación de cualquiera de las anteriores y otras no mencionadas.



### Ejercicio 2.1

Cree un *data.frame* (con valores inventados) de 4 individuos con las variables: nombre, peso, altura y IMC.

### 3. Acceder y modificar datos

Acceda a una observación concreta de un vector (variable) poniendo la posición entre corchetes, y modifíquela asignando un valor a la posición.



#### Ejemplo R

```
# Consultas
> Edad
[1] 64 52 61 150

> Edad[3]
[1] 61

# Modificación
> Edad[4] <- 50

> Edad
[1] 64 52 61 50
```

Para ver un *data.frame* completo, existen varias alternativas:

1. Clicar en el nombre del *data.frame* en la pestaña del *workspace* de la ventana C.
2. Con la instrucción *edit*, poner entre parentésis el nombre del *data.frame*: *edit(df)*
3. Escribir el nombre del *data.frame* en la consola: *df*

Para acceder a una **observación** concreta del *data.frame*:

1. Por la posición: [*número de fila*, *número de columna*]
2. Por los nombres: ["*nombre de fila*", "*nombre de columna*"] (por defecto, los nombres de las filas son su número de orden y los de las columnas los nombres de las variables iniciales. Se modifican con los comandos *rownames* y *colnames*)



#### Ejemplo R

```
# Valor concreto por la posición
> df[4,1]
[1] 73

# Valor concreto por los nombres
> df["4","Edad"]
[1] 73
```

Para ver únicamente una **variable** del *data.frame*:

1. Por la posición: [ , *número de columna*]
2. Por el nombre: [ , "*nombre de columna*"]
3. Por el nombre separado por un \$.



**Ejemplo R**

```
# Variable por la posición
> df[,2]
[1] H M M H
Levels: H M
# Variable por el nombre de la columna
> df[, "Genero"]
[1] H M M H
Levels: H M
> df$Genero
[1] H M M H
Levels: H M
```

Los niveles (*Levels*) que aparecen al final de la salida en una variable alfanumérica hacen referencia a los diferentes niveles (categorías) de la variable.

Para ver únicamente un **individuo** del *data.frame*:

1. Por la posición: [número de fila, ]
2. Por el nombre: ["nombre de fila" , ]

**Ejemplo R**

```
# Individuo por la posición
> df[2,]
  Edad Genero
2   52      M
# Individuo por el nombre de la fila
> df["2",]
  Edad Genero
2   52      M
```

La instrucción *head* permite ver únicamente un número determinado de observaciones, útil en *data.frames* extensos.

**Ejemplo R**

```
# Ver los tres primeros individuos
> head(df,3)
  Edad Genero
1   64      H
2   52      M
3   61      M
```

Para modificar un *data.frame*, al igual que pasaba con los vectores, asigne un valor o varios a la posición o posiciones que quiera modificar. Tenga en cuenta que las variables categóricas

(factores), sólo permiten valores ya existentes. Si se desea modificar todos los valores de una variable completa, use la instrucción *levels*.



### Ejemplo R

```
# Cambiar los niveles de una categórica
> levels(df$Genero)
[1] "H" "M"
> levels(df$Genero)[1] <- "Hombre"
> levels(df$Genero)[2] <- "Mujer"
> df
  Edad Genero
1   64 Hombre
2   52  Mujer
3   61  Mujer
4   73 Hombre
```



### Ejercicio 3.1

En el *data.frame* del Ejercicio 2.1, cambie el nombre del tercer individuo y elimine las cifras decimales del IMC con la función *round*

## 4. Importar datos

R permite importar datos desde casi cualquier formato. La siguiente tabla enumera las instrucciones para los formatos más habituales.

Tipo de fichero	Extensión	Instrucción en R	Paquete
Texto	<i>.txt</i> o <i>.dat</i>	<i>read.table</i>	<i>utils</i>
Texto separado por comas	<i>.csv</i>	<i>read.csv</i> <i>read.csv2</i>	<i>utils</i>
SPSS	<i>.sav</i>	<i>read.spss</i>	<i>foreign</i>
		<i>spss.get</i>	<i>Hmisc</i>
SAS	<i>.sas7bdat</i>	<i>read.sas7bdat</i>	<i>sas7bda</i>
		<i>sas.get</i>	<i>Hmisc</i>
STATA	<i>.dta</i>	<i>stata.get</i>	<i>Hmisc</i>
MINITAB	<i>.mtp</i>	<i>read.mtp</i>	<i>foreign</i>

**Nota:** Las instrucciones *read.table*, *read.csv* y *read.csv2* vienen con el paquete *utils*, ya instalado por defecto. Con *read.table* o *read.delim* puede “pegar” desde el portapapeles después de hacer un “copiar” en un conjunto de datos.

