CAPÍTULO 7

ELEMENTOS BÁSICOS DE *R*

1.- COMENZANDO A TRABAJAR

R es un programa de computación estadística de libre distribución en Internet. Se suministra con una licencia que permite su uso de forma absolutamente gratuita. *R*, además de ser un entorno para manipular datos, efectuar análisis estadísticos y producir gráficos, es un completo lenguaje de programación, lo que hace que sea un programa tremendamente flexible.

R es un clon del programa comercial *S-PLUS*, el cual está escrito en el lenguaje de programación estadística *S*, y del que *R* puede ser considerado como un "dialecto".

Para obtener el programa hay que acceder a la página de Internet http://cran.r-project.org/ y elegir el *mirror site* más próximo para descargarlo de la forma más rápida. En la misma página web se obtiene la información necesaria para instalar el programa.

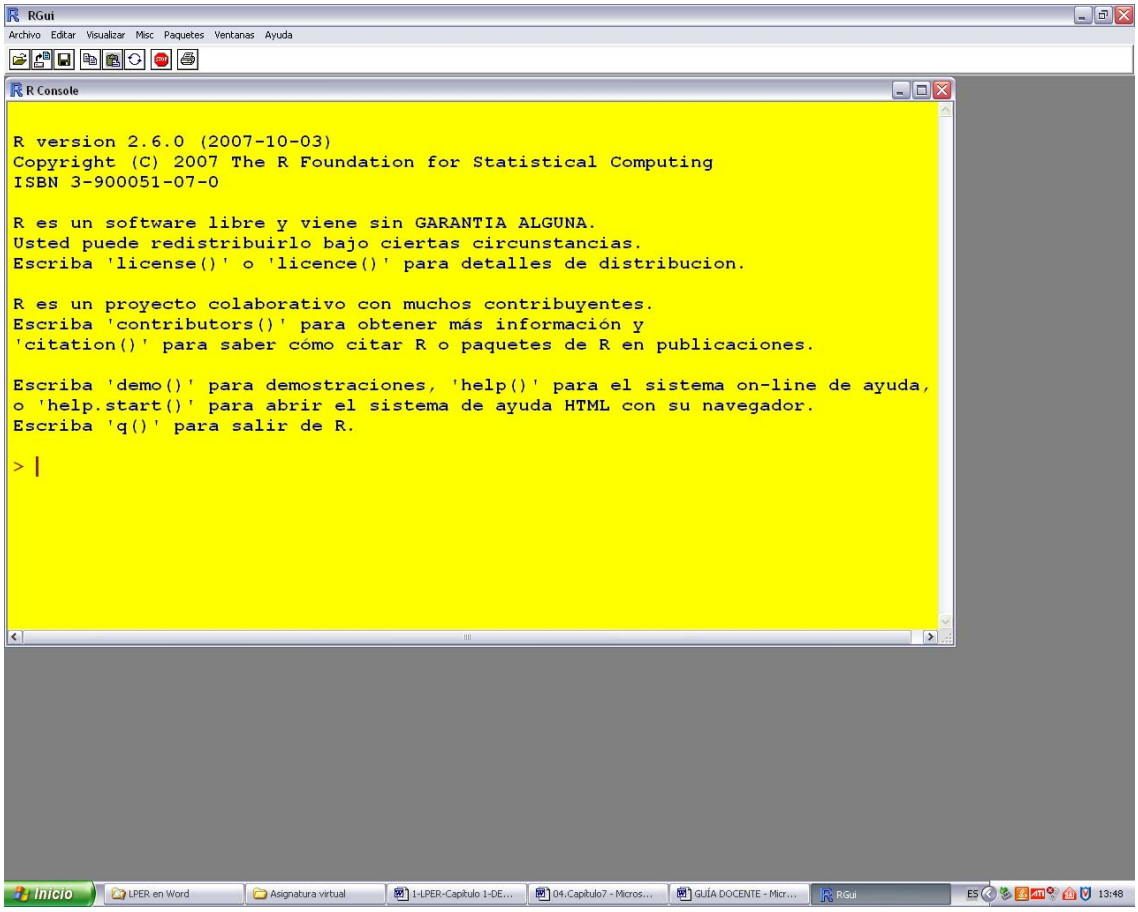
Existe otra interesante alternativa que es la utilización on-line de *R,* sin necesidad de instalar el programa en el propio ordenador. Para utilizar este servicio, obviamente gratuito, basta acceder a la página web http://bayes.math.montana.edu/Rweb/Rweb.general.html. Como resulta cada vez más habitual que los teléfonos móviles y otros dispositivos dispongan de conexión a Internet, con un simple móvil se puede acceder a un potente software y, como consecuencia de ello, a unas completas tablas estadísticas.

