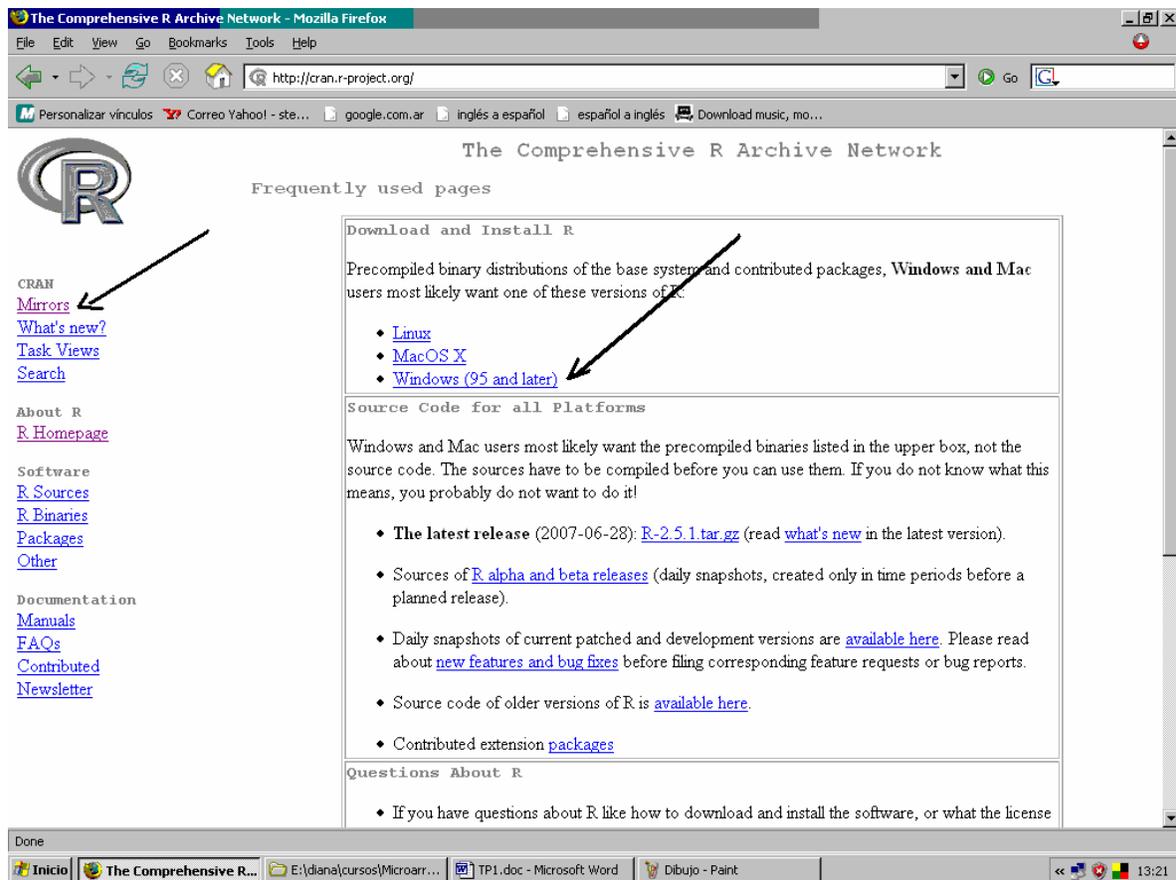


Trabajo Práctico 1: Introducción al lenguaje R

A partir de la sección 1.2 esta guía debe leerse ejecutando simultáneamente en R todos los comandos indicados.

1.0 Instalación del R

La página web del proyecto R es <http://cran.r-project.org/>. Este es el sitio principal para bajar el R y obtener documentación FAQs (FAQ son las iniciales de Frequently Asked Questions, o sea preguntas más frecuentes).



La versión actual de R es 2.5.1. Pero utilizaremos la 2.5.0

The screenshot shows the Mozilla Firefox browser window displaying the CRAN Mirrors page. The address bar shows <http://cran.r-project.org/>. The page title is "CRAN Mirrors". The main content area lists various mirrors for different countries, including Argentina, Australia, Austria, Brazil, Canada, Chile, Croatia, and Czech Republic, each with a URL and the name of the hosting institution. A sidebar on the left contains navigation links such as "CRAN Mirrors", "What's new?", "Task Views", "Search", "About R", "R Homepage", "Software", "R Sources", "R Binaries", "Packages", "Other", "Documentation", "Manuals", "FAQs", "Contributed", and "Newsletter". The browser's taskbar at the bottom shows several open applications, including "The Comprehensive R...", "Microarrays", "TP1.doc - Microsoft Word", and "fig1.bmp - Paint".

The screenshot shows the Mozilla Firefox browser window displaying the "R for Windows" page. The address bar shows <http://mirror.crioyt.edu.ar/>. The page title is "R for Windows". The main content area provides information about the Windows binaries, including a note that CRAN does not have Windows systems and cannot check these binaries for viruses. It lists subdirectories: "base" (Binaries for base distribution) and "contrib" (Binaries of contributed packages). A black arrow points from the "base" link in the subdirectories list to the "base" link in the text below. The sidebar on the left contains navigation links similar to the previous screenshot. The browser's taskbar at the bottom shows several open applications, including "The Comprehensive R...", "Microarrays", "TP1.doc - Microsoft Word", and "fig1.bmp - Paint".

File Edit View Go Bookmarks Tools Help

http://mirror.criyt.edu.ar/r/

Personalizar vínculos Correo Yahoo! - ste... google.com.ar inglés a español español a inglés Download music, mo...

