



# MANUAL DE PRÁCTICAS CON R

## 1. INTRODUCCIÓN

R, también conocido como “GNU S”, es un entorno y un lenguaje de programación para el cálculo estadístico y la generación de gráficos. R implementa un dialecto del premiado lenguaje S, desarrollado en los Laboratorios Bell por John Chambers et al. Provee un acceso relativamente sencillo a una amplia variedad de técnicas estadísticas y gráficas y ofrece un lenguaje de programación completo, bien desarrollado, con el que añadir nuevas técnicas mediante la definición de funciones. Entre otras características, dispone de almacenamiento y manipulación efectiva de datos; operadores para cálculo sobre variables indexadas (arrays) y en particular matrices; una amplia, coherente e integrada colección de herramientas y posibilidades gráficas para análisis de datos.

R es un entorno en el que se han implementado muchas técnicas estadísticas, tanto clásicas como modernas. Algunas están incluidas en el entorno base de R y otras se acompañan en forma de bibliotecas (packages, disponibles en <http://www.r-project.org>). Actualmente, R y S son los dos lenguajes más utilizados en investigación en estadística.

Existe una diferencia fundamental en la filosofía que subyace en R (o S) y la de otros sistemas estadísticos. En R, un análisis estadístico se realiza en una serie de pasos, con unos resultados intermedios que se van almacenando en objetos, que pueden ser observados o analizados posteriormente, produciendo unas salidas mínimas. Sin embargo en SAS o SPSS se obtendrá de modo inmediato una salida copiosa para cualquier análisis, por ejemplo, una regresión o un análisis discriminante.

Los grandes atractivos de R/S son:

- La capacidad de combinar, sin fisuras, análisis “preempaquetados” (por ejemplo, una regresión logística) con análisis ad-hoc, específicos para una situación.
- La capacidad de manipular y modificar datos y funciones.
- Los gráficos de alta calidad: visualización de datos y producción de gráficos.
- La comunidad de R es muy dinámica, con gran crecimiento del número de paquetes, e integrada por estadísticos de gran renombre.
- Hay extensiones específicas a nuevas áreas como bioinformática, geoestadística y modelos gráficos.
- Es un lenguaje orientado a objetos.
- Se parece a Matlab y a Octave, y su sintaxis recuerda a C/C++.
- Es gratuito y su descarga e instalación son sencillas

Entre otros, vamos a utilizar en estas prácticas el paquete R Commander (Rcmdr), que proporciona un entorno gráfico que facilita el manejo de R y sirve como generador de instrucciones. Para tareas sencillas, es posible que no se necesite otro nivel de uso que el que proporciona Rcmdr. Esta herramienta es bastante reciente y está creciendo de una manera bastante rápida. Para situaciones

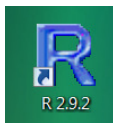
más complejas de trabajo e investigación, una vez superado el respeto inicial a la herramienta, lo habitual es decantarse por trabajar directamente con la consola de R, creando y editando instrucciones con una evidente economía de recursos y, lo que es más importante, con un control total sobre los procedimientos que en cada momento se van a aplicar.

