Tema 1:

R un entorno de análisis y programación estadísticos

1.1. Introducción

R es un entorno de análisis y programación estadísticos que, en su aspecto externo, es similar a S. y además es gratuito. Concretamente, se distribuye de acuerdo a **GNU GENERAL PUBLIC LICENSE** como puede consultarse en el propio programa escribiendo **? license**. Esta gratuidad implica, contra lo que pudiera parecer a primera vista, que un gran número de personas colaboran desarrollándolo.

Aunque comenzar a utilizar R es más complejo que comenzar a utilizar algunos programas manejados mediante menú, no es excesivamente difícil y sin embargo tiene muchas ventajas sobre ellos. A lo largo del texto veremos cómo utilizar R para realizar análisis convencionales, como los que se encuentran en BMDP, SAS, SPSS o STATGRAPHICS, y dejaremos patente en algunos momentos sus posibilidades de análisis más complejos, como GLIM o GENSTAT, así como el desarrollo de nuevos análisis.

1.2. Instalación

El primer paso para la utilización de R es instalarlo en el ordenador. R puede conseguirse en varias direcciones de Internet. Una de ellas, **http://www.r-project.org**, es la primaria y el resto son imágenes especulares (**mirrors**) de la misma. En esta dirección se encuentra todo el proyecto R. Este proyecto incluye las fuentes para construir R, lo que permite saber completamente cómo está construido el programa, así como las direcciones donde se pueden conseguir las herramientas necesarias para hacerlo. Muchas personas no desearán realizar este trámite, por lo que también existen distribuciones binarias de R. Además se encuentra allí documentación, programas en R, referencias a otras direcciones de interés, etc.

Si descarga el archivo, la instalación puede hacerla sin más que ejecutarlo,

1.3. Primer contacto

Aunque es posible leer estas notas sin un ordenador, es conveniente tener uno con el programa instalado e ir comprobando cada una de las explicaciones. Si tiene instalado el programa y lo ejecuta, puede escribir la orden

>demo(),

en la que no debe olvidar los paréntesis. El programa le contesta con una serie de opciones que puede seleccionar. Si, por ejemplo, escribe

>demo(graphics),

verá algunos de los gráficos que R es capaz de construir. Cuando termine, cierre el programa pulsando la cruz de la esquina superior derecha y seleccione la opción **No**.

1.4. Conceptos iniciales

R es un lenguaje interpretado. Esto quiere decir que puede escribir órdenes y R las ejecutará inmediatamente. Así, por ejemplo, puede escribir 2+2 o 7*2 y obtendrá los resultados esperados. De aquí se deduce que 2, 7, + y * son conceptos que el lenguaje entiende. De hecho puede utilizar R como una calculadora, ya que es capaz de manejar las operaciones elementales fácilmente. Incluso puede utilizar valores lógicos, **T** (true=verdadero) y **F** (false=falso), o números complejos, sin más que utilizar la letra para la parte imaginaria, y teniendo en cuenta el uso de paréntesis. Las operaciones de suma, resta, multiplicación, división, exponenciación, división entera y módulo se realizan