**Nota:** Las instrucciones *read.csv* y *read.csv2* se utilizan en ficheros de texto, según las columnas estén separadas por ‘,’ (comas) y ‘;’ (puntos y comas) —respectivamente.

**Nota:** En general, la lectura de un fichero Excel es posible pero complicada y el mismo R la desaconseja. En este caso, es mejor guardar la hoja de cálculo con un formato csv: Archivo → Guardar como → csv. Y leerlo en R con *read.csv2*.

A continuación verá un posible proceso de lectura y validación de un mismo conjunto de datos con tres formatos distintos.

## 4.1. Lectura

El primer parámetro es el nombre del fichero a importar. Su directorio (carpeta) se especifica junto al nombre, o se fija con la instrucción *setwd*.

Para poder ver el ejemplo, descargue primero los datos GPT de la página web del curso (<http://bioestadistica.upc.edu/node/30>). Guarde los tres ficheros (*txt*, *csv* y *sav*) en la carpeta 'C:/Documents'.



### Ejemplo R

```
# Instalar y cargar el paquete foreign para leer datos SPSS
> install.packages('foreign')
> library(foreign)
# Fijar el directorio donde estan los ficheros
> setwd('C:/Documents') # Debe cambiarse!
# Lectura en las tres extensiones
> datos1 <- read.table('GPT.txt',header=TRUE)
> datos2 <- read.csv2('GPT.csv',header=TRUE)
> datos3 <- read.spss('GPT.sav',to.data.frame = TRUE)
```

**NOTA:** El *header=TRUE* indica que la 1ª fila del archivo de origen contiene los nombres de las variables. El *to.data.frame=TRUE* indica que lo importe como *data.frame* (ya que, por defecto, lo importa como lista).

Para leer los ficheros de texto (no otros formatos) directamente desde una página web, únicamente se debe especificar la dirección dentro de la función *url*.



### Ejemplo R

```
# Lectura de los ficheros de texto desde una url

> datos1 <- read.table(url('http://www-
eio.upc.es/teaching/best/GPT.txt'),header=TRUE)

> datos2 <- read.csv2(url('http://www-
eio.upc.es/teaching/best/GPT.csv'),header=TRUE)
```

## 4.2. Vista

Una vez obtenidos los datos, se debe verificar que se han leído correctamente. Si el *data.frame* es largo, la instrucción *head* enseña únicamente las primeras filas (6, por defecto).



### Ejemplo R

```
# Vista de los datos anteriores (debe ser idéntica)

> head(datos1)
> head(datos2)
> head(datos3)

  id  sex age   gpt  hiv colester
1  58 Male  36  High HIV+    170
2 172 Male  33  High HIV-    116
3 190 Male  30  High HIV+    139
4 239 Male  33 Normal HIV+    166
5 312 Male  40  High HIV-    155
6 313 Male  32  High HIV-    221
```

## 4.3. Descriptiva global y tipos de variables

El capítulo 3 estudia la descriptiva numérica y gráfica. Avancemos ahora unas sentencias de R. La explicación de qué significan esos resultados se verá en el próximo capítulo.

**Nota:** Una vez comprobado que los 3 conjuntos de datos están correctamente leídos y son idénticos, trabajará únicamente con uno de ellos.

La instrucción *summary* aplicada a un *data.frame* proporciona una descriptiva global.



