Tema 2

Computación en modelización

2.1 Sobre R

El lenguaje S, desarrollado en los laboratorios de la compañía AT&T por Rick Becker, John Chambers y Allan Wilks, pretende simplificar tanto el almacenamiento de datos como su tratamiento estadístico. De este lenguaje se han realizado varias implementaciones, algunas de ellas comerciales[1].

La utilización de R se ha extendido de forma creciente y los investigadores lo han seleccionado por sus ventajas, como puede comprobarse en el creciente número de trabajos publicados en la revista *Journal of Statistical Software*, publicada por la `American Statistical Association`[2]. Entre ellas:

- se trata de un programa de distribución libre
- puede utilizarse con distintos sistemas operativos
- adecuación del lenguaje al manejo y análisis de los datos
- sencillez con la que pueden combinarse los distintos análisis estadísticos
- gráficos de alta calidad: visualización de datos y producción de gráficos, `ver ejemplos`[3]
- facilidad para incorporar nuevos procedimientos a los proporcionados por el sistema base
- muy completa documentación, que crece día a día
- la comunidad de R es muy dinámica, con gran crecimiento del número de paquetes, e integrada por científicos de gran renombre
- hay extensiones específicas a nuevas áreas como bioinformática, geoestadística, etc.
- es un lenguaje orientado a objetos
- se parece a Matlab y a Octave, y su sintaxis recuerda a C/C++

Sintetizando, puede describirse a R como un entorno integrado para trabajar con el lenguaje S, que proporciona:

- Un conjunto coherente y extensivo de instrumentos para el análisis y el tratamiento estadístico de datos.

[1] La aplicación comercial más conocida es S–plus.
[2] http://www.jstatsoft.org
[3] http://addictedtor.free.fr/graphiques/thumbs.php?sort=votes

- Un lenguaje para expresar modelos estadísticos y herramientas para manejar modelos lineales y no lineales.
- Utilidades gráficas para el análisis de datos y la visualización en cualquier estación gráfica o impresora.
- Un eficiente lenguaje de programación orientado a objetos, que crece fácilmente merced a la comunidad de usuarios.

Puede encontrarse abundante información sobre el funcionamiento del R en la `página principal proyecto`[4]. Se han publicado numerosos textos de tratamiento y análisis de datos con R (`libros de R y S`[5]). En la página de `wikipedia`[6] pueden encontrarse más referencias.

2.1.1 Inconvenientes de R

El principal inconvenientes de R radica en la formación inicial de sus usuario. Si estos conocen un lenguaje de programación se pueden adaptar con facilidad. Por contra, si están acostumbrados a entornos gráficos con una GUI (graphical user interface) el aprendizaje se hace más pesado. Puede utilizarse como ayuda un interfaz gráfico para R llamado `Rcmd`[7].

Otra dificultad de R está asociada a un lenguaje muy rico. La solución se basa en un potente y completo sistema de ayudas y documentación.

2.1.2 Uso de R

El uso de R se basa en un lenguaje que utiliza expresiones que son evaluadas por el programa proporcionando el resultado en forma de valores o gráficos.

Las expresiones componen de operadores, valores numéricos, valores lógicos y funciones.

2.1.3 Instalación de R

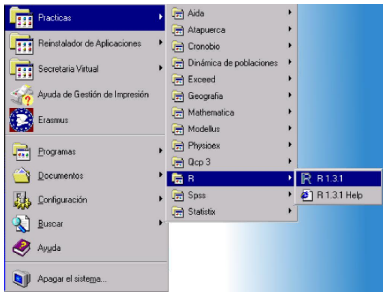
La instalación de R es relativamente sencilla y varia de un sistema operativo a otro, adecuándose a los procedimientos habituales en cada uno de ellos.

- winXX. Se accede a la página correspondiente a CRAN o `Comprehensive R Archive Network`[8] y se elige un servidor, preferiblemente el correspondiente a RedIris. La entrada `Windows (95 and later)` y posteriormente: `base` nos permiten seleccionar: `R-2.2.1-win32.exe`. A partir de este fichero se produce la instalación (la referencia de la versión puede cambiar con nueva versiones).
- Linux. Dependiendo de la distribución se instalará el paquete correspondiente, en el caso de la distribución debian este es: `r-base`.
- ...