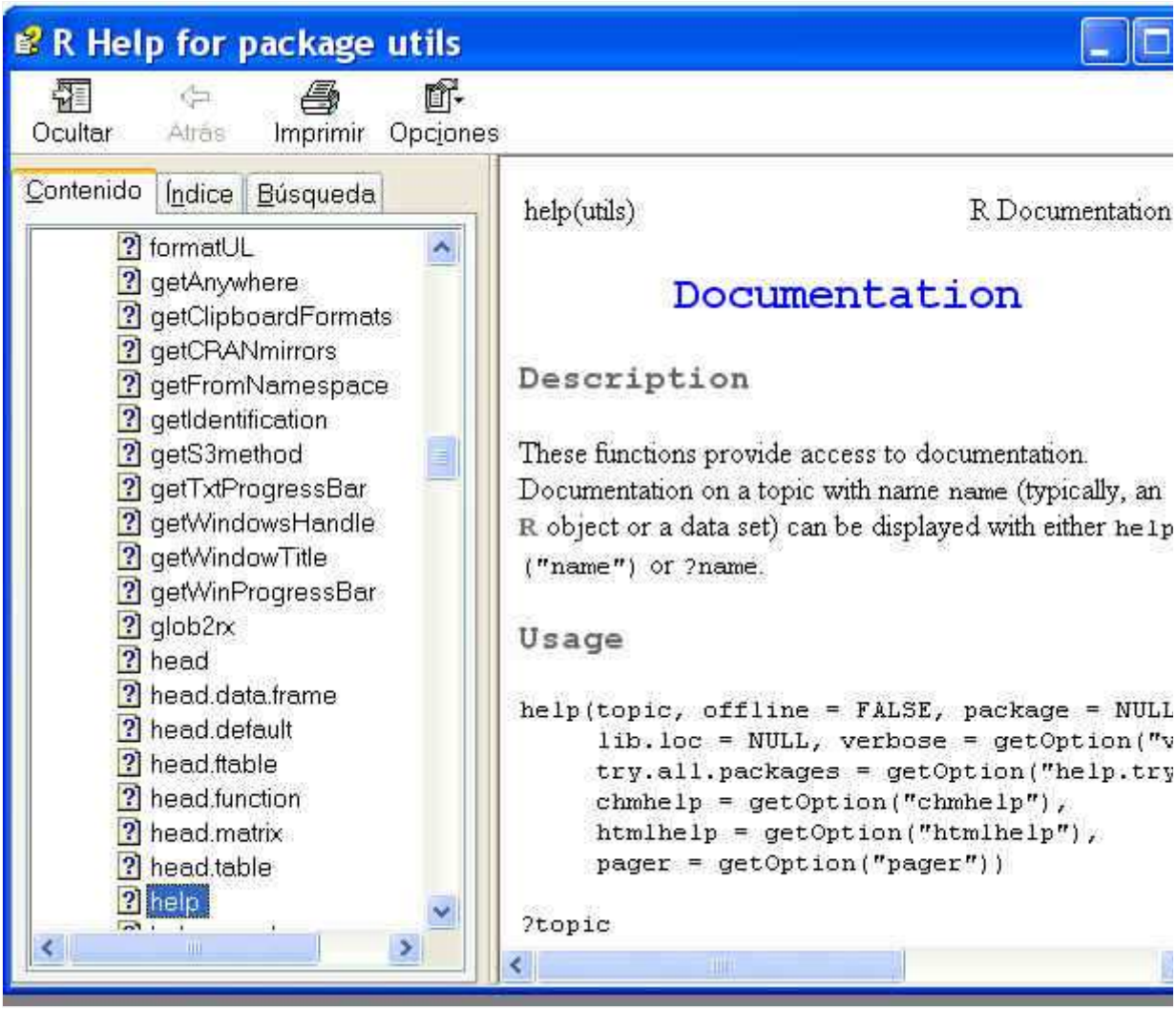
mediante los símbolos +, -, *, /, ^, %/% y %%.

```
> 2+3i*1-1i
[1] 2+2i
> (2+3i)*(1-1i)
[1] 5+1i
```

Tenga en cuenta que R no evalúa una expresión hasta que tiene la certeza de que se ha terminado su escritura. Así, si la expresión comienza con un paréntesis, podrá utilizar varias líneas para su escritura. También puede utilizar llaves, { }, con un sentido más amplio, ya que todo lo que incluyen se considera una sola acción.

```
> (2
+
+
+
+ *
+
+ 3)
[1] 6
```

R es un lenguaje funcional, esto es, realiza las tareas a través de funciones, que distingue entre mayúsculas y minúsculas. La primera función para estudiar es **help**. Si escribe **Help ()** obtendrá un error, ya que no existe esa función (la mayúscula hace que R considere una nueva función). Si escribe **help** aparece la definición de la función. Si quiere obtener el resultado de la ejecución de la función, debe escribir entre paréntesis el conjunto de argumentos que quiere suministrar a la misma, que podrá ser vacío en algún caso. Así, por ejemplo, **help()**. En este caso aparecerá la ayuda referida precisamente a la función **help ()** que ha solicitado.



En general, para cualquier función, su escritura sin los paréntesis indica al lenguaje que debe devolver la definición, en tanto que al incluir paréntesis, le indica que intente ejecutarla.

También puede llegar a la ayuda seleccionando la opción **Help** en el **menú.**

En la ayuda se encuentran a menudo ejemplos de uso. Puede copiar una parte de ellos y pegarla en la ventana de órdenes para probar su ejecución. Si desea ejecutarlos todos basta con escribir la función **example** cuyo argumento es el nombre de la función de la que desea ejecutar los ejemplos. Así, para la función **mean** puede ejecutar los ejemplos escribiendo **example(mean)**.

```
> example(mean)

mean> x <- c(0:10, 50)

mean> xm <- mean(x)

mean> c(xm, mean(x, trim = 0.1))
[1] 8.75 5.50
```

```
mean> mean(USArrests, trim = 0.2)
Murder Assault UrbanPop Rape
7.42 167.60 66.20 20.16
```

Si no conoce el nombre exacto de una función, puede pedir ayuda utilizando la función **apropos** que localiza los objetos cuyo nombre coincide parcialmente con el argumento que se le suministra.

```
> apropos("hel")
[1] "contr.helmert" ".helpForCall" "help" "help.search"
[5] "help.start" "link.html.help" "shell" "shell.exec"
```

La segunda función para estudiar es **q**. Si escribe **q** verá su definición.

```
function (save = "default", status = 0, runLast = TRUE)
.Internal(quit(save, status, runLast))
<environment: namespace:base>
```

Si escribe **help(q)** obtendrá la página de ayuda referente a la función, en donde podrá leer que permite terminar la sesión de trabajo. Si escribe **q()** la ejecutará, por lo que, en este caso, intentará terminar la sesión de trabajo. Si la escribe, R preguntará si desea salvar el espacio de trabajo, esto es, si se desea que los cambios realizados en la definición de objetos se hagan permanentes. Conteste afirmativamente si desea hacerlo y negativamente en caso contrario.

