

**Grado en Matemáticas. Doble Grado Matemáticas y Estadística.
Universidad de Sevilla**

Análisis de Datos Multivariantes

Introducción a la plataforma estadística R

Rafael Pino Mejías

**Departamento de Estadística e Investigación Operativa
Universidad de Sevilla**

1. Introducción
2. Tipos de datos.
3. Lectura y escritura de datos
4. Gráficos estadísticos
5. Bucles, “if” y funciones

1. Introducción

- R es un lenguaje y un entorno orientado a la realización de cálculos y gráficos estadísticos.
- Ofrece una gran variedad de técnicas estadísticas, disponibles como funciones que son llamadas desde código fuente R.
- Si bien el entorno gráfico de usuario no es demasiado amigable, se han desarrollado diversos sistemas que facilitan las tareas, por ejemplo RStudio y recientemente Eclipse.
- R puede considerarse como un “GNU S”, ya que implementa un “dialecto” del sistema S, merecedor del premio Software System Award de la Association of Computer Machinery 1988): " [S] has forever altered the way how people analyze, visualize and manipulate data”.

Principales ventajas de R:

- La capacidad de combinar análisis de tipo general con análisis específicos (por ejemplo uso de técnicas de remuestreo sobre modelos de regresión lineal): en definitiva, la capacidad de manipular y modificar datos y funciones.
- Ofrece gráficos de alta calidad.
- La comunidad de R es muy dinámica (crecimiento y actualización de paquetes).
- Dispone de extensiones específicas de nuevas áreas (bioinformática, geoestadística, modelos gráficos, etc.).
- Ofrece un lenguaje de programación orientado a objetos.
- Diversos operadores y funciones para realizar cálculos sobre vectores, matrices y arrays de mayor dimensión.
- Se puede llamar a código de otros lenguajes, como C/C++.

➤ licence()

This software is distributed under the terms of the **GNU General Public License**, either Version 2, June 1991 or Version 3, June 2007.

The terms of version 2 of the license are in a file called COPYING which you should have received with this software and which can be displayed by `RShowDoc("COPYING")`. Version 3 of the license can be displayed by `RShowDoc("GPL-3")`. Copies of both versions 2 and 3 of the license can be found at <https://www.R-project.org/Licenses/>.

A small number of files (the API header files listed in `R_DOC_DIR/COPYRIGHTS`) are distributed under the LESSER GNU GENERAL PUBLIC LICENSE, version 2.1 or later. This can be displayed by `RShowDoc("LGPL-2.1")`, or obtained at the URI given. Version 3 of the license can be displayed by `RShowDoc("LGPL-3")`.

'Share and Enjoy.'

Instalación de R

En la página Web <http://www.r-project.org/>

[Download CRAN > Mirror > Windows > base>Download R-3.3.1](#)

Tras ejecutar este programa, R queda instalado (por defecto) en el directorio

[C:\Archivos de Programa\R\R-3.3.1](#)

Las distintas librerías o paquetes se organizan a su vez en el subdirectorio
[C:\Archivos de Programa\R\R-3.3.1\library](#).

Instalación mínima: librería base más un conjunto de paquetes adicionales.

Algunas de estas librerías están disponibles al entrar en R, otras pueden ser cargadas por el usuario en cualquier momento.

Además existe un gran número de librerías disponibles en la página Web de R que pueden ser descargadas e instaladas, residen en

[Download CRAN > Mirror > Windows > contrib](#)

The Comprehensive R Archive Network - Internet Explorer, optimized for Bing and MSN


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

Archivo Edición Ver Favoritos Herramientas Ayuda

Favoritos Sitios sugeridos Galería de Web Slice

The Comprehensive R Archive Network

Página Seguridad Herramientas



The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2013-09-25, Frisbee Sailing) [R-3.0.2.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.

CRAN

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

About R

- [R Homepage](#)
- [The R Journal](#)

Software

- [R Sources](#)
- [R Binaries](#)
- [Packages](#)
- [Other](#)

Documentation

- [Manuals](#)
- [FAQs](#)
- [Contributed](#)

Listo

Internet 100%

Inicio Bandeja de entra... Microsoft PowerP... The Comprehensi... Dibujo - Paint ES 9:46

The Comprehensive R Archive Network - Internet Explorer, optimized for Bing and MSN

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


Archivo Edición Ver Favoritos Herramientas Ayuda

Favoritos Sitios sugeridos Galería de Web Slice

The Comprehensive R Archive Network

Home RSS Email Print Página Seguridad Herramientas ?

R for Windows



Subdirectories:

base	Binaries for base distribution (managed by Duncan Murdoch). This is what you want to install R for the first time .
contrib	Binaries of contributed packages (managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
Rtools	Tools to build R and R packages (managed by Duncan Murdoch). This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Duncan Murdoch or Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the [R FAQ](#) and [R for Windows FAQ](#).

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

CRAN

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

About R

- [R Homepage](#)
- [The R Journal](#)

Software

- [R Sources](#)
- [R Binaries](#)
- [Packages](#)
- [Other](#)

Documentation

- [Manuals](#)
- [FAQs](#)
- [Contributed](#)

Internet 100%

Inicio Bandeja de entra... Microsoft PowerP... The Comprehensi... Dibujo - Paint ES 9:47



CRAN

[Mirrors](#)

[What's new?](#)

[Task Views](#)

[Search](#)

About R

[R Homepage](#)

[The R Journal](#)

Software

[R Sources](#)

[R Binaries](#)

[Packages](#)

[Other](#)

Documentation

[Manuals](#)

[FAQs](#)

[Contributed](#)

R-3.3.1 for Windows (32/64 bit)

[Download R 3.3.1 for Windows](#) (70 megabytes, 32/64 bit)

[Installation and other instructions](#)

[New features in this version](#)

If you want to double-check that the package you have downloaded exactly matches the package distributed by R, you can compare the [md5sum](#) of the .exe to the [true fingerprint](#). You will need a version of md5sum for windows: both [graphical](#) and [command line versions](#) are available.