## 1.1 INSTALACIÓN E INICIO DE R Y RCMDR

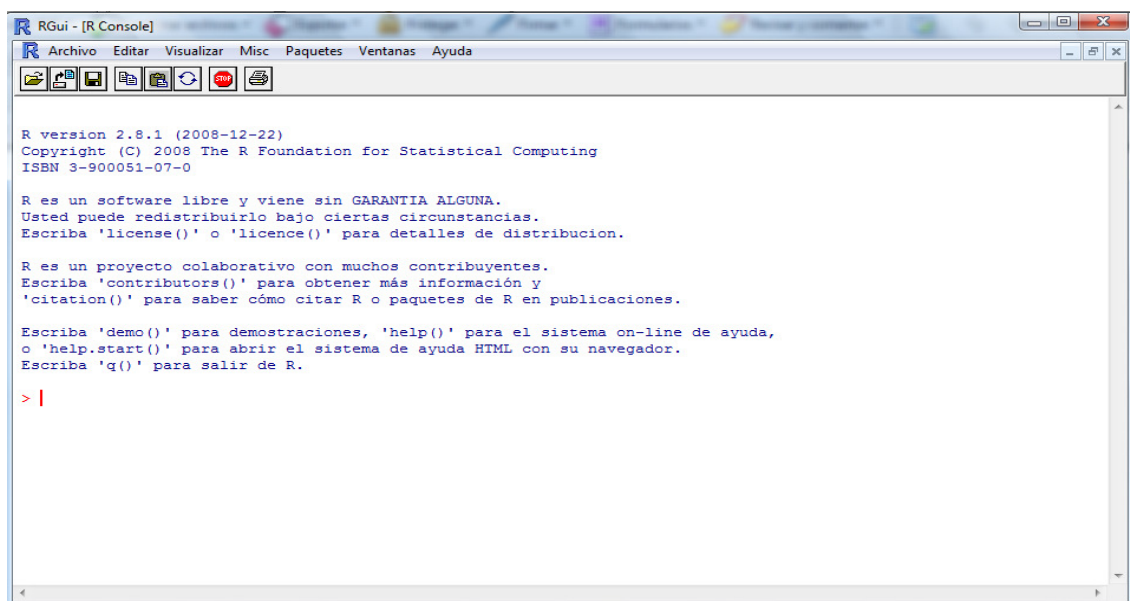
1. Descargar el archivo de instalación o el código que corresponda al sistema operativo del ordenador en la URL <http://www.r-project.org> (download R), por ejemplo desde Madrid (<http://cran.es.r-project.org/>). No es necesario instalar las contribuciones adicionales, basta con el subdirectorio base.
2. Para instalar un paquete, como puede ser Rcmdr, el programa tiene integrado un gestor de paquetes. Hay que:
  1. Entrar en el menú *Paquetes -> Instalar paquete(s)...*
  2. En la ventana que aparece, elegir un repositorio (o espejo) desde el que descargar el paquete. Cuanto más cercano esté, mejor.
  3. Elegir el nombre del paquete que se quiere instalar, en este caso Rcmdr.

En el caso de este paquete se van a instalar muchos otros subpaquetes (dependencias), que son necesarios para que puedan funcionar todas las opciones de los menús del entorno gráfico.

Una vez que hemos instalado R, aparecerá en el escritorio su icono correspondiente:



Tras abrir el programa R, comienza su ejecución, apareciendo una ventana similar a la siguiente:



Además de utilizar R introduciendo directamente el código en la ventana de comando, hay una forma cómoda de ir evaluando —y guardando, si se desea— el código de R que tengamos escrito. Basta dividir la ventana principal de R en dos partes: en una estará el código que vamos escribiendo, desde la que será enviado a la otra para que se evalúe. Para ello hay que hacer que la ventana de comandos ocupe una mitad, y la ventana de editar código la otra. La subventana para editar código aparece al entrar en el menú:

*Archivo -> Nuevo script,*

si el código no está guardado en ningún archivo, o

*Archivo -> Abrir script...*

si ya lo está.

Ahora, para evaluar una parte del código, basta seleccionarla y pulsar *Ctrl+R*, o, menos cómodo, entrar en el menú *Editar -> Correr línea o selección*. Para evaluar todo el código del archivo, se puede entrar en el menú

*Editar -> Ejecutar todo*

o seleccionarlo todo y utilizar la forma anterior.

## 1.2 AYUDA DE R

R dispone de un sistema de ayuda que se puede consultar desde el menú que aparece al iniciar el programa (opción **Help**) o invocando la ayuda en línea.

La ayuda desde el menú se estructura de la siguiente manera:

<b>Console</b>	Ayuda sobre el uso de las teclas y sus combinaciones en R
<b>R language (standard)</b>	Proporciona ayuda sobre funciones concretas
<b>R language (html)</b>	Arranca un entorno de ayuda completo en formato html
<b>Manuals</b>	Da acceso al manual de referencia de R en formato pdf
<b>Apropos</b>	Da información sobre las funciones relacionadas con una dada
<b>About</b>	Informa de la versión actual de R

Algunos ejemplos de ayuda en línea mediante comandos:

`> help()` Muestra una ventana de ayuda general sobre R.

`> help.start()` Arranca un manual de ayuda completo en formato html, utilizando el navegador del sistema.

`> help(log)` Muestra una ventana de ayuda sobre la función logaritmo.

`> apropos("plot")` Muestra las funciones u objetos que incluyen la cadena "plot" en su nombre.