### Ejemplo R

```
# Descriptiva global del conjunto de datos

> summary(datos1)

      id          sex          age
```

```

Min.      : 58      Female: 73      Min.      :19.00
1st Qu.:1644      Male   :325      1st Qu.:27.00
Median   :1826                                Median   :30.00
Mean     :1793                                Mean     :30.83
3rd Qu.:1975                                3rd Qu.:34.00
Max.     :2218                                Max.     :67.00

      gpt          hiv          colester
High   :204      HIV-:175      Min.    : 0.0
Normal:184      HIV+:223      1st Qu.:132.0
NA's   : 10                                Median  :151.0
                                           Mean    :151.3
                                           3rd Qu.:170.0
                                           Max.    :283.0
    
```

Nótese que la descriptiva es distinta para las variables numéricas (sean enteras o continuas) que para las categóricas. Para las primeras (*id*, *age* y *colester*), proporciona mínimo, máximo, media y cuartiles; y para las segundas (*sex*, *gpt* y *hiv*), las frecuencias de cada categoría.

**Nota:** R interpreta que una variable con caracteres alfanuméricos es categórica, como debe ser. Pero las que contienen sólo números pueden ser también categóricas. Por ejemplo, R interpreta que la variable *id* (identificador del paciente) es numérica, por lo que calcula la media, lo que no tiene sentido. Ver las frecuencias permite comprobar que no existan dos casos con el mismo identificador, pero no sirve para nada más. El parámetro *colClasses* en la instrucción *read.table* comunica a R el tipo de variable: “*numeric*” (continuas), “*integer*” (enteras), “*factor*” (categóricas), “*character*” (cadena de caracteres), “*boolean*” (lógica)... Definirlo correctamente permite a R calcular las funciones aplicables a ese tipo de variables.

La instrucción *sapply*, sobre cierto *data.frame*, y con la función *class*, informa sobre el tipo de la variable.



```

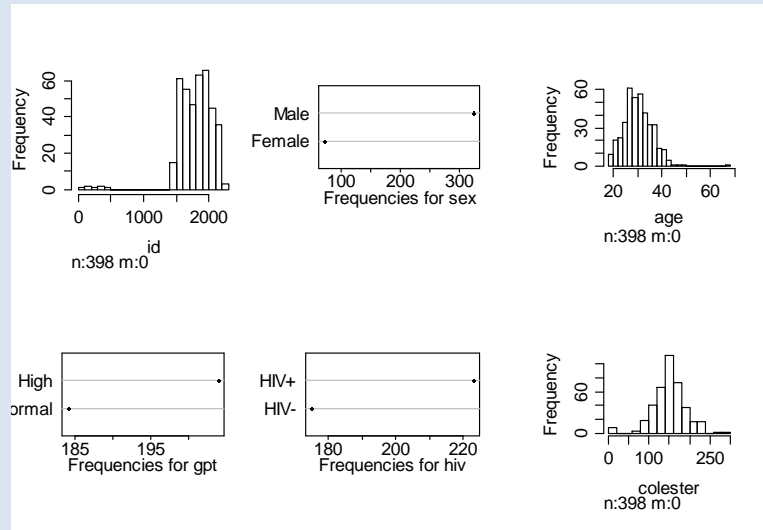
Ejemplo R
# Tipo de cada variable
> sapply(datos1,class)
      id      sex      age
"integer" "factor" "integer"
      gpt      hiv  colester
"factor"  "factor" "integer"
    
```

La instrucción *hist.data.frame* ( ) del paquete *Hmisc* realiza el histograma para variables numéricas e indica las frecuencias de las categóricas.



### Ejemplo R

```
# Histograma de numéricas y frecuencias de factores
> install.packages("Hmisc")
> library(Hmisc)
> hist.data.frame(datos1)
```



El siguiente capítulo profundiza en la estadística descriptiva.

## 4.4. Datos ausentes: “missings”

R codifica los valores ausentes con *NA* (*Not Available*). Obsérvese que antes ha indicado 10 *NA*'s (missings) en la variable *gpt*. Puede gravar los datos ausentes ya como *NA*'s; o especifique en el *read.table* dentro del parámetro *na.strings*, el código usado. Por ejemplo, *read.table(fichero.txt, na.strings="9999")*.

**Nota:** Los espacios en blanco también son interpretados como datos ausentes dentro de variables numéricas. En las variables categóricas se consideran como una categoría más.

La instrucción *is.na()* retorna TRUE o FALSE dependiendo de si ese valor de la variable tiene un missing. La instrucción *which()* identifica cuál es la posición en el vector de los casos que cumplen una condición lógica —como ser faltante.