Frequently asked questions

- [Does R run under my version of Windows?](#)
- [How do I update packages in my previous version of R?](#)
- [Should I run 32-bit or 64-bit R?](#)

Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for Windows-specific information.

Other builds

- 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-devel snapshot build](#).
- [Previous releases](#)

Note to webmasters: A stable link which will redirect to the current Windows binary release is [<CRAN MIRROR>/bin/windows/base/release.htm](#).

Last change: 2016-06-21, by Duncan Murdoch

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

Documentación:

Documentation > Manuals:

Manuales de R, ya incluidos en las instalaciones y disponibles en la página Web:

An Introduction to R (disponible la traducción al castellano).

The R language definition: describe los tipos de objetos y detalla el proceso de evaluación de las expresiones.

Writing R Extensions: cómo crear nuevos paquetes, cómo escribir ficheros R de ayuda, y en particular la interacción con otros lenguajes (C, C++, Fortran, ...).

R Data Import/Export : trata sobre los recursos para importar y exportar datos.

R Installation and Administration.

R Internals: estructura interna y detalles para el equipo gestor de R.

The R Reference Index: todos los ficheros de ayuda de R y de los paquetes recomendado.

La página Web dispone de más documentación en el apartado *Documentation > Manuals > contributed documentation*. Destacan algunos textos en castellano y algunas guías breves, como la **R Reference Card**.

Cómo citar R:

>citation()

To cite R in publications use:

R Core Team (2016). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.
URL <https://www.R-project.org/>.

A BibTeX entry for LaTeX users is

```
@Manual{  
  title = {R: A Language and Environment for Statistical Computing},  
  author = {{R Core Team}},  
  organization = {R Foundation for Statistical Computing},  
  address = {Vienna, Austria},  
  year = {2016},  
  url = {https://www.R-project.org/},  
}
```

We have invested a lot of time and effort in creating R, please cite it when using it for data analysis. See also 'citation("pkgname")' for citing R packages.

Cómo citar un paquete de R:

> citation("data.table")

To cite package 'data.table' in publications use:

M Dowle, A Srinivasan, T Short, S Lianoglou with contributions from R Saporta and E Antonyan (2015). data.table: Extension of Data.frame. R package version 1.9.6. <https://CRAN.R-project.org/package=data.table>

A BibTeX entry for LaTeX users is

```
@Manual{  
  title = {data.table: Extension of Data.frame},  
  author = {M Dowle and A Srinivasan and T Short and S Lianoglou with contributions from R Saporta and E Antonyan},  
  year = {2015},  
  note = {R package version 1.9.6},  
  url = {https://CRAN.R-project.org/package=data.table},  
}
```

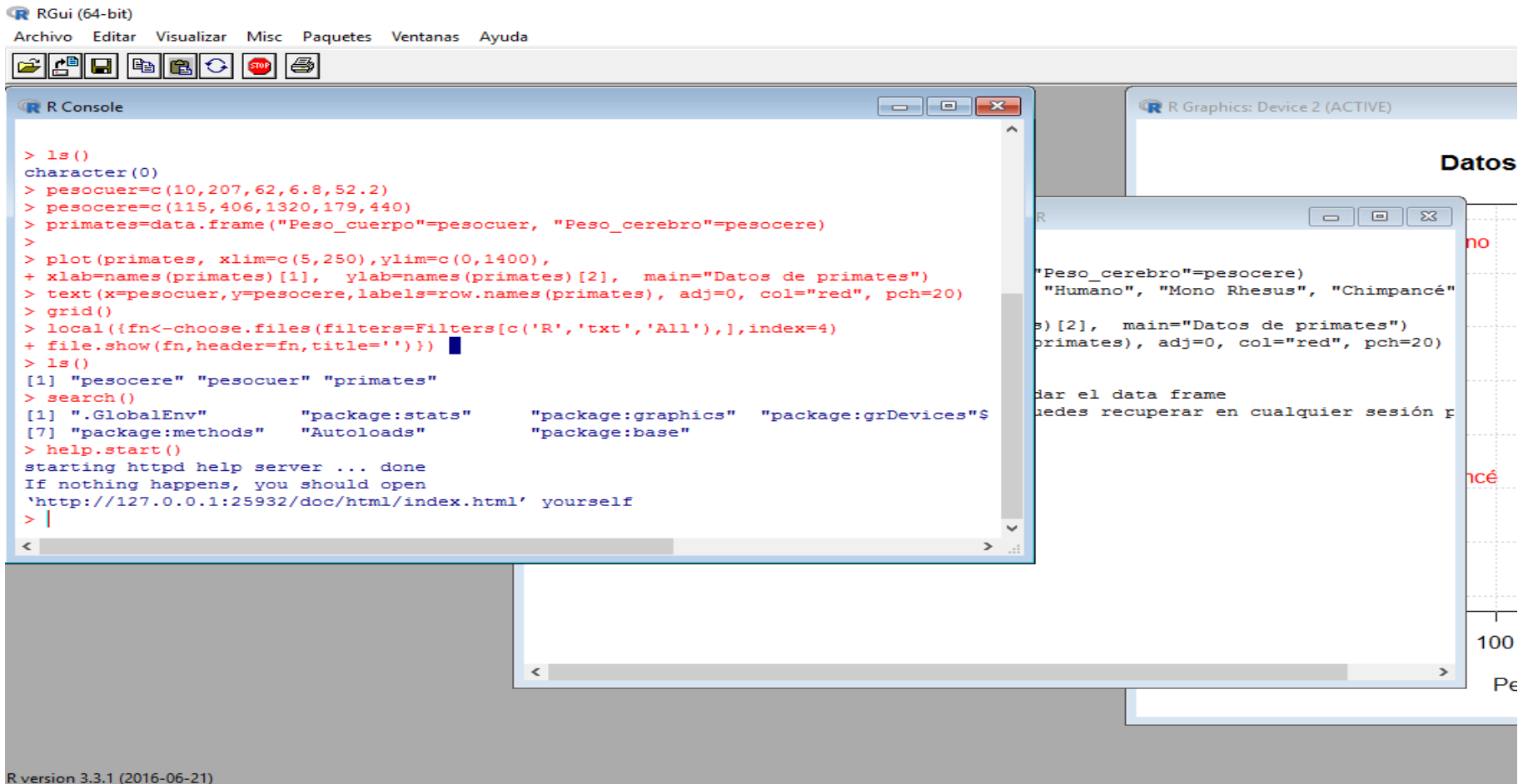
ATTENTION: This citation information has been auto-generated from the package DESCRIPTION file and may need manual editing, see 'help("citation")'.