`> help.search("plot")` Muestra la ayuda sobre las funciones que incluyen la cadena "plot".

## 2. USO COMO CALCULADORA CIENTÍFICA

R puede ser utilizado como una calculadora de modo interactivo.

Operadores		Significado
Aritméticos	%%	Resto de una división
	%/%	Cociente entero de la división
	*	Multiplicación
	+	Suma
	-	Resta
	^	Elevar a una potencia
	/	División
Lógicos	<	Menor que
	<=	Menor o igual que
	>	Mayor que
	>=	Mayor o igual que
	==	Igual
	!=	Distinto

Función	Significado
exp(x)	Exponencial de x ( $e^x$ )
log(x)	Logaritmo neperiano de x
log(x,n)	Logaritmo en base n de x
log10(x)	Logaritmo en base 10 de x
sqrt(x)	Raíz cuadrada de x
factorial(x)	Factorial de x (x!)
floor(x)	Mayor valor entero < x
ceiling(x)	Menor valor entero > x
trunc(x)	Valor entero más próximo a x, entre x y 0. Coincide con floor(x) para valores positivos y con ceiling(x) para valores negativos
round(x, digits=0)	Redondea el valor de x, siendo digits el número de decimales que se permiten.
signif(x, digits=6)	Devuelve el valor de x con 6 dígitos en notación científica
cos(x)	Coseno de x (en radianes)
sin(x)	Seno de x (en radianes)
tan(x)	Tangente de x (en radianes)
acos(x), asin(x), atan(x)	Funciones trigonométricas inversas
abs(x)	Valor absoluto de x

**Ejercicio 1:** Realizar algunos cálculos aritméticos en R.

### 3. VARIABLES Y VECTORES.

En R, a diferencia de otros lenguajes, no es necesario declarar a priori el nombre y el tipo de las variables. Para asignarle un valor, se usa el signo `<-`. Los nombres de las variables empiezan por una letra, que puede ir seguida de más letras, dígitos o los caracteres punto (.) o línea de subrayado (\_).

Los distintos tipos de valores que R asigna a las variables son:

Modo	Tipo
<b>logic</b>	valores lógicos: TRUE (T) o FALSE (F)
<b>numeric</b>	números reales
<b>integer</b>	valores enteros
<b>double</b>	valores reales con doble precisión
<b>complex</b>	números complejos
<b>character</b>	caracteres (letras y/o números)
<b>NA</b>	dato no disponible ( <i>not available</i> )
<b>NaN</b>	no es un número ( <i>not a number</i> )

El tipo básico de objetos de R son los vectores. Se trata de estructuras atómicas, puesto que todos sus elementos del vector deben ser del mismo tipo (numérico, complejo, lógico o de carácter). Ejemplos de creación de vectores son:

```
> c(1,2,3,4,5)
[1] 1 2 3 4 5
> c(T,F,T,T)
[1] TRUE FALSE TRUE TRUE
> x<-c("Badajoz","Cáceres")
> x
[1] "Badajoz" "Cáceres"
> c(1,2,3,"cuatro")->x
> x
[1] "1" "2" "3" "cuatro"
> c(1,2,3,F)
[1] 1 2 3 0
```

La función `c()` permite concatenar vectores. Por ejemplo:

```
> x<-c(1,3,5)
> y<-c(2,4,6)
> c(x,y)
[1] 1 3 5 2 4 6
```

Podemos extraer elementos de un vector de este modo:

```
> x<-c(1,2,3,4,5,6,7,8,9,10);x[3];x[-3];x[c(1,5,7)]
[1] 3
[1] 1 2 4 5 6 7 8 9 10
[1] 1 5 7
```

O evaluar expresiones para el vector

```
> x>8
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE
```

Para crear patrones hay varias formas. Veamos cómo funcionan:

```
> 1:5
[1] 1 2 3 4 5
> seq(1,6)
[1] 1 2 3 4 5 6
> seq(1,6,by=0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0
> seq(1,6,length=10)
[1] 1.000000 1.555556 2.111111 2.666667 3.222222 3.777778 4.333333
4.888889
[9] 5.444444 6.000000
> rep(1,5)
[1] 1 1 1 1 1
> rep(c(1,2),5)
[1] 1 2 1 2 1 2 1 2 1 2
> rep(1:4,2)
[1] 1 2 3 4 1 2 3 4
> rep(1:3,c(1,4,5))
[1] 1 2 2 2 2 3 3 3 3 3
```

**Ejercicio 2:** Crear sendos vectores con los 50 primeros números pares e impares, a los que llamaréis vp y vi, respectivamente. Concatenar ambos vectores y seleccionar aquellos valores que ocupan posiciones que sean múltiplos de 3.