[4] http://www.r-project.org
[5] http://www.r-project.org/doc/bib/R-books.html
[6] http://en.wikipedia.org/wiki/R_programming_language
[7] http://socserv.socsci.mcmaster.ca/jfox/Misc/Rcmdr/
[8] http://cran.r-project.org/

2.2 Primeros pasos con R

Los usuarios de un sistema winXX utilizarán el menú del sistema, tal como muestra la figura:



Para usuarios de sistemas Linux bastará con invocarlo desde la línea de comandos, escribiendo a continuación del *prompt* del sistema el nombre del programa `R`:

```
[usuario@directorio]$ R
```

El sistema entrará en el programa y ofrecerá información básica sobre él y quedará a la espera de recibir una orden[9]:

```
R : Copyright 2005, The R Foundation for Statistical Computing
Version 2.1.0  (2005-04-18), ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for a HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

>
```

El signo *mayor que* en la última línea indica que podemos escribir una expresión con la sintaxis de R. Tras pulsar retorno de carro este la evaluará y actuará en consecuencia.

Para abandonar el programa bastará con escribir `q()` y el resultado será:

```
> q()
Save workspace image? [y/n/c]:
```

que nos permite:

- y guardar los datos incorporados al programa y el histórico con las ordenes utilizadas durante la sesión de trabajo.

[9] esta presentación puede variar ligeramente de una versión a otra del programa.

n perder datos e histórico de ordenes.

c cancelar la orden dada para abandonar el programa y permanecer en la sesión de trabajo.

La ventaja de conservar los datos y el histórico es que permite retomar la sesión de trabajo con posterioridad en el punto donde la abandonamos.

Al iniciar una sesión, si existe un histórico bastará pulsa la tecla $\boxed{\uparrow}$ para acceder las expresiones escritas con anterioridad. Además, la expresiones pueden modificarse con la ayuda de las $\boxed{\leftarrow}$ y $\boxed{\rightarrow}$.

2.2.1 Ayuda en R

Una de las cualidades de R es su sistema de documentación y ayuda. Para obtener ayuda sobre distintos aspectos del programa puede utilizarse la orden `help()` o `help.start()`.

Además se dispone de algunas demostraciones del uso, para localizarla se utiliza la orden `demo()`.

También es posible utilizar los ejemplo que aparecen en la ayuda con la función `example()`

Las funciones de ayuda en R son:

- `help()`: proporciona ayuda sobre el sistema de ayuda
- `help(clave)`: proporciona ayuda sobre una palabra clave
- `?clave`: es una forma abreviada de la anterior
- `help.start()`: proporciona ayuda sobre R en un navegador, si se utiliza todas las ayudas se canalizan por el navegador.
- `example(clave)`: Muestra los resultados propuestos en el ejemplo de la palabra clave.
- `demo()`: muestra la relación de “demos” disponibles
- `library()`: muestra la relación de bibliotecas de funciones disponibles
- `data()`: muestra la relación de datos de ejemplo disponibles
- `apropos("texto")`: Muestra la relación de términos que en su nombre incluyen el texto y para los que se encuentra información disponible

2.2.2 Expresiones en R

R utiliza una sintaxis muy sencilla en la que la unidad está constituida por la *expresión.* La expresión más sencilla es un simple número entero.

```
> 1
[1] 1
```

R evalúa la expresión y devuelve un valor de esta. La notación `[1]` indica que el primer valor de la línea de respuesta, es el primer valor.

Los valores pueden relacionarse mediante operadores (lógicos o algebraicos) y funciones.

```
> sqrt(3^2 + 5^2)
[1] 5.830952
```

Insistiremos más adelante en las funciones (aquí se ha utilizado la correspondiente a la raíz cuadrada).

En R tenemos los siguientes operadores que puede utilizarse para:

`+, -, *, /`	suma, resta, producto, cociente
`%%, %/%, ^`	módulo, cociente entero, potencia
`==, !=, !`	igual, distinto, no
`>, >=, <, <=`	mayor que, mayor o igual que, menor que, menor o igual que
`|, ||, &, &&`	y, y, o, o
`<-, ->, _`	asignar a la izquierda, asignar a la derecha, asignar a la izquierda
`:`	generar una serie

Para obtener información de los distintos operadores puede solicitarse la ayuda mediante `?"+"`.