Comandos y conceptos básicos

- Ayuda: ?, Comandos `help`, `help.search()` y `apropos()`
- Cuidado con las mayúsculas y las minúsculas
- Importar código: comando `source("comandos.R")` permite ejecutar un fichero de instrucciones R.
- Workspace: El comando `getwd()` nos devuelve el directorio de trabajo y el comando `setwd()` nos permite modificarlo. Al iniciar el programa R se abre automáticamente un espacio de trabajo, en él se almacenan todos los datos y funciones usadas durante esa sesión. En cualquier momento se pueden guardar todos los objetos del espacio de trabajo como un archivo con extensión `".RData"`.
- Librerías: Al iniciar el programa R se cargan por defecto unas librerías básicas. A veces es necesario cargar otras librerías para realizar ciertos análisis, a través del comando `library(nombre)`
- El comando `ls()` nos proporciona un listado de los objetos que hay actualmente en el espacio de trabajo. El comando `search()` nos da una lista de las librerías cargadas. Para eliminar uno o varios objetos del espacio de trabajo se usa el comando `rm(objectos)`. El comando `history()` nos devuelve un archivo de texto con los últimos comandos ejecutados.
- R puede ser utilizado como una calculadora de modo interactivo.

Entorno gráfico de usuario de R (R GUI)

Es el entorno que ofrece R. Visualiza el menú de opciones así como las pantallas que estén abiertas en la sesión, en este ejemplo se observa la consola de resultados, una fichero de instrucciones (script) y una gráfica. El menú de opciones depende de la pantalla que esté activa (última donde se haya pulsado).



Entorno gráfico de usuario de R (R GUI)

Con la consola activa, las principales acciones que se pueden realizar son;

Archivo:

- Ejecutar un script
- Crear o cargar un script
- Cargar o guardar áreas (espacios) de trabajo
- Cargar o guardar Históricos (conjuntos de instrucciones ejecutadas en la sesión)
- Cambiar el directorio de trabajo
- Imprimir o guardar una selección de la información mostrada en la consola

Editar:

- Copiar y pegar
- Limpiar consola (la vacía)
- Editor de datos (de un data frame existente)
- Preferencias de la interface gráfica: permite modificar aspectos como el tamaño y color de la letra, el número de filas visualizado de la consola, modificar el fondo, etc.

Entorno gráfico de usuario de R (R GUI)

Con la consola activa, las principales acciones que se pueden realizar son;

Visualizar:

- Permite ver (o desactivar) la barra de herramientas (debajo del menú principal, son iconos para ejecutar directamente ciertas acciones) y la barra de estatus (en la parte inferior de pantalla, para mostrar la versión de R).

Misc:

- Detener los cálculos
- Listar los objetos existentes en la sesión
- Eliminar todos los objetos

Paquetes:

- Cargar un paquete ya instalado
- Instalar o actualizar paquetes desde un espejo CRAN
- Definir el espejo CRAN (mirror)
- Instalar un paquete a partir del fichero .zip

Ventanas:

- Controlar la disposición de las distintas ventanas en la pantalla

Ayuda

Entorno gráfico de usuario de R (R GUI)

Cuando se activa un cuadro de script (Editor R) o una gráfica cambian las opciones disponibles (y la barra de herramientas).

Por ejemplo, al activar una gráfica:

Archivo:

- Guardar el gráfico en un fichero con un determinado formato (entre otros, pdf y jpeg).

Histórico:

- La opción Grabando, permite almacenar los distintos gráficos generados, se puede navegar por ellos mediante las teclas Página Previa y Página Posterior.
- Limpiar el historial de gráficos.

Como alternativa a este entorno, se han desarrollado otros productos que pueden ser más eficientes (no siempre, es habitual encontrar algunos problemas, sobre todo cuando se modifican las opciones de los cuadros de gráficos). Se puede destacar el entorno RStudio.