**Ejercicio 3:** Crear un vector con los números de vuestro D.N.I., donde cada uno de los números que lo componen debe aparecer repetido un número de veces igual a él mismo.

#### 4. MATRICES, ARRAYS Y DATA.FRAMES

Una matriz en R es un conjunto de objetos indizados por filas y columnas. Un array en R es lo mismo, salvo que puede tener más de dos dimensiones.

La sintaxis general de la orden para crear una matriz es:

```
matrix(data, nrow, ncol, byrow=F)
```

---

data	datos que forman la matriz
nrow	número de filas de la matriz
ncol	número de columnas de la matriz
byrow	Los datos se colocan por filas o por columnas según se van leyendo. Por defecto se colocan por columnas.

---

Podemos crear matrices del siguiente modo:

```
> matrix(1:12)

> m1<-matrix(1:12, nrow=4)

> m2<-matrix(1:12, ncol=4)

> matrix(c(1,2,3,4,5,6,7,8,9,10,11,12),ncol=4,nrow=3)

> matrix(c(1,2,3,4,5,6,7,8,9,10,11,12),ncol=4,nrow=3,byrow=T)
```

Los datos que contiene una matriz, igual que ocurriría para los vectores, deben ser todos del mismo tipo (numérico, carácter o lógico).

<i>Funciones con matrices</i>	
<code>dim()</code>	Devuelve las dimensiones de una matriz
<code>dimnames()</code>	Devuelve el nombre de las dimensiones de una matriz
<code>colnames()</code>	Devuelve el nombre de las columnas de una matriz
<code>rownames()</code>	Devuelve el nombre de las filas de una matriz
<code>mode()</code>	Devuelve el tipo de datos de los elementos de una matriz
<code>length()</code>	Devuelve el número total de elementos de una matriz
<code>is.matrix()</code>	Devuelve T si el objeto es una matriz, F si no lo es
<code>[ , ]</code>	Accede a elementos dentro de la matriz
<code>apply()</code>	Aplica una función sobre las filas o columnas de una matriz
<code>cbind()</code>	Añade una columna a una matriz dada o crea una matriz leyendo los elementos por columnas
<code>rbind()</code>	Añade una fila a una matriz dada o crea una matriz leyendo los elementos por filas
<code>t()</code>	Traspone la matriz dada

Un ejemplo del uso de estos comandos es el siguiente:

```
> x <- cbind(x1 = 3, x2 = c(4:1, 2:5))
```

```
> x
```

```
      x1 x2
[1,]  3  4
[2,]  3  3
[3,]  3  2
[4,]  3  1
[5,]  3  2
[6,]  3  3
[7,]  3  4
[8,]  3  5
```

```
> y<-cbind(x,c(1:8))
```

```
> y
```

```
      x1 x2
[1,]  3  4  1
[2,]  3  3  2
[3,]  3  2  3
[4,]  3  1  4
[5,]  3  2  5
[6,]  3  3  6
[7,]  3  4  7
[8,]  3  5  8
```

```
> apply(x, 2, sort)
```

```
      x1 x2
[1,]  3  1
[2,]  3  2
[3,]  3  2
[4,]  3  3
[5,]  3  3
[6,]  3  4
[7,]  3  4
[8,]  3  5
```

```

> dimnames(x)[[1]] <- letters[1:8]
> x
  x1 x2
a  3  4
b  3  3
c  3  2
d  3  1
e  3  2
f  3  3
g  3  4
h  3  5
> col.sums <- apply(x, 2, sum)
> row.sums <- apply(x, 1, sum)
> rbind(cbind(x, Rtot = row.sums), Ctot = c(col.sums, sum(col.sums)))
  x1 x2 Rtot
a   3  4   7
b   3  3   6
c   3  2   5
d   3  1   4
e   3  2   5
f   3  3   6
g   3  4   7
h   3  5   8
Ctot 24 24  48