R-2.5.1 for Windows

This directory contains a Windows binary distribution of R-2.5.1.

Patches to this release are incorporated in the [r-patched snapshot build](#).

A build of the development version (which will eventually become the next major release of R) is available in the [r-d build](#).

In this directory:

- [README R-2.5.1](#) Installation and other instructions.
- [CHANGES](#) New features of this Windows version.
- [NEWS](#) New features of all versions.
- [R-2.5.1-win32.exe](#) Setup program (about 29 megabytes). Please download this from a [mirror near you](#).
- [old](#) Previous releases.

CRAN
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

About R
[R Homepage](#)

Software
[R Sources](#)
[R Binaries](#)

The Comprehensive R Archive Network - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://mirror.criyt.edu.ar/r/

Personalizar vínculos Correo Yahoo! - ste... google.com.ar inglés a español español a inglés Download music, mo...

Previous Releases of R for Windows

This directory contains previous binary releases of R to run on Windows 95, 98, ME, NT4.0, 2000 and XP on Intel/ci

The current release, and links to development snapshots, are available [here](#). Source code for these releases and others i through [the main CRAN page](#).

In this directory:

- [R 2.5.0 \(April, 2007\)](#)
- [R 2.4.1 \(December, 2006\)](#)

CRAN
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

The Comprehensive R Archive Network - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://mirror.criyt.edu.ar/r/

Personalizar vínculos Correo Yahoo! - ste... google.com.ar inglés a español español a inglés Download music, mo...

R-2.5.0 for Windows

This directory contains a Windows binary distribution of R-2.5.0.

In this directory:

- [README R-2.5.0](#) Installation and other instructions.
- [CHANGES](#) New features of this Windows version.
- [NEWS](#) New features of all versions.
- [R-2.5.0-win32.exe](#) Setup program (about 29 megabytes). Please download this from [mirror near you](#).
- [md5sum.txt](#) md5sum output for the setup program. A Windows GUI version of md5sum is available at <http://www.md5summer.org/>; a Windows command line version is available at <http://www.etree.org/md5com.html>.

CRAN
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

About R
[R Homepage](#)

1.1 Un poco de historia

R es un conjunto de programas integrados para manejo de datos, simulaciones, cálculos y realización de gráficos. Es además un lenguaje de programación orientado a objetos.

R es una implementación libre, independiente, open-source del lenguaje de programación S que actualmente es un producto comercial llamado S-PLUS y es distribuido por Insightful Corporation. El lenguaje S, que fue desarrollado a mediados de los años 70 en Bell Labs (de AT&T y actualmente Lucent Technologies). Originalmente un programa para el sistema operativo Unix, R ahora puede obtenerse también en versiones para Windows y Macintosh y Linux. A pesar de que hay diferencias menores entre R y S-PLUS (la mayoría en la interfase gráfica), son esencialmente idénticos.

El proyecto R fue iniciado por Robert Gentleman y Ross Ihaka (de donde se deriva “R”) del Statistics Department, University of Auckland, en 1995.

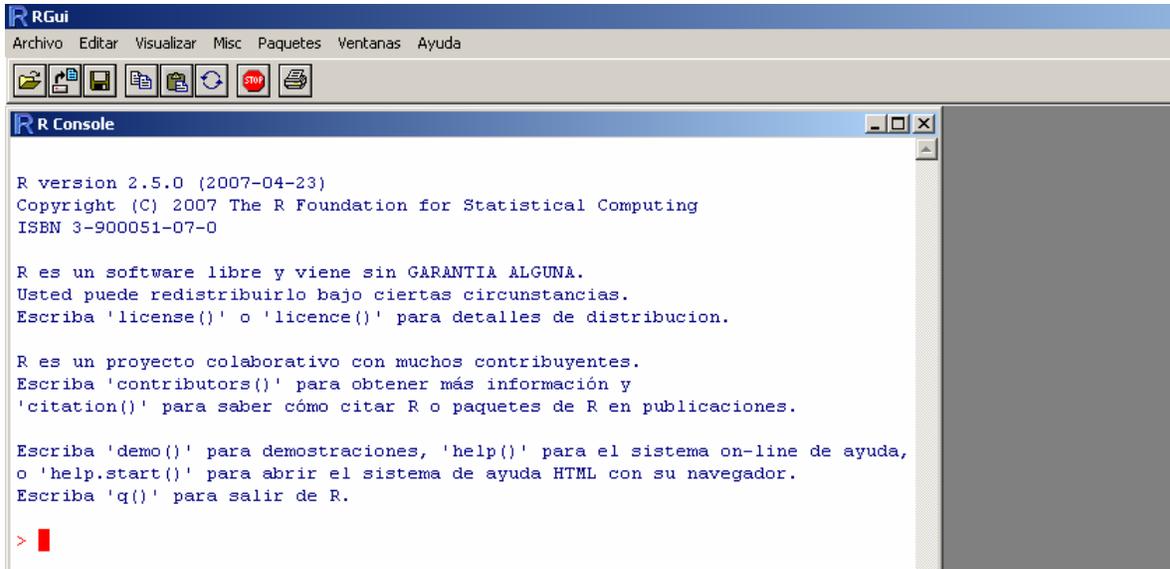
Actualmente R es mantenido por un grupo internacional de desarrolladores *voluntarios*: Core development team.