Choose Your Version of RStudio

RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management. [Learn More](#)

	RStudio Desktop (Free License)	RStudio Desktop (Commercial License)	RStudio Server (Free License)	RStudio Server Pro (Commercial License)
Integrated Development Environment for R	✓	✓	✓	✓
Priority support		✓		✓
Access via Web Browser			✓	✓
Enterprise Security and Access Controls				✓
Project Sharing				✓
Access to Multiple Versions of R				✓
Multiple Concurrent Sessions				✓
Administrative Dashboard				✓
Load Balancing and Resource Management				✓
License	AGPL	Commercial	AGPL	Commercial
Pricing	FREE	\$995/yr*	FREE	\$9,995/yr*

The image shows the top section of the RStudio website. On the left is the RStudio logo. To its right is a horizontal navigation bar with links: 'Products' (highlighted with a blue underline), 'Resources', 'Pricing', 'About Us', and 'Blog'. A magnifying glass icon for search is on the far right. Below the navigation bar, there are four green buttons: 'DOWNLOAD' (with a green border), 'BUY NOW', 'DOWNLOAD', and 'DOWNLOAD'. Below each button is a 'Learn More' link. The first 'Learn More' link is blue, while the others are green. The fourth 'Learn More' link includes the text '(45 Day Free Trial)'.

RStudio Desktop 0.99.903 — Release Notes

RStudio requires R 2.11.1+. If you don't already have R, download it [here](#).

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 0.99.903 - Windows Vista/7/8/10	77.1 MB	2016-07-18	716f28f2143c5e21f4acea5752e284f8
RStudio 0.99.903 - Mac OS X 10.6+ (64-bit)	60.8 MB	2016-07-18	d14a1585b5a5ac0839507b9c04d460d6
RStudio 0.99.903 - Ubuntu 12.04+/Debian 8+ (32-bit)	81.6 MB	2016-07-18	761eae80b0ba4d4cd9051a802ac44e2
RStudio 0.99.903 - Ubuntu 12.04+/Debian 8+ (64-bit)	88.3 MB	2016-07-18	98ea59d3db0e0e0083d3e40535147f64d
RStudio 0.99.903 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	81 MB	2016-07-18	ce2ea1023d99175cb909d0fe66bea4
RStudio 0.99.903 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	81.9 MB	2016-07-18	152f2472e8e6904cf3354afb7b3b9b9

Zip/Tarballs

Zip/tar archives	Size	Date	MD5
RStudio 0.99.903 - Windows Vista/7/8/10	110.6 MB	2016-07-18	53817c5703a5fefbba513e6d05133e1d
RStudio 0.99.903 - Ubuntu 12.04+/Debian 8+ (32-bit)	82.8 MB	2016-07-18	bc2c16be996ed08200f1fde7b9e2b93a
RStudio 0.99.903 - Ubuntu 12.04+/Debian 8+ (64-bit)	89.2 MB	2016-07-18	44c418d506e395c718046df458b07882b
RStudio 0.99.903 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	81.6 MB	2016-07-18	c854a4e536fb717819744fba7aec9e35b5
RStudio 0.99.903 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	82.8 MB	2016-07-18	ad5761417fa07cc4db7dfb91aa535b5

Source Code

A tarball containing source code for RStudio v0.99.903 can be downloaded from [here](#)

RStudio

RStudio es un entorno de programación en R [disponible en la mayor parte de los sistemas operativos](#). Está organizado en cuatro zonas de trabajo distintas:

- ✓ En la zona superior izquierda pueden abrirse y editarse ficheros con código R (aunque también otros ficheros de otro tipo).
- ✓ En la zona inferior izquierda hay una consola de R en la que pueden ejecutarse comandos de R.
- ✓ La zona superior derecha tiene dos pestañas:
 - *Environment*, donde aparece la lista de los objetos creados en memoria.
 - *History*, que contiene el histórico de las líneas de código ejecutadas en R
- ✓ La zona inferior derecha dispone de cinco pestañas:
 - *Files*, que da acceso al árbol de directorios y ficheros del disco duro.
 - *Plots*, donde aparecen los gráficos creados en la consola.
 - *Packages*, que facilita la administración de los paquetes de R instalados en la máquina.
 - *Help*, en el que se abren las páginas de ayuda.
 - *Viewer*, para ver contenido Web local.

2. Tipos de datos

R es un lenguaje de programación orientado a **objetos**.

Un objeto es cualquier estructura de datos admisible en R. Un objeto está formado por uno o más componentes, cada uno de un determinado tipo.

Tipos básicos de datos: *numeric, integer, double, complex, logical, character.*

Una **variable** es una estructura de datos formado por un determinado número de elementos, todos del mismo tipo, y cuyos valores pueden ser modificados durante la ejecución del programa.

Las variables vienen caracterizadas por su tamaño y su tipo de datos.

Las variables más simples sólo contienen un valor, que se asigna con el operador de asignación "=", o bien "<=":

```
A= 10; B= 4+2i; b<- TRUE; trabajo= "profesor"
```

Por el contrario, **una constante**, como indica el nombre, no puede ser modificada. Por ejemplo, 14, 18.21, 2.1E7.

R incluye constantes predefinidas, entre otras: pi, Inf, TRUE, FALSE, NA, NaN.

Se dispone de una gran cantidad y variedad de funciones y operadores predefinidos que actúan sobre los objetos de R.

Operaciones matemáticas básicas: +, -, *, /, ^ (**), sqrt(), %/%, %%

Comparaciones lógicas: <, >, <=, >=, ==, !=

Operaciones lógicas: !, &, |, xor()

Funciones matemáticas: sqrt, cos, log, log10, exp, factorial, choose, gamma, Re, Im,...

Funciones estadísticas: min, max, median, var, sum, cumsum,...