```

El producto matricial se realiza mediante el operador “%\*%”, mientras que “\*” realiza el producto componente a componente.

```

> m1%*%m2
      [,1] [,2] [,3] [,4]
[1,]   38   83  128  173
[2,]   44   98  152  206
[3,]   50  113  176  239
[4,]   56  128  200  272

```

Los comandos para manejar arrays son similares a los que manejan matrices. Su definición general es de la forma

```
array(data, dim=length(data))
```

donde `data` son los datos que forman el array y `dim` es un vector con las dimensiones del array.

```

> array(1:3, c(2, 4))
      [,1] [,2] [,3] [,4]
[1,]    1    3    2    1
[2,]    2    1    3    2

```

**Ejercicio 5:** Crear una matriz con cuatro alumnos (de este curso) y su edad, año de nacimiento y número de teléfono. Deberá aparecer el nombre de la columna (“edad”, “año de nacimiento”, “teléfono”) y el nombre de la fila, que será el nombre del alumno al que corresponden los datos.

Los `data.frames` u hojas de datos son listas que generalizan a las matrices, en el sentido de que las columnas (variables a menudo) pueden ser de diferente tipo entre sí (no todas numéricas, por ejemplo). Sin embargo, todos los elementos de una misma columna deben ser del mismo tipo. Al igual que las filas y columnas de una matriz, todos los elementos de un `data.frame` deben tener la misma longitud. De este modo, pueden usarse funciones tales como `dimnames`, `dim`, `nrow` sobre un `data.frame` como si se tratara de una matriz y se pueden imprimir en forma matricial.

Para crear un objeto de este tipo, usaremos la función



```
data.frame( )
```

La manera más sencilla de construirlo es utilizar la función `read.table( )` para leer la hoja de datos desde un archivo, por ejemplo de texto o excel. Detallaremos el uso de este comando a continuación.

## 5. ORGANIZACIÓN DEL TRABAJO

Los datos suelen leerse desde archivos externos y no teclearse de modo interactivo. Las capacidades de lectura de archivos de R son sencillas y sus requisitos son bastante estrictos, pues se presupone que el usuario es capaz de modificar los archivos de datos con otras herramientas, como editores de texto, para ajustarlos a las necesidades de R. Generalmente esta tarea es bastante simple.

Si queremos almacenar los datos en hojas de datos, podemos leerlos directamente con la función `scan( )`. La orden

```
>scan("datos")
```

lee los datos de un fichero de nombre "datos".

También podemos utilizar esta orden para introducir datos desde la propia ventana de comandos. Vamos introduciendo datos, hasta que queramos finalizar, momento en que pulsamos ENTER. Por ejemplo, si queremos introducir los datos 1, 2 y 3, haremos:

```
>scan()  
1: 1  
2: 2  
3: 3  
4:  
Read 3 items  
[1] 1 2 3
```

La función `read.table` lee datos desde un archivo, donde aparecen en forma de tabla, y crea un `data.frame` a partir de ellos.

Su sintaxis es la siguiente:

```
read.table("file", head=T)
```

Por defecto el separador entre campos es el espaciado, el carácter que denota los decimales es ".". `head=T` indica que la primera línea de los datos contiene los nombres de las variables.

Si escribimos

```
>read.table("datos.dat", sep=",")
```

estaríamos leyendo los datos del archivo *datos.dat*, teniendo en cuenta que están separados por comas y que no hay cabecera.

En la distribución de R se incluyen algunos objetos con datos, tanto en la biblioteca base como en el resto. Usamos la orden

```
>library(car)
```

para cargar el paquete o biblioteca `car`.

Para obtener una lista de todos los datos disponibles de las bibliotecas conectadas en un determinado momento, basta con teclear

```
>data()
```

y para cargar unos datos concretos

```
>data(nombre)
```

La orden

```
>ls()
```

proporciona una lista de todos los objetos creados en R durante la sesión de trabajo.