Una de las aplicaciones más simples, pero a la vez más prácticas de *R*, es su uso como calculadora, presentando una ventaja considerable, en cuanto a comodidad en el cálculo.

Los operadores correspondientes a las funciones elementales, y la constante π son los siguientes:

Suma	`+`
Diferencia	`-`
Producto	`*`
Cociente	`/`
Potencia	`^`
Raíz cuadrada	`sqrt`
Logaritmo natural	`log`
Función exponencial	`exp`
Seno	`sin`
Coseno	`cos`
Tangente	`tan`
Constante π	`pi`

Una vez iniciado *R* aparece la siguiente ventana:



Esta ventana es la consola en la que introduciremos las diferentes sentencias y en la que obtendremos los correspondientes resultados. En el menú **Editar → Preferencias** de la interface gráfica se puede cambiar el color del fondo y otros elementos.

Trabajar con *R* es como mantener una conversación; introducimos una sentencia (pregunta) y *R* responde con un resultado:

```
> 8+9
[1] 17
```

Para asignar un valor a una variable se utilizan los símbolos **<-**. Para insertar comentarios basta anteponerles el símbolo #:

```
> # Ahora vamos a definir la variable x
> x<-5
> x
[1] 5
> x^3
[1] 125
> x*8
[1] 40
```

Cuando falta algo para completar una sentencia aparece el símbolo+:

```
> sqrt(3
+
```

Si completamos ahora la expresión, en este caso con el cierre del paréntesis, el programa escribe el resultado:

```
> sqrt(3
+ )
[1] 1.732051
```

2.- INTRODUCCIÓN Y SALIDA DE DATOS

Para construir un vector se utiliza la sentencia **c()**:

```
> pesoenkg<-c(5,2.8,3.7,4.6,8.1,3.2)
> pesoenkg
[1] 5.0 2.8 3.7 4.6 8.1 3.2
> pesoengr<-pesoenkg*1000
> pesoengr
[1] 5000 2800 3700 4600 8100 3200
```

En lugar de utilizar la construcción anterior se puede utilizar la sentencia **scan().** De este modo los datos se introducen de uno en uno mediante la tecla **Enter**. Para indicarle al programa que hemos terminado de introducir datos se debe dar dos veces a esa tecla:

```
> volumen<-scan()
1: 23
2: 21
3: 23
4: 45
5:
Read 4 items
> volumen
[1] 23 21 23 45
```

Para construir series de valores, por ejemplo los números impares comprendidos entre 1 y 65, hacemos:

```
> seq(1,65,2)
[1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29
31 33 35 37 39 41 43 45 47 49
[26] 51 53 55 57 59 61 63 65
```

Mediante la función **rep()** es posible repetir un patrón dado, incluso caracteres:

```
> rep(8,4)
[1] 8 8 8 8
> rep(98,5)
[1] 98 98 98 98 98
> rep(c("sí","no"),3)
[1] "sí" "no" "sí" "no" "sí" "no"
```

Si se desea cargar datos desde un archivo externo, por ejemplo si se quiere leer el archivo *Tensión de rotura.txt* que tenemos almacenado en la unidad C, debemos especificar con total exactitud primero la ruta y finalmente el nombre del archivo, todo ello entre comillas. Es conveniente poner el símbolo \\ en lugar de \:

```
> tensión<-read.table("C:\\Tensión de rotura.txt")
> tensión
     V1
1  4.05
2  4.58
3  4.42
...
49 3.54
50 4.84
```

La función **`read.table()`** genera un objeto del tipo “marco de datos” o *data frame.* Con la función **`class`** podemos comprobar qué tipo de objeto es **`tensión:`**

```
> class(tensión)
[1] "data.frame"
```

Supongamos ahora que tenemos el siguiente archivo Excel, que hemos guardado como *Texto delimitado por tabulaciones* con el nombre *DF.txt*:

Costo.unit	Costo.mat	Costo.mano.de.obra
13.59	87	80
15.71	78	95
15.97	81	106
20.21	65	115
24.64	51	128

Para leer este archivo hacemos:

```
> df<-read.table("C:\\DF.txt",header=T)
> df
  Costo.unit Costo.mat Costo.mano.de.obra
1      13.59        87                  80
2      15.71        78                  95
3      15.97        81                 106
4      20.21        65                 115
5      24.64        51                 128
```

El argumento **`header=T`** significa que la primera línea del archivo leído contiene los nombres de las variables.