Vector: Colección ordenada de elementos del mismo tipo. Se pueden definir de diversas formas, una de ellas utiliza el operador *concatenate*:

```
x = c(1, 2, 3); y = c("a", "b", "Hola"); z1 = c(TRUE, TRUE, FALSE)
```

#Se puede ver la estructura de cualquier objeto con str():

```
> str(x)
```

```
num [1:3] 1 2 3
```

```
> str(y)
```

```
chr [1:3] "a" "b" "Hola"
```

```
> str(z1)
```

```
logi [1:3] TRUE TRUE FALSE
```

Para acceder a elementos de un vector se utiliza []:

```
> x[1]
```

```
[1] 1
```

```
> y[-1]
```

```
[1] "b"  "Hola"
```

```
> z1[c(1,3)]
```

```
[1] TRUE FALSE
```

Existen vectores predefinidos, por ejemplo:

```
> LETTERS
```

```
[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"  
[20] "T" "U" "V" "W" "X" "Y" "Z"
```

```
> letters
```

```
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"  
[20] "t" "u" "v" "w" "x" "y" "z"
```

Se pueden aplicar funciones a vectores:

```
> sum(x)
```

```
[1] 6
```

```
> sum(x==2)
```

```
[1] 1
```

```
> 100*mean(z1)
```

```
[1] 66.66667      #Porcentaje de elementos TRUE en z1
```

```
> length(y)
```

```
[1] 3              #Número de elementos de un vector
```

Concatenación de vectores

```
x=c(2,10,4,3,1,NA,8); y=c(15,11,9); z=c(x,y)
```

#NA: Not Available, código R para identificar los valores perdidos

Función is.na

```
>is.na(z)
```

```
[1] FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE
```

```
z[is.na(z)]<-0 #Sustituye NA por 0
```

Ordenación

```
> sort(y)
```

```
[1] 9 11 15
```

```
> sort(y,decreasing=TRUE) #ver help(sort)
```

```
[1] 15 11 9
```

```
> rev(y)
```

```
[1] 9 11 15
```

```
> rank(y)
```

```
[1] 3 2 1
```

Secuencias de valores repetidos

```
> rep(3,4)
[1] 3 3 3 3
> rep(c("a", "b"),2)
[1] "a" "b" "a" "b"
> rep(c("a", "b"),each=3)
[1] "a" "a" "a" "b" "b" "b"
```

Secuencias con incrementos

```
> seq(1,9, by = 2)
[1] 1 3 5 7 9
> seq(0,1, length=11)
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> seq(1,6, by = 3)
[1] 1 4
```

Operadores de conjuntos

```
> x <- 1:2; y <- c(1, 5, 8:10)
> union(x, y)
[1] 1 2 5 8 9 10
> intersect(x, y)
[1] 1
> setdiff(y, x)
[1] 5 8 9 10
> setdiff(x,y)
[1] 2
> is.element(2,x) #2 %in%x
[1] TRUE
```


#Muestras

```
> x<- 1:5
```

```
> sample(x) #permutación aleatoria
```

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

```
> sample(x, replace=TRUE) #muestra bootstrap
```

```
[1] 4 1 5 1 2
```

```
> sample(x, size = 2* length(x), replace = TRUE)
```

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

```
> sample(30, 6) #muestra sin reemplazamiento, también simple(1;30,6)
```

```
[1] 27 19 4 26 23 21
```

Para muestrear con probabilidades no iguales:

```
>sample(1:7,3,prob=c(0.05,0.05,0.1,0.3,0.1,0.25,0.15))
```

```
[1] 6 1 4
```

El generador de números (pseudo)aleatorios depende de un valor inicial llamado semilla, se calcula a partir de la hora del ordenador, por lo que es diferente para cada equipo. Se puede inicializar mediante la función `set.seed()`, por ejemplo

```
>set.seed(12345)
```

Un vector cuyos valores son categorías puede ser tratado como **factor**.

Ejemplo:

```
puntuaciones=read.table("puntuaciones.txt",header=TRUE)
puntuaciones
summary(puntuaciones)
puntuaciones$grupo=factor(puntuaciones$grupo) #definir tipo factor
summary(puntuaciones)
levels(puntuaciones$grupo)=c("A","B","C") #definir las etiquetas
summary(puntuaciones)
```

```
puntos  grupo
Min.    :2    A:8    #se trata grupo como factor
1st Qu.:4    B:8    #equivale a table(puntuaciones$grupo)
Median :4    C:8
Mean    :4
3rd Qu.:5
Max.    :5
```

```
by(puntuaciones$puntos,puntuaciones$grupo,mean)
```

```
puntuaciones$grupo: A
[1] 3.75
```

```
-----
puntuaciones$grupo: B
[1] 4.375
```

```
-----
puntuaciones$grupo: C
[1] 3.875
```

Array: Generalización del vector a una estructura con dos o más dimensiones.

En el siguiente ejemplo se define un array bidimensional de 3 filas y 2 columnas:

```
x<-array(c(1,2,3,4,5,6),c(3,2))
```

```
str(x)
```

```
x[3,1]; x[1,]; x[c(1,2),]; x[,2]; x[,-1]; x[2,-1] #Acceso a elementos del array:
```

El array bidimensional es una matriz, y puede ser definido mediante la función **matrix**:

```
A<- matrix(1:10, nrow=5,ncol=2) #llenado por columnas
```

```
B<- matrix(1:10, nrow=5,ncol=2,byrow=TRUE) #llenado por filas
```

```
dim(A) #dimensiones de A (número de filas y columnas)
```

```
C<- t(A) #La traspuesta
```

```
B%*%C #Producto matricial
```

```
C %*%B
```

```
D<- C%*%B
```

```
det(D) #Determinante
```

```
solve(D) #Inversa
```