Para escribir los datos `x` en el archivo `file`, podemos usar la función `write`.

```
write(x, file="data", sep="", dec=".")
```

La orden `write.table` escribe el elemento `x` en el archivo `file` tras haberlo convertido en un `data.frame` si no lo era.

```
write.table(x, file="", sep=" ", dec=".")
```

## **6. REPRESENTACIÓN GRÁFICA DE FUNCIONES**

Para representar gráficas de funciones usamos la orden `plot()`. Por ejemplo,

```
> plot(sin, -pi, 2*pi)
```

Otro uso de esta función es producir un diagrama de puntos de `y` frente a `x`. Por ejemplo,

```
> x<-seq(-3,3,0.01)
```

```
> plot(x,dnorm(x),type="p",col=2,main="Densidad N(0,1)")
```

La orden `type` sirve para cambiar la representación de los puntos, que por defecto se unen mediante líneas ("`p`" corresponde a puntos, "`l`" a líneas y "`b`" a puntos unidos por líneas). El color de la gráfica se puede definir mediante `col` y la orden `main` permite añadir un título a la gráfica.

Además, las funciones `lines()` y `points()` permiten superponer líneas a un gráfico ya existente. La primera pinta líneas y la segunda puntos.

```
> lines(x,dnorm(x,1),col=3)
```

## 7. ESTRUCTURAS DE PROGRAMACIÓN

R permite escribir comandos en la misma línea, utilizando el carácter “;”. Los comandos pueden agruparse entre llaves:

```
{comando1; comando2; comando3; ...}
```

### 7.1 CONDICIONALES

Para la ejecución condicional de sentencias, la orden que usaremos es `if`

```
if (expr_1) expr_2 else expr_3
```

donde `expr_1` debe producir un valor lógico, de modo, que, si éste es verdadero (T), se ejecuta `expr_2` y, si es falso (F), se ejecuta `expr_3`.

Por ejemplo, vamos a crear dos listas; una para guardar los números pares de 1 a 10, y otra para los impares:

```
> n <- 10
> pares <- c()
> impares <- c()
> for(i in 1:n){ if(i%%2==0) pares<-c(pares,i) else impares<-
c(impares,i)}
> pares
[1] 2 4 6 8 10
> impares
[1] 1 3 5 7 9
```

### 7.2 CICLOS

Para crear un bucle repetitivo `for`, la sintaxis es la siguiente:

```
for (nombre in expr_1) {expr_2}
```

donde `nombre` es la variable de control de la iteración, `expr_1` es un vector (usualmente de la forma `m:n`) y `expr_2` es la expresión a ejecutar, que se evalúa repetidamente.

Por ejemplo,

```
> for(i in 1:10) {print(i)}
```

En cuanto a `while`, la sintaxis es como sigue:

```
while (condicion) {expr}
```

que ejecuta `expr` mientras que se cumpla `condición`.

Por ejemplo,

```
> n<-1
> while (n<=10) {print(n);n=n+1}
```

**Ejercicio 6:** Encontrar cuál es el mayor número entero cuyo cuadrado no excede de 1000, utilizando la orden `while`.

## 8. ESCRITURA DE NUEVAS FUNCIONES

R permite crear nuevas funciones, que se pueden utilizar a su vez en expresiones posteriores. La forma general de hacerlo es la siguiente:

```
nombre <- function (arg_1 , arg_2, ..... ) expr
```

donde `nombre` es el nombre que le damos a la función, `arg_1`, `arg_2`, ... contienen los parámetros de la misma, y `comandos` es una expresión que utiliza los argumentos de la función para calcular un valor.

Las variables definidas dentro del cuerpo de una función son locales, y desaparecen al terminar la ejecución de la función.

Si queremos fijar alguno de los argumentos, simplemente le asignamos el valor usando el signo “=”

```
nombre <- function(arg_1, arg_2=val, ...) expr
```

Por ejemplo, supongamos que queremos definir una función que calcule el valor medio de dos números:

```
> media2<-function(x,y){return((x+y)/2)}  
  
> media2(10,20)
```

La función `return` le dice a la función creada qué valor debe devolver. No es necesario que toda función contenga necesariamente la instrucción `return`; hay muchas funciones que efectúan distintas tareas sin necesidad de devolver un resultado.

**Ejercicio 7:** Escribir una función que intercambie los valores de dos variables, denominadas `x` e `y`.

**Ejercicio 8:** Escribir una función que genere los  $n$  primeros términos de la serie de Fibonacci.

**Ejercicio 9:** Escribir una función que escriba “Hola” si el valor de la variable es menor o igual que 0 y escriba “Adiós” si es mayor. Evaluar la función para los valores 5 y -5.