### Ejemplo R

```
# Datos ausentes en gpt
> is.na(datos1$gpt)
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[8] FALSE FALSE FALSE FALSE FALSE TRUE FALSE
```

```
[15] FALSE FALSE FALSE FALSE FALSE FALSE FALSE ...
> which(is.na(datos1$gpt))
[1] 13 71 76 104 144 231 295 300 332 360
```

También puede ver los datos en otras variables de los individuos con datos ausentes en una concreta.



### Ejemplo R

```
# Guarda en gpt.na los individuos con missings
> gpt.na <- which(is.na(datos1$gpt))

# Imprime los datos de aquellos casos con missing en gpt
> datos1[gpt.na,]
      id  sex age  gpt  hiv colester
13  1458 Male  29 <NA> HIV+         0
71  1580 Male  31 <NA> HIV+         0
76  1587 Male  29 <NA> HIV+         0
104 1650 Male  38 <NA> HIV+         0
144 1730 Male  30 <NA> HIV+         0
231 1876 Male  29 <NA> HIV-        205
295 1969 Male  37 <NA> HIV+         0
300 1976 Male  35 <NA> HIV+         0
332 2040 Female 37 <NA> HIV+        116
360 2101 Male  32 <NA> HIV+        159
```

## 4.5. Validación

La instrucción *which()* permite detectar datos incongruentes. Suponga que uno de los criterios de selección fuese tener una edad comprendida entre 18 y 65 años.



### Ejemplo R

```
# Guarda en age.val los casos con edades
fuera de rango

# El simbolo "|" indica ó

> age.val <- which(datos1$age<18 | datos1$age>65)
# Imprime los datos de los casos con edades fuera del rango
> datos1[age.val,]
      id  sex age  gpt  hiv colester
281 1950 Male  67 High HIV-        128
```

## 5. R-Comander

El paquete *Rcmdr* o R Commander ofrece un sistema de ventanas y menús que hace R más amigable. No obstante, es menos flexible, ya que limita el uso de opciones de muchas funciones.

Para activar R Commander, instale el paquete *Rcmdr* con `install.packages('Rcmdr')` y después cárguelo mediante `library(Rcmdr)`.

Haga clic en las ventanas que no sabe qué es lo que hacen para aprender, no para analizar los datos.

**LECTURA:** el libro de Arriaza y Fernández de la Universidad de Cádiz: <http://knuth.uca.es/repos/ebrcmdr/pdf/actual/ebrcmdr.pdf> es una buena referencia para aprender a usar R Commander

## 6. Referencias

Existen múltiples guías para el uso de R. Aquí enumeramos algunas de las más útiles:

Paradis, Emmanuel. R para principiantes. Disponible en [http://cran.r-project.org/doc/contrib/rdebuts\\_es.pdf](http://cran.r-project.org/doc/contrib/rdebuts_es.pdf)

Díaz-Uriarte, Ramón. Introducción al uso y programación del sistema estadístico R Disponible en <http://cran.r-project.org/doc/contrib/curso-R.Diaz-Uriarte.pdf>

Venables and Smith. An Introduction to R. Disponible en <http://cran.r-project.org/doc/manuals/R-intro.pdf>

Correa y González. Gráficos Estadísticos con R. Disponible en <http://cran.r-project.org/doc/contrib/grafi3.pdf>



**Soluciones a los ejercicios**

- 1.1** `> 2 + 3` → R devuelve el resultado de la suma. Es una instrucción correcta y completa.  
`> 2 + "a"` → R proporciona un mensaje de error al ser incapaz de realizar la suma de un número y un carácter. Es una instrucción incorrecta.  
`> 2 +` → R queda a la espera de que se finalice la instrucción. Se puede escribir un `3` y, a continuación *Enter* para acabar la sentencia o bien apretar *Esc* para reiniciar la instrucción. Es una instrucción incompleta.

- 1.2** El resultado de la ejecución es el siguiente:

```
> pi
[1] 3.141593
```

El valor de la constante  $\pi$ , R lo tiene guardado en memoria dentro del objeto *pi*.

```
> 5*3;6/2
[1] 15
[1] 3
```

Se ejecutan las dos operaciones separadas por el `';`

```
> 3 + 4 # debería dar 7
[1] 7
```

Se realiza la suma de `3 + 4` y se ignora el comentario posterior.