La función **apply** permite aplicar una función a cada fila o cada columna de una matriz

```
x <- array(rnorm(200),c(100,2))  
#Media de cada columna  
medias<-apply(x,2,mean)  
colMeans(x)  
medias<-apply(x,2,mean,trim=0.2)
```

```
#Total de cada fila  
apply(x,1,sum)  
cbind(apply(x,1,sum))  
rowSums(x)
```

list: Una lista está formada por varias componentes (que pueden ser a su vez otras listas), que pueden ser de distinto tipo.

```
lista <- list(vector = 1:10,  
             palabra = "Hola",  
             matriz = matrix(rnorm(20), ncol = 5),  
             lista.2 =  
               list(a = 5, b =c("Azul", "Rojo" )))
```

```
str(lista)
```

```
lista$vector[4]
```

```
lista$matriz[,2]
```

```
lista$lista.2$b
```

```
lista[[1]] #Otra forma de acceder a las componentes de una lista
```

```
lista[[4]][[1]]
```

lapply y sapply

Permiten aplicar una función a cada element de una lista.

```
x <- list(a = 1:10, beta = exp(-3:3), logico=c(TRUE,FALSE,FALSE,TRUE))
```

```
lapply(x,mean)
```

```
#mediana y cuartiles
```

```
lapply(x, quantile, probs = 1:3/4)
```

```
#sapply:versión más amigable
```

```
sapply(x, quantile)
```

Una estructura de gran importancia es el **data frame**, usualmente empleado para representar un conjunto de datos.

Es una lista que pertenece a una clase particular, *data.frame*, y se puede considerar en general como una matriz de casos por variables, que pueden ser de diferente tipo.

Ejemplo:

```
pesocuer=c(10,207,62,6.8,52.2)
pesocere=c(115,406,1320,179,440)
primates=data.frame("Peso_cuerpo"=pesocuer, "Peso_cerebro"=pesocere)
rownames(primates)=c("Mono Potar", "Gorila", "Humano", "Mono Rhesus", "Chimpancé")
plot(primates, xlim=c(5,250),ylim=c(0,1400),
xlab=names(primates)[1], ylab=names(primates)[2], main="Datos de primates")
text(x=pesocuer,y=pesocere,labels=row.names(primates), adj=0, col="red", pch=20)
grid()
save(primates,file="primates.RData") #Guardar el data frame en un espacio de trabajo
load(file="primates.RData")         #Se puedes recuperar en cualquier sesión posterior
```

```
> primates
```

	Peso_cuerpo	Peso_cerebro
Mono Potar	10.0	115
Gorila	207.0	406
Humano	62.0	1320
Mono Rhesus	6.8	179
Chimpancé	52.2	440

```
> str(primates)
'data.frame':   5 obs. of  2 variables:
 $ Peso_cuerpo : num  10 207 62 6.8 52.2
 $ Peso_cerebro: num  115 406 1320 179 440
```

tapply

Aplica una función a una variable para cada valor posible de una o más variables.

```
library(MASS)
```

```
data(quine)
```

```
?quine
```

```
attach(quine)
```

```
tapply(Days, Age, mean)
```

```
tapply(Days, list(Sex, Age), mean)
```

```
tapply(Days, list(Sex, Age), function(y) sqrt((var(y)/length(y))))
```


3. Lectura y escritura de datos

3.1. Ficheros de texto.

R dispone de una gran variedad de recursos para acceder (importar) a conjuntos de datos en otros formatos, y al contrario, para almacenar datos en otros formatos (exportar).

El principal documento de referencia es **R Data Import/Export**.

Cualquier programa permite guardar los datos en formato texto, que puede ser leído en R de varias formas.

read.table: Permite leer un fichero texto con estructura tabular, es decir, n casos (filas), cada uno de los cuales contiene m variables (columnas).

La forma más simple solo requiere el nombre del fichero de texto:
`datos=read.table("fichero.txt")`

El resultado, en este caso `datos`, es un *data frame*.

Sin embargo admite diversas opciones, destacando las siguientes. Si en la primera fila aparecen los nombres de las variables, pueden leerse mediante la opción *header=TRUE*.

También puede controlarse el código para valores perdidos, por defecto NA, mediante *na.strings="C"*.

Otra opción es *row.names*. Puede ser un vector con los nombres de los casos, o un número indicando la columna del fichero que los contiene. Si hay cabecera y tiene un dato menos que la segunda fila, se toma la primera columna como nombres de fila.

#Ejemplo

```
datos<-read.table("repara.dat", header=TRUE)
str(datos)
attach(datos)
plot(unidades, minutos,col="blue")
```

write.table: Para grabar ficheros en formato texto

#Ejemplo

```
write.table(datos,"Fichero_salida.txt", row.names=FALSE)
```

Para ficheros en formato fijo se puede utilizar read.fwf:

#Ejemplo: los datos siguientes

A1.3-1

B0.1-2

A1.6-1

#se leerían mediante

```
datos<- read.fwf("datosfijo.dat",widths=c(1,3,2))
```

#Para leer las variables 1 y 3:

```
datos<- read.fwf("datosfijo.dat",widths=c(1,-3,2))
```

También admite diversas opciones, entre ellas header.

Otras funciones de lectura:

read.csv #ficheros con separación por comas

read.delim #separación por tabuladores

#Otra opción es **scan**, más eficiente al leer matrices de gran dimensionalidad:

#Ejemplo

#Lectura de una matriz 200x2000 usando scan()

```
t0<- proc.time()
```

```
A <- matrix(scan("matriz.dat", n = 200*2000), 200, 2000, byrow = TRUE)
```

```
t1<- proc.time()
```

```
t1-t0
```

#Lectura de una matriz 200x2000 mediante read.table()

```
t0<- proc.time()
```

```
A <- as.matrix(read.table("matriz.dat"))
```

```
t1<- proc.time()
```

```
t1-t0
```

3.2. Ficheros Excel.

Existen varias librerías que ofrecen funciones para importar el contenido de hojas de cálculo.