## 9. INICIO Y MANEJO DE R COMMANDER.

Una vez que R se está ejecutando, simplemente cargando el paquete Rcmdr mediante la instrucción

```
>library(Rcmdr)
```

en la consola de R, o bien seleccionando en el menú Paquetes->Cargar Paquete, se inicia la interfaz gráfica de usuario de R Commander:



Las ventanas de R Commander y R Console flotan libremente en el escritorio. Siempre que sea posible, es más sencillo usar los menús y cuadros de diálogo de R Commander

### Elementos de R Commander

La interfaz R Commander incluye los siguientes elementos:

#### Menú

Aparece en la parte superior de la ventana, con las siguientes opciones:

<b>Fichero</b>	Opciones de menú para cargar y guardar archivos de instrucciones, de resultados y/o el área de trabajo de R; cambiar directorio de trabajo y salir.
<b>Editar</b>	Opciones para editar los contenidos de las ventanas (Copiar, Cortar, Pegar, Borrar, Buscar, Seleccionar, Deshacer y Rehacer) y limpiar la ventana en la cual esté situado el cursor.
<b>Datos</b>	Opciones para leer y manipular datos.
<b>Estadísticos</b>	Submenús que contienen opciones de menú para una variedad de análisis estadísticos básicos.
<b>Gráficos</b>	Opciones de menú para crear gráficos estadísticos simples.

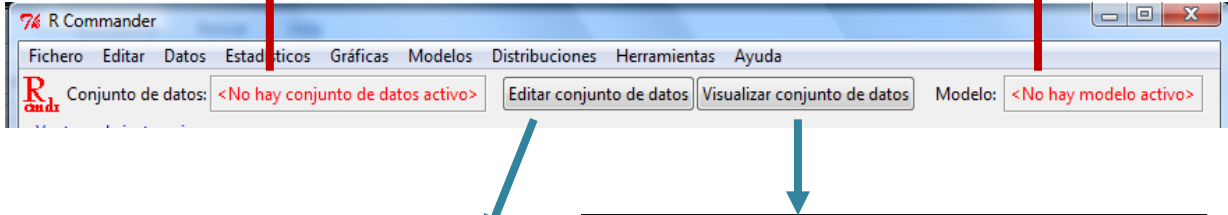
<b>Modelos</b>	Opciones de menú para obtener resúmenes, intervalos de confianza, test de hipótesis, diagnósticos y gráficos para un modelo estadístico, así como para añadir cantidades diagnósticas, como residuos, a la serie de datos.
<b>Distribuciones</b>	Probabilidades, cuantiles y gráficos, para distribuciones estadísticas estándares.
<b>Herramientas</b>	Opciones para cargar paquetes R, y para establecer algunas opciones.
<b>Ayuda</b>	Opciones de menú para obtener información sobre R Commander.

Cuando una opción de menú aparece en gris, significa que está inactiva y no puede aplicarse en el contexto actual.

Bajo los menús aparece una **barra de herramientas** con una fila de botones:

*Muestra el nombre de la serie de datos activa. Inicialmente no hay ninguna. Presionándolo podemos elegir entre las series de datos que estén en memoria*

*Muestra el nombre del modelo estadístico activo (un modelo lineal, lineal generalizado, logit multinomial,...). Inicialmente no hay ninguno activo. Pulsando sobre el botón podemos elegir entre los modelos en memoria.*



*Abre el editor de datos para modificar la serie de datos activa*

*Permite visualizar la serie de datos activa. Puede permanecer abierto mientras se realizan otras operaciones.*

**Ventana de instrucciones:** En ella aparecen las instrucciones de R generadas por R Commander. También es posible escribir directamente o modificar dichas instrucciones, como lo haríamos en R, tras el símbolo de sistema `>`. Sin embargo, el propósito de utilizar R Commander es evitar esto. Presionando el botón Ejecutar, que está a la derecha bajo la ventana, la línea que contiene el cursor o bien el bloque de instrucciones seleccionadas se ejecuta para su resolución.

**Ventana de resultados:** En ella aparecen los resultados que se van generando al ejecutar las instrucciones. Las instrucciones escritas en esta ventana aparecen en rojo y los resultados en azul igual que ocurre en R Console.

**Mensajes:** Aparece sombreada en la parte inferior y muestra mensajes de error, advertencias y otro tipo de información de utilidad.

**Gráficos:** Cuando creamos gráficos, éstos aparecen en una ventana separada.