1.4.1. Teclas de edición

Puede obtener ayuda sobre las teclas de edición seleccionando en el menú la opción **Help** y a continuación **Console**. La mayoría son las comunes de **Windows**.

Para repetir cualquier orden, puede marcar esta en la pantalla y copiarla, tras lo cual puede pegarse, bien en la
propia ventana de órdenes bien en cualquier procesador desde el que posteriormente se seguirá el camino inverso.

1.4.2. Redireccionamiento

De modo análogo a **Unix** y a **MS-DOS**, si escribe una expresión, puede redireccionar su salida, de tal modo que no aparezca escrita en pantalla, sino que vaya dirigida a otro sitio. Así, si escribe

```
> 7*2->kk,
```

no obtendrá ningún resultado en pantalla, sino que este se almacena en un *objeto* de nombre **kk**.

Puede recuperar este objeto y utilizarlo en cualquier momento sin más que escribir su nombre

> kk

devuelve el valor 14:

[1] 14

Este redireccionamiento puede escribirse en la dirección en que se ha escrito o en la contraria, en la cual es más intuitivo que la parte de la izquierda se está definiendo. Así,

>kk <- -23*5,

almacena el resultado de la operación en el objeto **kk**, borrando el anterior valor almacenado.

También puede usar el símbolo **igual**, **=**, para realizar esta asignación, como en

>kk = -23*5

Los objetos así creados se almacenan en el disco duro, por lo que son permanentes, en el sentido de que aún terminando la sesión y comenzando posteriormente otra, los objetos siguen estando a disposición del usuario, siempre que salve el espacio de trabajo.

El espacio de trabajo puede salvarse en cualquier momento, eligiendo en la barra de menú, la opción **Archivo** y a continuación **Guardar área de trabajo**....

R dispone de dos funciones que permiten gestionar los objetos, que son **ls** y **rm**, y corresponden, respectivamente, a observar qué objetos existen y a eliminar un objeto concreto.

Los parámetros de una función pueden darse por dos métodos: *Por posición* o *por nombre*.

- ***Por posición*** se hace coincidir cada uno de los argumentos con el que ocupa la misma posición en la definición de la función.
- ***Por nombre***, se precede cada argumento con el nombre al que corresponde.

1.4.3. Edición de un objeto

Si desea editar un objeto, puede escribir su definición en pantalla, marcarla, copiarla y pegarla en un editor; allí editarla y luego, con el paso inverso, pegarla en la ventana de órdenes, con lo cual se actualiza. Este es el método de trabajo general de Windows, pero en otro entorno sería diferente. Es posible hacerlo con un método directo, utilizando la función **edit**.

La sintaxis es:

edit(name, file, title, editor),

donde

- **name** es el nombre de un objeto cualquiera que se desea editar, si no se especifica ninguno se utilizará el contenido de (file),
- **file** es el nombre del archivo donde debe guardarse el objeto durante la edición y que puede omitirse siempre, puesto que el sistema suministra uno
- **editor** es una variable de tipo cadena de caracteres que especifica el nombre de un editor. Si no se especifica ninguno, **ls** usa el predeterminado de Windows, *Notepad*. En este último caso, **title** se utilizará como identificador en la cabecera.

Es importante tener en cuenta que esta función no modifica el objeto, sino que el objeto editado, en su caso, es devuelto al sistema, por lo que debe redirigirse a un objeto, que puede ser otro o el mismo que se está editando, para que quede almacenado.

Así, la orden

>Tarta<-edit(Tarta,,"q")

edita el objeto **Tarta** con el editor **q** y guarda el resultado en el propio objeto **Tarta**.

Si, después de ejecutar la función, escribe **edit()**, recuperará el último texto escrito.

Ejemplo:

> Malo <- 1
> Malo <- edit(Malo)

Si en el editor escribe **function(x) x+**, al terminar la edición obtendrá

Error in edit(name, file, title, editor) :
An error occurred on line 3
use a command like
x <- edit()
to recover

y, consecuentemente, no se altera el objeto,

> Malo
[1] 1

pero puede recuperar lo escrito en el editor sin más que escribir

> Malo <- edit()

Si ahora escribe en el editor \verb|function(x) x+2|, al terminar la edición obtendrá

> Malo
function(x) x+2

Si el objeto a editar es una matriz o una hoja de datos, estructuras que se introducirán más adelante, el editor conduce a la función **data.entry**.