Sin embargo suelen dar problemas con la version de 64 bits de R, por lo que en general es preferable utilizarlas con la version de 32 bits.

Se pueden destacar:

- El paquete **xlsx** (función *read.xlsx*) para leer ficheros Excel 2007/2010.
- El paquete **XLConnect** para ficheros Excel anteriores.

Estos y otros paquetes también ofrecen funciones para grabar datos en formato Excel.

Si es posible, se recomienda generar un fichero tipo texto desde Excel y utilizar en R procedimientos para leer ese formato.

Un formato tipo texto es el formato csv (“comma separated values”), que se puede leer en R con `read.csv` o bien `read.csv2`

- `read.csv` supone que los campos están separados por “,”, mientras que
- `read.csv2` trata ficheros donde el separador es “;”

3.3. Acceso a bases de datos

Se pueden acceder a bases de datos en otros formatos.

Para acceder a conjuntos de datos de otros paquetes estadísticos como EpiInfo, Minitab, S-PLUS, SAS, SPSS, Stata, Systat: Librería ***foreign***.

Para bases de datos relacionales, pueden destacarse:

RODBC

ROracle

RPostgreSQL

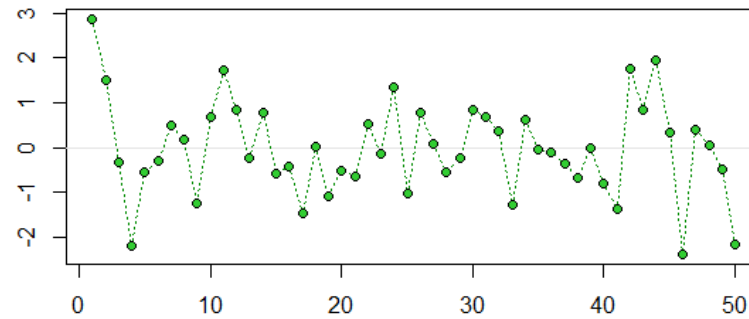
RSQLite

RJDBC

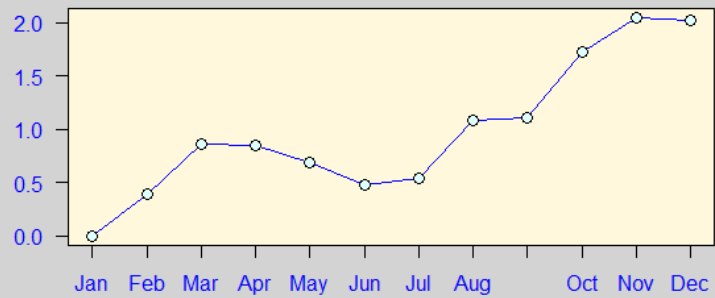
4. Gráficos estadísticos

- Los gráficos son una representación visual de una serie de datos estadísticos. Los gráficos disponibles en R son todos de gran calidad.
- Podemos diferenciar entre dos tipos de funciones gráficas: funciones de alto nivel que crean una nueva gráfica y las funciones de bajo nivel, que añaden elementos a una gráfica ya existente.
- Librerías gráficas: R base, Lattice, Ggplot2, Plotrix ,Grid, etc.
- Como ejemplo podemos ejecutar la función demo:
demo (“graphics”)

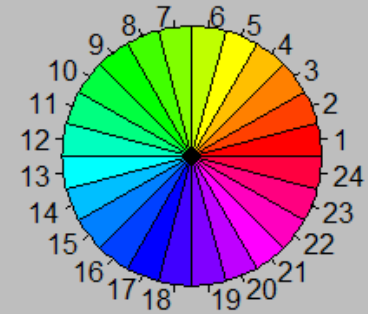
Simple Use of Color In a Plot



The Level of Interest in R



A Sample Color Wheel



Funciones Gráficas Básicas (I)

lines()	Añade líneas al gráfico activo
abline()	Función que añade una o más líneas rectas.
plot()	Función genérica para representar en el plano xy puntos, líneas, etc....
barplot()	Diagramas de barras.
pie()	Diagramas de sectores.
hist()	Histogramas.
boxplot()	Diagramas de box-and-whisker (Caja y bigotes).
stripplot()	Similares a boxplot() con puntos.
sunflowerplot()	Representación en el plano xy de diagramas de girasol.
qqnor()	Diagramas de cuantil a cuantil frente a la distribución normal.
qqplot()	Diagramas de cuantil a cuantil de dos muestras.
qqline()	Representa la línea que pasa por el primer y el tercer cuartil.

Funciones Gráficas Básicas (II)

lines()	Añade líneas a un gráfico.
points()	Añade puntos a un gráfico.
segments()	Añade segmentos a un gráfico.
arrows()	Añade flechas a un gráfico.
polygon()	Añade polígonos a un gráfico.
rect()	Añade rectángulos a un gráfico.
abline()	Añade una recta.
curve()	Representa una función dada.

Gráficos en R base

`plot(x, y, ...)`

Cuando recibe como argumentos dos vectores `x` e `y` de la misma longitud construye la nube de puntos $(x[i], y[i])$.