Cada operador implica a los dos elementos que se sitúan a izquierda y derecha en la expresión, existe una jerarquía de operadores que determina cual de ellos se evalúa primero en la expresión, el orden de evaluación puede alterarse mediante el uso de paréntesis. Por ejemplo:

```
> 9 * 5 / 2
[1] 22.5
> 9 * 5 / 2 * 3
[1] 67.5
> 9 * 5 / (2 * 3)
[1] 7.5
> 9 * 5 / 2 / 3
[1] 7.5
```

que efectúan $\frac{9 \times 5}{2}$, $\frac{9 \times 5 \times 3}{2}$, $\frac{9 \times 5}{2 \times 3}$ y, equivalentemente $\frac{9 \times 5}{2 \times 3}$.

Los operadores lógicos proporcionan como resultado el valor VERDADERO (anotado por `TRUE`, o `T`) o el valor FALSO (anotado por `FALSE`, o `F`)

```
> 3 != 2
[1] TRUE
```

Los espacios en blanco son obviados por R y, en todo caso, deben utilizarse para que la expresión quede clara al lector. Por ejemplo, la expresión: `2^(3/2)/(5*6)`, quedará más legible utilizando: `2 ^ (3/2) / (5*6)`.

Cuando se necesita más de una expresión estas se pueden escribir en líneas separadas, si bien, puede reunirse en una sola línea más de una expresión separándolas por “;”.

En algunos casos es posible escribir una expresión en más de una línea, así cuando R entiende que la expresión es incompleta devuelve un *prompt* $+$ en lugar del habitual $>$.

2.2.3 Variables

Podemos utilizar variables para realizar una asignación, es decir, guardar el resultado de una expresión:

```
> a <- 1
>
```

El programa no devuelve mensaje alguno, pero desde ahora la variable `a` contiene valor 1, así si escribimos:

```
> a
[1] 1
```

el sistema nos devuelve el valor de la variable.

La asignación se realiza mediante el signo compuesto: `<-` o `->`, y en la actualidad R admite el uso de signo `=` para la asignación, así son equivalentes las siguientes expresiones: `a <- 1`, `1 -> a` y `a = 1`; sin embargo, no puede utilizarse `1 = a`.

La asignación significa una destrucción del valor de la variable, tras efectuarla se pierde el viejo valor y se queda el que resulta de la expresión asignada.

```
> a <- 1
> a <- 5
> a
[1] 5
```

Los nombres de las variables deben comenzar obligatoriamente por una letra, distinguiéndose entre mayúsculas y minúsculas, y a continuación, opcionalmente, una combinación de letras y números, a ellos puede incorporarse el punto "."; así son nombre válidos: `a, A, A.1, altura, densidad,` ...Los nombre de las variables pueden coincidir con los nombre de funciones, aunque resulta poco aconsejable.

Resulta conveniente que los nombres de las variables mantenga una relación con los valores que contienen: así, la variable que describe la 'cobertura lineal del palmito (*Chamaerops humilis*)' puede indicarse como: `cl`, haciendo referencia a la idea de cobertura lineal, si no tenemos otras especies; `cl.chahum`, como en el caso anterior pero cuando tenemos varias especies; o `chahum`, cuando solo tenemos información para esta especie de su cobertura lineal.

Además de valores numéricos podemos utilizar las cadenas de texto. Para indicarle al ordenador que toda la cadena es un sólo objeto, esta debe entrecomillarse `a="Hola"`

```
nombre.estacion<-"Pluviómetro de San Javier"
```

Las variables pueden contener valores sencillos, atendiendo a su naturaleza tenemos:

Lógicos:	TRUE,FALSE,T,F
Enteros:	-10, 1, 1000, . . .
Precisión doble:	-10.1, 6.02310e24, . . . , -Inf, Inf, NaN
Complejos:	1+3i, 1+0i, 9i, . . .
Carácter:	"Hola", "Enero", "sin(x)", "pino", . . .
Perdidos:	Na

también cabe la posibilidad de tener más de un valor en una variable, siendo entonces, según la estructura:

Vector	conjunto ordenado de datos del mismo tipo básico.
Array	vector con atributo de dimensión, es válido cualquier número de dimensiones.
Matriz	es un array con dos dimensiones
Factor	tipo especial de vector en el que se codifican las clases.
Lista	conjunto de elementos que pueden ser de distintos tipos.
Estructura de datos	mezcla de matriz y lista.