Para analizar los datos de microarreglos utilizaremos el software R y programas específicos desarrollados por BioConductor. Este también está integrado por desarrolladores voluntarios de software para el análisis y comprensión de datos de genómica, especialmente microarreglos, y se inició a mediados de 2001 (<http://www.bioconductor.org/whatisit>)

Retomaremos el tema de BioConductor en la próxima sección

1.2 Iniciando y cerrando R

La forma más fácil de usar R es en forma interactiva mediante la línea de comandos. Una vez instalado hay que hacer un doble click en el ícono de R (en Unix/Linux, se escribe R desde el símbolo de comandos (command prompt)). Cuando R se inicia, aparece la ventana del programa “Gui” (graphical user interface) con un mensaje de apertura

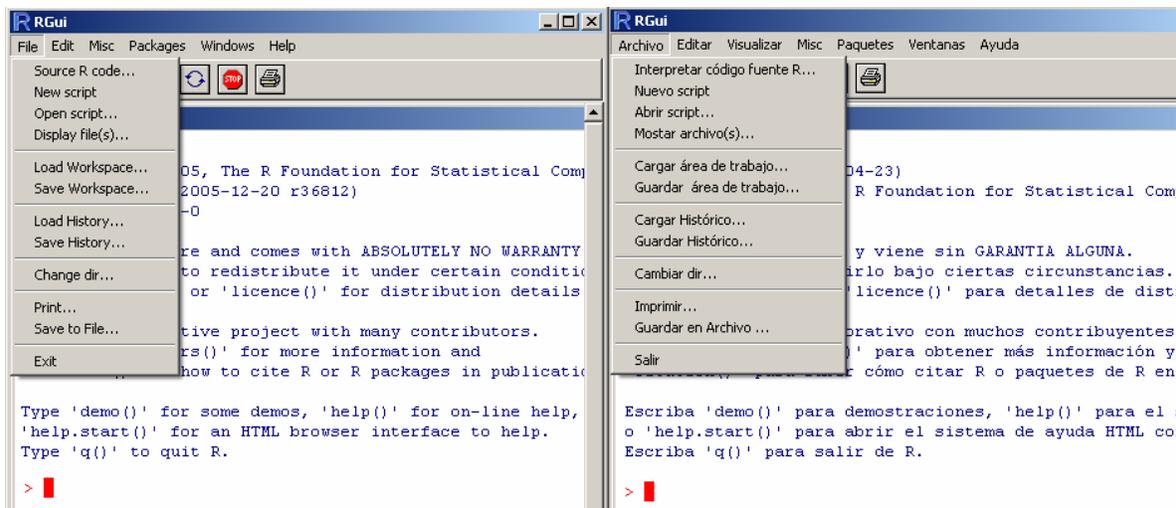


Debajo del mensaje de apertura en la Consola de R se encuentra el “prompt” que es el símbolo > (“mayor”).

>

La mayoría de las expresiones en R se escriben directamente a continuación del “prompt” en la Consola de R.

Si se escribe e intenta ejecutar un comando que se ha escrito en forma incompleta, el programa presenta un “+” que es el prompt de continuación.



También es posible generar un archivo “script” con los comandos que se quieren ejecutar. Veremos esto luego.

Comencemos agregando a R los paquetes específicos de microarreglos de la página de BioConductor: <http://www.bioconductor.org>

BioC Release 2.0

Bioconductor 2.0 was released 26 April, 2007. This release is designed for R 2.5.0. View the packages here

En <http://www.bioconductor.org/docs/install-howto.html> vemos

Right now, the easiest way to install Bioconductor packages is using the `biocLite.R` installation script. Here's how to use it:

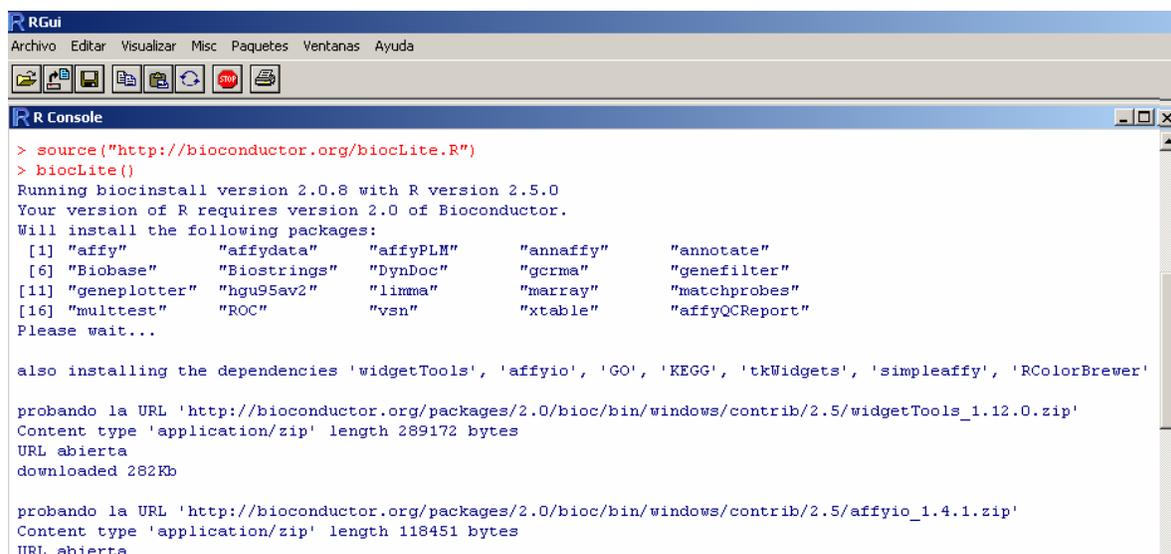
In an R command window, type the following:

```
source("http://bioconductor.org/biocLite.R")
biocLite()
```