- 1.3** El resultado de las operaciones es el siguiente:

```
> n <- 5*2 + sqrt(144)
> n
[1] 22
```

*n* es `5x2` más la raíz (*sqrt*) de 144

```
> m = 4^0.5
> m
[1] 2
```

*m* es 4 elevado a 0.5 (equivalente a hacer la raíz cuadrada de 4)

```
> n + m -> p
> p
[1] 24
```

*p* es la suma de *n* y *m*

- 1.4** Una posible función sería:

```
IMC <- function(peso,altura) {
  imc <- peso/altura^2
  return(imc)
}
```

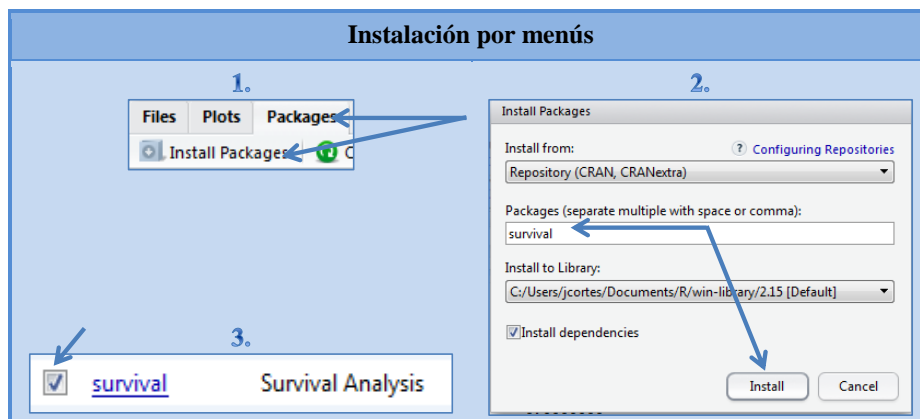
Se realiza una prueba:

```
> IMC(peso=75,altura=1.75)
[1] 24.4898
```

O también, como R introduce en orden los valores:

```
> IMC(75,1.75)
[1] 24.4898
```

**1.5** La instalación por menús se puede realizar con los siguientes pasos:



A continuación se muestra la instalación por comandos y el resto del ejercicio

```
> # Instalación del paquete
> install.packages ('survival')
> library (survival)

> # Pedir ayuda
> ?plot.survfit

> # Ejecutar instrucciones del ejemplo
> leukemia.surv <- survfit(Surv(time, status) ~ x, aml)
> plot(leukemia.surv, lty = 2:3)
> legend(100, .9, c("Maintenance", "No Maintenance"),lty = 2:3)
> title("Kaplan-Meier Curves\nfor AML Maintenance Study")
```

```
> lsurv2 <- survfit(Surv(time, status) ~ x, aml, type='fleming')
> plot(lsurv2, lty=2:3, fun="cumhaz", xlab="Months", ylab="Cumulative
Hazard")
```

Obsérvese que obtiene 2 gráficos. Uno para las curvas de supervivencia y otro para las curvas de riesgo de unos datos almacenados en la memoria de R.

**2.1** Primero cree las variables y luego únalas con la instrucción *data.frame*. Use la función creada en el Ejercicio 1.4.

```
> Nombre <- c("Juan","Pedro","María","Luisa")
> Peso <- c(75,85,69,56)
> Altura <- c(1.75,1.76,1.64,1.61)
> Imc <- IMC(peso=Peso,altura=Altura)
> datos <- data.frame(Nombre,Peso,Altura,Imc)
> datos
  Nombre Peso Altura      Imc
1  Juan   75   1.75 24.48980
2  Pedro  85   1.76 27.44060
3  María  69   1.64 25.65437
4  Luisa  56   1.61 21.60410
```

**3.1** Por ejemplo, con el código:

```
>levels(datos$Nombre)
[1] "Juan" "Luisa" "María" "Pedro"
>levels(datos$Nombre)[3] <- "Marta"
>datos$Imc <- round(datos$Imc)
>datos
  Nombre Peso Altura Imc
1  Juan   75   1.75  24
2  Pedro  85   1.76  27
3  Marta  69   1.64  26
4  Luisa  56   1.61  22
```

Note que para modificar el nombre de María se ha tenido que cambiar el tercer nivel de la variable *Nombre*. No se hubiese podido modificar directamente esta observación.