Para este capítulo consideraremos, por simplificar, vectores y matrices y no las otras estructuras de datos que se verán más adelante.

2.2.4 Vectores

Para disponer de un vector basta con asignar a una variable un conjunto de valores:

```
> x <- c(1,22,9,8,36,12)
```

crea el vector `x` siendo $x_1 = 1, x_2 = 22, \ldots, x_6 = 12$, con la ayuda de la función `c()` que permite concatenar una serie de valores. Los valores de `x` pueden consultarse sencillamente:

```
> x
[1]  1 22  9  8 36 12
```

En el caso de vectores sus elementos pueden referirse por su posición mediante el subíndice:

```
> x[3]
[1] 9
```

como puede apreciarse claramente, los subíndices se indican entre corchetes.

Un operador interesante es ":", así, si escribimos:

```
> 1:5
[1] 1 2 3 4 5
```

obtenemos cinco valores consecutivos: los valores del uno al cinco. Podemos utilizar este operador para generar series. Los rangos así obtenidos pueden utilizarse en distintos casos, por ejemplo como subíndices:

```
> x[1:3]
[1]  1 22  9
```

con esta expresión seleccionamos los tres primeros elementos del vector `x`.

Cuando no se expresa subíndice alguno no referimos al vector completo, por ello `x` y `x[ ]` son expresiones equivalentes.

Un subíndice negativo equivale a eliminar el elemento indicado del vector, así: `x[-5]`, indica el vector `x` excluido el quinto elemento.

En ocasiones necesitamos seleccionar de un vector elemento condicionados a una propiedad, por ejemplo, si deseamos utilizar sólo los elementos del vector con valor par lo expresamos: `x[x%%2==0]`.

2.2.5 Funciones

En R la mayor parte del trabajo implica el uso de funciones, por ejemplo:

```
> max(x)
[1] 36
```

nos devuelve el máximo valor que presenta un elemento del vector *x*. El número y utilidad de las funciones de R es enorme y variado. Además, y esta es una de las virtudes de R, pueden definirse fácilmente nuevas funciones.

Las funciones se indican con una palabra clave, o nombre de la función, y entre paréntesis —opcionalmente— parámetros. Por ejemplo, la función `q()`, que se utiliza para abandonar R, no necesita parámetro alguno; la función `sqrt()` necesita un valor para obtener su raíz cuadrada —está claro que este valor puede ser el resultado de una expresión compleja—; y la función `matrix()` necesita 3 parámetros: el conjunto de elementos de la matriz, número de filas y el número de columnas;

```
> matrix(12:1,4,3)
     [,1] [,2] [,3]
[1,]   12    8    4
[2,]   11    7    3
[3,]   10    6    2
[4,]    9    5    1
```

Entre otras podemos destacar las siguientes funciones:

`c()`	Concatenar los elementos que se indican, separados por comas.
`seq()`	Generar una secuencia numérica.
`rep()`	Generar un conjunto de valores repetidos.
`t()`	Transponer una matriz.
`sqrt()`	Raíz cuadrada.
`abs()`	Valor absoluto.
`sin(), cos(), ...`	Funciones trigonométricas.
`log(), exp()`	Logaritmo y exponencial.
`round()`	Redondeo de valores numéricos.
`ls()`	Relación de objetos disponibles.
`rm()`	Elimina uno o varios objetos.
`for(), while()`	Evalúa una o un conjunto de expresiones repetitivamente.
`if(), ifelse()`	Evalúa una expresión condicionalmente.

Podemos definir con sencillez una función propia, por ejemplo para pasar de grados centígrados a grados Fahrenheit, mediante:

```
> mifuncion<-function(x) x*9 / 5 + 32
```

Puede calcularse la transformación sin más que, por ejemplo, calcular cual el la temperatura de ebullición del agua en grados Fahrenheit:

```
> mifuncion(100)
[1] 212
```

que proporciona lógicamente el valor esperado.