Es decir que la forma más sencilla de instalar los paquetes es utilizando el script de instalación `biocLite.R` (en la próxima sección veremos que es un script, aquí simplemente lo utilizamos). Así es como se usa

En el prompt de se escribe

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite()
```



```
RGui
Archivo Editar Visualizar Misc Paquetes Ventanas Ayuda

R Console
> source("http://bioconductor.org/biocLite.R")
> biocLite()
Running biocinstall version 2.0.8 with R version 2.5.0
Your version of R requires version 2.0 of Bioconductor.
Will install the following packages:
 [1] "affy"           "affydata"      "affyPLM"       "annaffy"       "annotate"
 [6] "Biobase"       "Biostrings"    "DynDoc"        "gcrma"         "genefilter"
[11] "genefilter"    "hgu95av2"      "limma"         "marray"        "matchprobes"
[16] "multtest"      "ROC"           "vsn"           "xtable"        "affyQCReport"
Please wait...

also installing the dependencies 'widgetTools', 'affyio', 'GO', 'KEGG', 'tkWidgets', 'simpleaffy', 'RColorBrewer'

probando la URL 'http://bioconductor.org/packages/2.0/bioc/bin/windows/contrib/2.5/widgetTools_1.12.0.zip'
Content type 'application/zip' length 289172 bytes
URL abierta
downloaded 282Kb

probando la URL 'http://bioconductor.org/packages/2.0/bioc/bin/windows/contrib/2.5/affyio_1.4.1.zip'
Content type 'application/zip' length 118451 bytes
URL abierta
```

Tarda ...

Guarda los paquetes en la subcarpeta “library” donde se encuentra instalado R:
C:\Archivos de programa\R\R-2.5.0\library

Ya tenemos instalados los programas bajados de Bioconductor

Para salir de R, se escribe `q()` o se utiliza la opción Exit que se encuentra abajo en el menú File.

1.3 Tipos de objetos en R

Los objetos primitivos o atómicos de R son:

- variables:
- numéricas (integer, double, complex)
 - de caracteres
 - lógicas
- funciones

Presentaremos primero ejemplos de los distintos tipos de variables y veremos más adelante cómo se pueden construir objetos más complejos como vectores, matrices y listas. Presentaremos funciones incorporadas en el R y finalmente veremos cómo escribir las propias funciones.

1.3.1 Variables

```
> a <- 49 # <- operador de asignación
# no se pueden poner espacios entre el “menor” y el “menos”
# se asigna el valor 49 a la variable a

> a <- 49 # variable numérica
> class(a)
[1] "numeric"
> a
[1] 49
> sqrt(a)
[1] 7

> a <- "El perro se comió mi informe"
> a
[1] "El perro se comió mi informe"
> class(a) # variable caracter
[1] "character"
```

```
> b <- sub("perro","gato",a) # sustituye una cadena de caracteres
> b
[1] "El gato se comió mi informe"
```

En el siguiente ejemplo, se asigna a la variable **a** el resultado (verdadero o falso) de una comparación.

```
> a <- (1+1==3) # operador de comparación " == "
> class(a)
[1] "logical"
> a
[1] FALSE
```

1.3.2 Vectores

Un vector es una colección ordenada de datos del mismo tipo. Utilizamos la función `c()` y el operador `:`, para generar vectores. Volveremos a ver el operador `:` más adelante.

```
> c(TRUE, FALSE) # función c()
[1] TRUE FALSE

> aaa <- c(TRUE, FALSE) # asignación <-
> aaa
[1] TRUE FALSE
> class(aaa)
[1] "logical"

> bbb <- c(2.1,3.8) # vector numérico
> bbb
[1] 2.1 3.8
> class(bbb)
[1] "numeric"
```

Otra forma de generar vectores lógicos

```
> bbb > 3 # vector lógico
[1] FALSE TRUE

> bbb1 <- bbb > 3
> class(bbb1)
[1] "logical"

> bbb1
[1] FALSE TRUE
> class(bbb1)
[1] "logical"

> ccc <- c("a","b","c") # vector de caracteres
> ccc
[1] "a" "b" "c"
```

```
> class(ccc)
[1] "character"

> ddd <- 1:3           # vector de enteros
> ddd
[1] 1 2 3
> class(ddd)
[1] "integer"
```

Ejemplo: La intensidad estimada de 15488 spots de un microarray se almacena en un vector de 15488 números enteros.

Hemos visto un comando útil en R para entrar una cantidad pequeña de datos es la función `c()`. Esta función *combina, concatena* elementos del mismo tipo. Veamos otro ejemplo.

Supongamos que los valores siguientes representan ocho tiradas de un dado equilibrado:

```
2 5 1 6 5 5 4 1
```

Para ingresar estos valores en una sesión de R, *asignándolos* al objeto `queridostodos` escribimos

```
> queridostodos <- c(2,5,1,6,5,5,4,1)
```

y si escribimos

```
> queridostodos
```

y oprimimos enter, aparecerá lo siguiente

```
[1] 2 5 1 6 5 5 4 1
>
```

Observaciones:

- Hemos asignado valores a una variable llamada `queridostodos`. R distingue mayúsculas y minúsculas de manera que podría haber otra variable distinta llamada `QueridosTodos`. El nombre de una variable, tiene que comenzar con una letra y puede contener la mayoría de las combinaciones de letras, números y puntos. No puede nombrarse con números únicamente.
- El operador de asignación “<-”; está compuesto de un < (“menor”) y un - (“menos” o “guión”) escritos juntos. Se lee como “*toma*”, la variable `queridostodos` *toma* el valor `c(2,5,1,6,5,5,4,1)`. Alternativamente, a partir de la versión 1.4.0 de R también puede utilizarse “=” como el operador de asignación.

- El valor de `queridostodos` no aparece en pantalla automáticamente. Sí lo hace cuando se escribe el nombre en la línea de comandos, como vimos más arriba.
- El valor de `queridostodos` está precedido con un `[1]`. Esto indica que el valor es un vector.
- Al concatenar valores numéricos con caracteres, éstos deben estar entre comillas y el vector resultante es un vector de caracteres.