Con **`attach(df)`** las variables del marco de datos **`df`** son accesibles por su nombre a partir de ese momento, y con **`names(df)`** se consigue una lista de las variables existentes:

```
> attach(df)
```

```
> names(df)
[1]  "Costo.unit"                    "Costo.mat"
"Costo.mano.de.obra"
> Costo.mat
[1] 87 78 81 65 51
```

Para generar marcos de datos en una sesión de *R* utilizamos la función **`data.frame()`**. Por ejemplo, vamos a generar el marco de datos denominado **`valores:`**

```
> a<-c(1,3,5,7,9)
> b<-c(3.2,1.3,1.2,4.5,7)
> valores<-data.frame(a,b)
> valores
  a   b
1 1 3.2
2 3 1.3
3 5 1.2
4 7 4.5
5 9 7.0
```

Si ahora queremos exportar el marco de datos anterior y guardarlo en el disco C como un archivo de texto de nombre **`salida.txt`**, hacemos lo siguiente:

```
> write.table(valores,file="C:\\salida.txt")
```

Algunos paquetes (*packages*) para aplicaciones especiales son parte de la instalación básica de *R*; otros pueden ser obtenidos en la misma página web de donde se descarga el programa. Para ver qué paquetes están instalados hacemos

```
> library()
```

y, por ejemplo, para cargar el paquete **`datasets`**, hacemos

```
> library(datasets)
```

o bien en el menú usamos la herramienta **Paquetes.**

Para ver el contenido de **`datasets`** hacemos

```
> library(help=datasets)
```

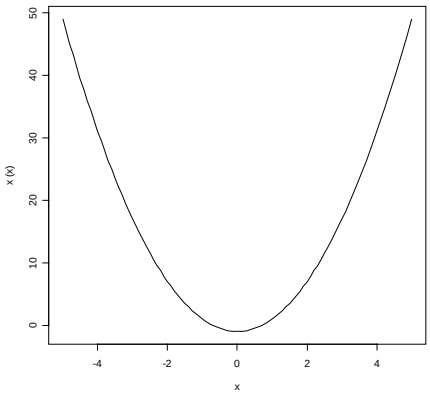
3.- FUNCIONES

Para definir nuevas funciones en *R*, como por ejemplo $f(x) = 2x^2–1$, se hace:

```
> f<-function(x) 2*x^2-1
> f(3)
[1] 17
```

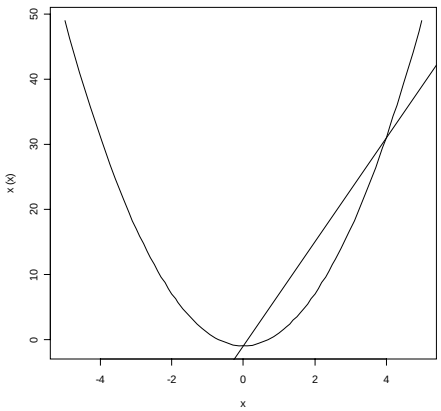
Podemos dibujar un gráfico de la función **`f`**, entre las abscisas –5 y 5:

```
> plot(f,-5,5)
```



Si queremos superponer sobre el gráfico anterior la recta de ecuación y = 8x-1 hacemos:

```
> abline(-1,8)
```



Los gráficos que genera *R* son interactivos, en el sentido de que es posible interactuar con ellos para obtener, por ejemplo, las coordenadas de un punto. Una de las funciones que permite interactuar con un gráfico es **`locator().`** Supongamos que estamos interesados en conocer las coordenadas de los puntos de intersección entre la recta y la parábola de la figura. Entonces, poniendo como argumento de esa función **n=2** para indicar que buscamos dos puntos, pulsando Enter y, por último, accionando el botón izquierdo del ratón al colocar éste sobre los puntos correspondientes, obtenemos sus coordenadas.

```
> locator(n=2)
$x
[1] -0.009351335  4.012124941
$y
[1] -0.9119548 31.3391999
```

Debido a la dificultad de posicionar con mucha precisión el ratón sobre los puntos de intersección, los valores obtenidos son aproximados. Las coordenadas exactas son (0,-1) y (4,31).

También se pueden definir funciones de dos o más variables y aprovechar la forma en que trabaja *R* con vectores para realizar todas las operaciones con una sola evaluación. Supongamos, por ejemplo, que queremos evaluar la función $z = \frac{x^2 + y^2}{x + y}$ en 5 puntos (x,y). Podemos comenzar creando los vectores que contienen los valores de interés:

```
> x<-c(1,3,4,2,5)
> x
[1] 1 3 4 2 5
> y<-c(3,2,4,2,3)
> y
[1] 3 2 4 2 3
```

Finalmente definimos la función z, cuyo valor será evaluado en los puntos (x,y) definidos anteriormente:

```
> z<-(x^2+y^2)/(x+y)
> z
[1] 2.50 2.60 4.00 2.00 4.25
```