```
> nuevovector <- c(1,2,3, Pedro, Juan)
Error: objeto "Pedro" no encontrado
> nuevovector <- c(1,2,3, "Pedro", "Juan")
> nuevovector
[1] "1"      "2"      "3"      "Pedro" "Juan"
> class(nuevovector)
[1] "character"
```

1.3.3 Matrices

Consideraremos la función `matrix()`. Miremos el help.

```
> ? matrix
```

Del help obtenemos:

```
Matrices
Description:
'matrix' creates a matrix from the given set of values.
...
Usage:
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE)
Arguments:
data: the data vector
nrow: the desired number of rows
ncol: the desired number of columns
byrow: logical. If 'FALSE' the matrix is filled by columns,
otherwise the matrix is filled by rows.
...
```

Vemos que esta función toma vectores y los transforma en objetos con estructura de matriz. Son 4 los argumentos de esta función. Especifican los datos de entrada, el tamaño del objeto matriz creado y al argumento `byrow` que toma los valores `TRUE` o `T` (verdadero) y `FALSE` o `F` (falso) para especificar cómo son llenados los valores en la matriz.

Habitualmente los argumentos de las funciones tienen valores por *defecto* (*default values*), y vemos que esto ocurre en los argumentos de la función `matrix()` de manera que llamar a la función `matrix()` sin argumentos da como resultado una matriz que tiene una fila y una columna con un único valor `NA` (missing o “Not Available”).

```
> matrix()
```

```
      [,1]  
[1,]  NA  
>
```

El siguiente ejemplo es más interesante:

```
> a <- c(1,2,3,4,5,6,7,8)  
> A <- matrix(a,nrow=2,ncol=4, byrow=FALSE) # a is diferente de A  
> A  
[,1] [,2] [,3] [,4]  
[1,] 1 3 5 7  
[2,] 2 4 6 8  
>
```

Observe que podríamos haber eliminado `byrow=FALSE` argument, ya que éste es el valor por defecto. Es más, como los argumentos tienen un orden específico podríamos haber escrito

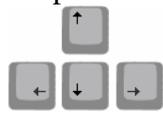
```
> A <- matrix(a,2,4)  
ó  
> A <- matrix(a,2)
```

para obtener el mismo resultado. Pero para evitar confusiones, es conveniente incluir el nombre del argumento en la llamada a una función.

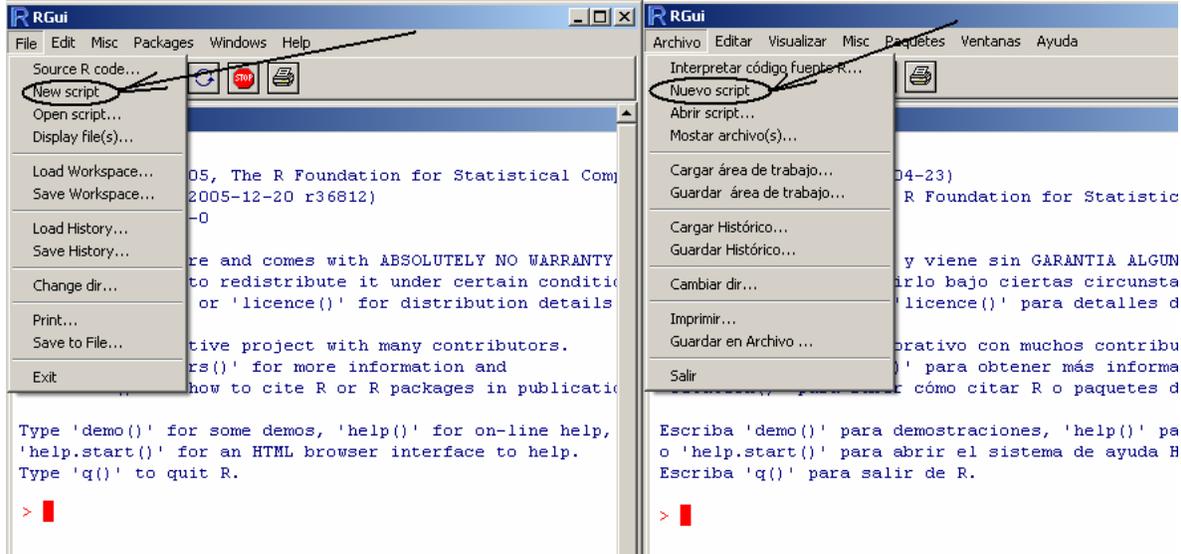
1.4 Facilidades del entorno de R

1.4.1 History

Cada comando que se ejecuta es guardado como Historia (History) y se puede ahorrar mucho tiempo en la escritura de los comandos en R si se aprende a utilizar el teclado de

 flechas,  adecuadamente. La flecha hacia arriba (↑) navega hacia atrás sobre esta historia de comandos y la flecha hacia abajo (↓) lo hace hacia delante. Las flechas hacia la izquierda (←) y hacia la derecha (→) mueven el cursor en esos sentidos dentro de la línea de comandos. Con teclas, el mouse y las opciones de copiar y pegar es fácil ejecutar comandos previos.

1.4.2 Script



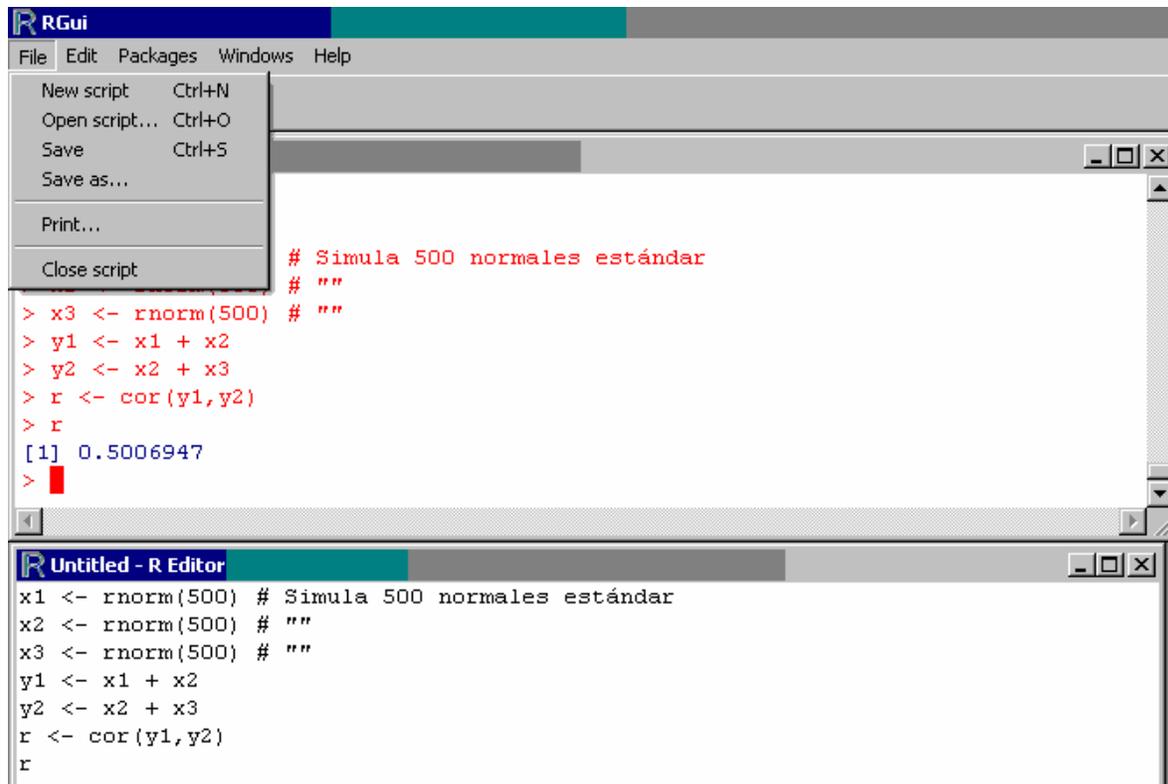
Cuando interesan ejecutar muchos comandos es más útil tener abierta junto con la ventana de comandos de R una ventana del editor que permite generar un archivo **script**. En dicha ventana se puede escribir como en un editor de textos y además ejecutar las instrucciones

Para abrir una ventana con un script nuevo, simplemente se clickea en el menú principal:

File -> New script

Para ejecutar un comando en la ventana del script hay que seleccionar el o los comandos que interesan y clickear en el ícono de ejecución del menú del editor.

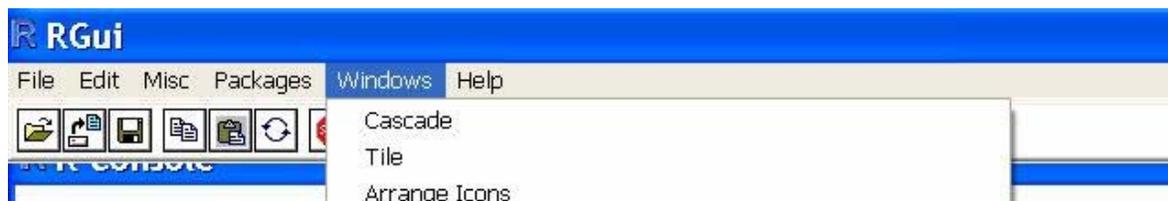
Ejemplo: Se simulan tres muestras normales standard de tamaño 500 asignándolos a las variables x_1 , x_2 , x_3 . Se suman los valores de la primera y la segunda muestra asignándolos a la variable y_1 , también se suman los de la segunda y la tercera variable asignándolos a la variable y_2 . Luego se calcula la correlación entre y_1 e y_2 .



ícono para correr los comandos seleccionados



Es útil acomodar las dos ventanas, la de comandos y la del editor, para poder verlas simultáneamente.



Esto puede realizarse cambiando la forma de las ventanas “a mano” o mediante la opción: **Windows -> Tile** del menú principal (ver figura anterior).

Cuando se ejecuta un comando desde el **script** los resultados se muestran en la ventana de la consola inicial del R.

Con la ventana **script** activa es posible guardarlo desde el correspondiente menú

File -> Save

o

File -> Save as...

Si por ejemplo, lo guardamos como **corsim.R** (no pone la extensión por defecto) en el directorio de trabajo, este archivo **script** se puede volver a utilizar abriéndolo desde el menú principal:

File -> Open script
Archivo -> Abrir script

o se lo puede correr, sin abrirlo, escribiendo en la línea de comandos:

```
> source("E:\\diana\\cursos\\Microarrays 2007\\Cordoba  
2007\\Prácticas\\corsim.R")  
> r  
[1] 0.4904142
```

Observación: el resultado no es el mismo que antes, a que se puede deber eso?

Nota: para representar directorios y subdirectorios se utiliza la barra de dividir (/), y no (\) como habitualmente se escribe en los sistemas operativos DOS y Windows.

1.4.3 El espacio de trabajo – Workspace

Todas las variables u “objetos” creados en R están guardados en lo que se llama el espacio de trabajo o *workspace*. Para ver qué variables están en el espacio de trabajo pueden usarse cualquiera de las funciones: **ls()** u **objects()** (ests funciones no necesitas argumentos entre los paréntesis). Actualmente tenemos **queridostodos** más las generadas al correr el script:

```
> ls()  
[1] "a"           "A"           "aaa"         "b"  
[5] "bbb"         "bbb1"        "ccc"         "ddd"  
[9] "nuevovector" "queridostodos" "r"           "x1"  
[13] "x2"          "x3"          "y1"          "y2"  
> objects()  
[1] "a"           "A"           "aaa"         "b"  
[5] "bbb"         "bbb1"        "ccc"         "ddd"  
[9] "nuevovector" "queridostodos" "r"           "x1"  
[13] "x2"          "x3"          "y1"          "y2"
```

Definamos una nueva variable, una función simple de la variable **queridostodos**. Ésta será agregada al espacio de trabajo:

```
> nuevaqueridostodos<- queridostodos/2 # divide cada elemento por 2
> nuevaqueridostodos
[1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5
> ls()
[1] "nuevaqueridostodos" "queridostodos"      "r"
[4] "x1"                  "x2"                  "x3"
[7] "y1"                  "y2"
>
```

Más observaciones:

- A la nueva variable `nuevaqueridostodos` le ha sido asignado el valor de `queridostodos` dividido 2. Continuaremos con expresiones algebraicas en la próxima sección.
- Puede agregarse un comentario en la línea de comandos iniciándolo con el carácter `#`
R ignora todo lo que figura después de `#` en una línea de comandos.

Para eliminar objetos en el espacio de trabajo use la función `rm()`:

```
> rm(nuevaqueridostodos,r,x1,x2,y1,y2,a,A,aaa,b,bbb,bbbl,ccc,ddd,nuevovector)
> objects()
[1] "queridostodos" "x3"
```

También se pueden eliminar todos los objetos via la opción “Remove all objects” en el “Misc” del menú. Sin embargo, es posible que interese más quitar algunos objetos y guardar otros.

1.4.4 Ventanas de ayuda

Supongamos por ejemplo que interesa saber algo más sobre la función `log()` en R. Los dos comandos siguientes producen el mismo resultado:

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

Se abre una *Ventana de Ayuda* (*Help Window*) con la siguiente información:

Logarithms and Exponentials

Description:

```
'log' computes natural logarithms, 'log10' computes common (i.e.,
base 10) logarithms, and 'log2' computes binary (i.e., base 2)
logarithms. The general form 'logb(x, base)' computes logarithms
with base 'base' ('log10' and 'log2' are only special cases).
```

```
. . . (salteamos una parte)
```

Usage:

```
log(x, base = exp(1))
logb(x, base = exp(1))
log10(x)
log2(x)
. . .
```

Arguments:

x: a numeric or complex vector.

base: positive number. The base with respect to which logarithms are computed. Defaults to `e='exp(1)'`.

. . . .

Value:

A vector of the same length as 'x' containing the transformed values. 'log(0)' gives '-Inf' (when available).

Vemos que la función `log()` de R es la función matemática logaritmo. Esta función toma dos argumentos: “**x**” es la variable u objeto al que se le tomará logaritmo y “**base**” define qué logaritmo será calculado. Note que por defecto la base es $e = 2.718281828\dots$, que corresponde al logaritmo natural. También vemos que hay otras funciones asociadas `log10()` y `log2()` para el cálculo de los logaritmos en base 10 y base 2 respectivamente.

Ejemplos:

```
> log(exp(3))
[1] 3
> log(100)
[1] 4.60517
> log2(16) # igual a log(16,base=2) o simplemente log(16,2)
[1] 4
> log(1000,base=10) # igual a log10(1000)
[1] 3
>
```

Debido a la naturaleza orientada a objetos de R, también podemos utilizar la función `log()` para calcular el logaritmo de vectores numéricos y matrices, componente a componente:

```
> log2(c(1,2,3,4)) # log base 2 del vector (1,2,3,4)
[1] 0.000000 1.000000 1.584963 2.000000
>
```

También se puede acceder al Help desde el menú en la ventana de R. Este incluye al help que se puede acceder a través de la web. También se puede realizar una búsqueda mediante

una palabra clave mediante la función `apropos()`. Por ejemplo, para encontrar todas las funciones de R que contienen la cadena `norm`:

```
> apropos("norm")
[1] "dlnorm"      "dnorm"      "plnorm"      "pnorm"      "qlnorm"
[6] "qnorm"      "qqnorm"     "qqnorm.default" "rlnorm"     "rnorm"
[11] "normalizePath"
```

Observe que ponemos doble comillas a la palabra clave pero también funciona con comillas simples (' ').

Vea qué obtiene con

```
> apropos('test')
```

No podremos acceder utilizando la función `help` a información sobre un tema que no corresponde al nombre de un objeto de R.