4.- CÓMO GESTIONAR UNA SESIÓN DE *R*

Al iniciar una sesión de *R* hay un directorio de trabajo donde el programa busca, por defecto, cualquier elemento que sea solicitado y donde coloca los elementos que se crean durante la sesión. Es conveniente utilizar distintos directorios para distintos proyectos. Para ver en qué directorio se está trabajando se hace

```
> getwd()
[1] "C:/Archivos de programa/R/R-2.6.0"
```

Si queremos cambiar de directorio elegimos en el menú

Archivo → Cambiar dir…

Todos los *objetos* (variables) creados en *R* se almacenan en un área de trabajo o *workspace* (puede haber varios). Para ver qué objetos se han definido elegimos en el menú

Misc → Listar objetos

y para eliminarlos todos

Misc → Remover todos los objetos

Si sólo se quieren eliminar, por ejemplo, los objetos **pesoenkg** y **pesoengr** hacemos

```
> rm(pesoenkg,pesoengr)
```

Es posible guardar el área de trabajo en un archivo en cualquier momento haciendo

Archivo → Guardar área de trabajo…

El *workspace* está formado sólo por objetos, y no por ninguna entrada obtenida durante la sesión. Si se desea guardar las entradas, con sus correspondientes salidas, hay que utilizar

Archivo → Guardar en Archivo…

o bien **Cortar y Pegar**. Así mismo es interesante la opción **Imprimir...** para obtener un archivo PDF con el registro completo de la sesión.

La historia de los comandos introducidos en una sesión puede ser guardada y cargada mediante

Archivo → Guardar Histórico.../Cargar Histórico...

De este modo podríamos acceder, en orden inverso, a los comandos introducidos en la sesión anterior.

Para ver todos los comandos introducidos durante una sesión hacemos

```
> history(Inf)
```

Una forma muy práctica de gestionar una sesión de *R* consiste en lo siguiente: una vez terminada la sesión, copiamos y pegamos la salida de la instrucción **history(Inf)** en un editor de texto, generando un archivo de extensión .txt. Este archivo es susceptible de ser manipulado, cambiando datos, introduciendo nuevas variables, etc. Posteriormente se copia y se pega en *R,* obteniéndose las salidas correspondientes a los cambios efectuados. Veamos, a continuación, un ejemplo.

Supongamos que, partiendo de los datos del vector x, queremos obtener sus cuadrados, sus cubos y la suma de todos los valores al cuadrado. Haríamos lo siguiente:

```
> x<-c(1,-2,3,-8.23,9.06,-2.8,6,-0.5,3,-4)
> x
 [1]  1.00 -2.00  3.00 -8.23  9.06 -2.80  6.00 -
0.50  3.00 -4.00
> y<-x^2
> y
 [1]   1.0000   4.0000   9.0000  67.7329  82.0836
7.8400 36.0000  0.2500  9.0000
[10] 16.0000
```

```
> sum(y)
[1] 232.9065
> history(Inf)
```

La salida de la última instrucción la introducimos en un editor de texto obteniendo el archivo “Documento1.txt”:

```
x<-c(1,-2,3,-8.23,9.06,-2.8,6,-0.5,3,-4)
x
y<-x^2
y
sum(y)
```

Supongamos ahora que el primer elemento del vector x lo cambiamos, y en lugar de 1 ponemos 13, y en vez de calcular los cuadrados de x calculamos las potencias cuartas. Efectuamos esos cambios en el archivo “Documento1.txt”, después lo seleccionamos todo y lo pegamos en *R* obteniendo lo siguiente:

```
> x<-c(13,-2,3,-8.23,9.06,-2.8,6,-0.5,3,-4)
> x
 [1] 13.00 -2.00  3.00 -8.23  9.06 -2.80  6.00 -
0.50  3.00 -4.00
> y<-x^4
> y
 [1] 28561.0000     16.0000     81.0000   4587.7457
6737.7174    61.4656
 [7]  1296.0000     0.0625    81.0000   256.0000
> sum(y)
[1] 41677.99
```

5.- ALGUNAS OBSERVACIONES IMPORTANTES

- La tecla ↑ sirve para copiar la última sentencia editada.
- Para designar objetos en *R* se distingue entre mayúsculas y minúsculas.
- En el menú **Ayuda** se pueden obtener muchos tipos de ayudas, incluidos manuales completos de *R*. Concretamente, haciendo clic en **Ayuda → Funciones R (texto)...** se puede obtener ayuda sobre una función concreta de *R*.
- Para conseguir ayuda si se está conectado a Internet se puede hacer **Ayuda → FAQ en R**, o bien utilizar la sentencia **`help.start()`**.
- Para insertar un gráfico de *R* en un editor de texto copiamos el gráfico como *metafile* y luego lo pegamos.
- Si tenemos abierta una ventana gráfica, por ejemplo porque acabamos de generar un gráfico y, manteniendo esa ventana, queremos abrir otra podemos usar la función **`windows()`**, o bien **`x11()`**.