```
> help("DNA")
No documentation for 'DNA' in specified packages and libraries:
you could try 'help.search("DNA")'
> help.search("DNA")
```

La función `help.search` permite la búsqueda en documentación del sistema de una cadena de caracteres especificada.

La función `help.start` despliega una ventana desde donde se puede acceder a toda la documentación disponible, uso:

```
> help.start()
```

1.4.5 Imprimir , Guardar el trabajo y salir

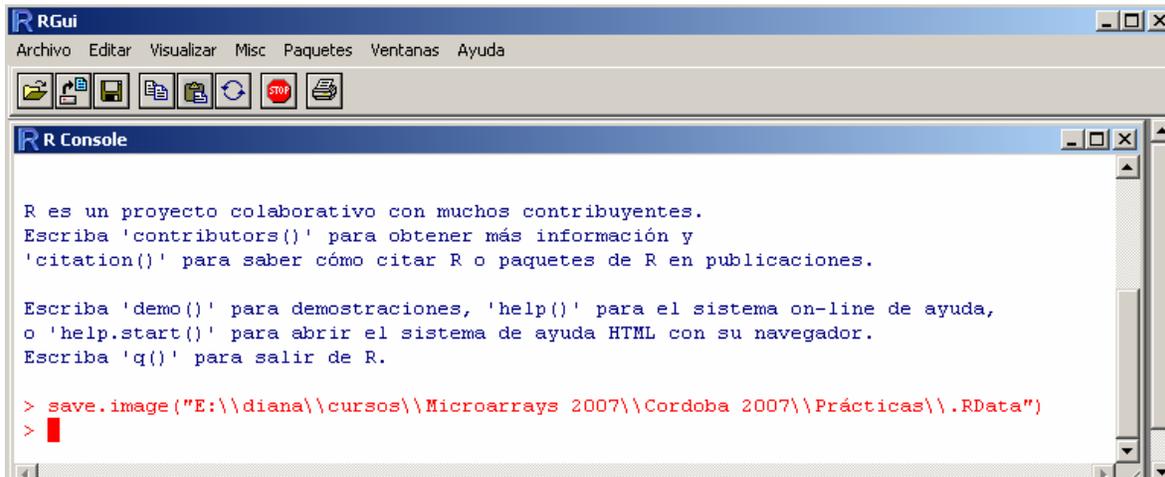
Seleccionando **File** -> **print** del menú principal se pueden imprimir todos los comandos realizados en la sesión junto con todos los resultados aparecidos en la pantalla (incluidos los errores). Otra forma consiste en copiar y pegar todo lo necesario a un editor de textos (sugerencia: utilice el **Courier font** de manera que el formato sea idéntico al de la **Consola de R**). También es posible salvar todo lo que está en la **Consola de R** utilizando **File** -> **Save to File** del menú principal, parados en la Consola de R.

Al salir de R, el programa pregunta si quiere guardar la imagen de su espacio de trabajo (workspace image). Si se clickea en “yes” todos los objetos (los nuevos creados en la sesión actual y los creados en las anteriores) serán guardados y estarán disponibles en la siguiente sesión. Si clickea en “no”, se perderán todos los objetos nuevos y el espacio de trabajo será restaurado al último momento en que la imagen fuera guardada. Es recomendable salvar el trabajo.

Más conveniente aún es *guardar el espacio de trabajo en una carpeta específica*, por ejemplo en C:\microarreglos. Para ello en el **menú principal** de R, seleccionar

File -> Save Workspace -> Save image in
Archivo -> Guardar área de trabajo -> Guardar imagen en

se abre un navegador y se puede elegir la carpeta donde guardar el espacio de trabajo.



Al guardarse se genera un ícono de inicio de R; permite abrir el programa desde esa carpeta y con el espacio de trabajo allí guardado.



1.4.6 Otras fuentes de referencia

Es imposible escribir todo sobre R en un documento de un tamaño razonable, pero se pueden encontrar muchos manuales tutoriales y libros como ayuda para aprender a usar R.

La página

<http://cran.r-project.org/>

es muy útil para ampliar la bibliografía.

Aquí algunos ejemplos más de la documentación disponible

- **El programa R:** Desde el Help del menú se puede acceder a los manuales que vienen con el software. Estos están escritos por el grupo de desarrollo de R. Algunos son largos y específicos, pero el manual “*An Introduction to R*” es una buena fuente de información útil.
- **Documentación:** El sitio de CRAN <http://cran.r-project.org/> tiene contribuciones de los usuarios y algunas son en castellano, como la traducción de “*An Introduction to R*”. Allí también se encuentran

W. J. Owen 2006. **The R Guide**

<http://cran.r-project.org/doc/contrib/Owen-TheRGuide.pdf>

guía de la que he tomado gran parte de este material;

R reference card by Tom Short es un resumen de 4 páginas muy útil.

Emmanuel Paradis 2003. **R para Principiantes**. En castellano
http://cran.r-project.org/doc/contrib/rdebuts_es.pdf

• **Libros:**

Introductory Statistics with R. Peter Dalgaard, Springer-Verlag (2002).
Modern Applied Statistics with S, 4th Ed. by W.N. Venables and B.D. Ripley,
Springer-Verlag (2002).

1.5 Ejercicios

1. Utilice el help para hallar información sobre las funciones `mean` y `median` de R.
2. Obtenga una lista de todas las funciones en R que contienen la cadena de caracteres `test` en su nombre.
3. Cree el vector `info` que contiene su edad, altura (en cm) y código postal.
4. Cree la matriz `Ident` que es la matriz identidad de 3x3.
5. Guarde todos los comandos ejecutados y sus resultados de esta sesión en el archivo `PrimeraR.txt`. (en el menú principal, seleccionar **Archivo** -> **Guardar en Archivo**)