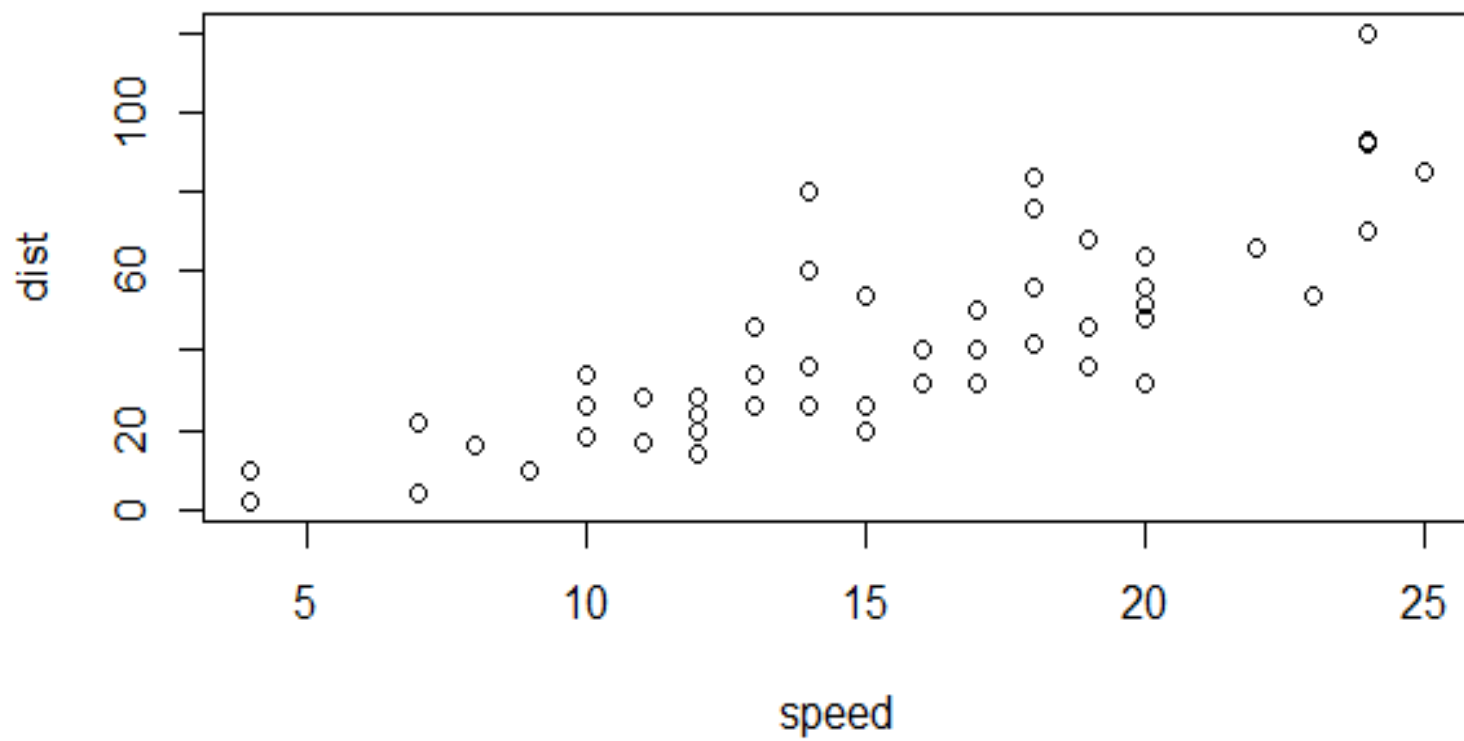
Parámetros:

- `type`: indica el tipo de gráfico a representar.
 - `"p"` puntos
 - `"l"` líneas
 - `"b"` ambos (puntos y líneas)
 - `"h"` histograma , etc
- `main`: añadir título principal
- `sub`: añadir subtítulo con letra más pequeña
- `xlim/ ylim`: especifica los límites inferiores y superiores de los ejes;

Gráficos en R base

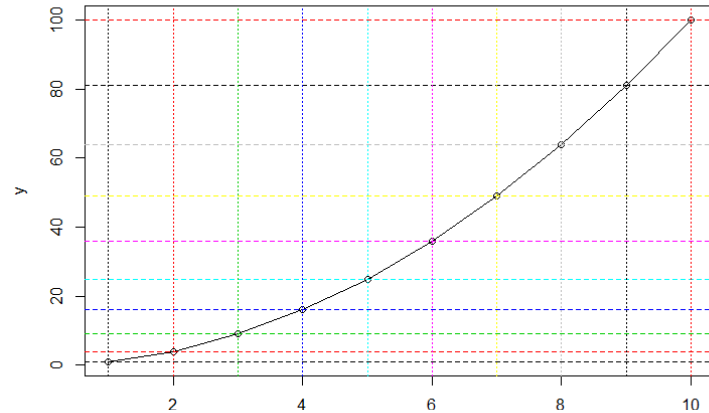
- ***pch*** recibe como valor un entero 0:18 cambia el formato de los puntos de la nube.
- ***col*** recibe como valor una cadena de caracteres que especifica el color a utilizar (blue, red, green, black, etc) en los puntos de la nube.
- Los parámetros ***xlab/ylab*** reciben como valor cadenas de caracteres para etiquetar el eje x e y respectivamente.
- ***main*** recibe como valor una cadena de caracteres para poner un título al gráfico.
- Una vez creado un gráfico se pueden añadir más puntos si no cerramos la ventana del gráfico utilizando la función ***points*** que puede recibir los mismo argumentos que plot.
- Sin cerrar la ventana del gráfico podemos añadir una legenda utilizando la función ***legend***. Su primer argumento es una cadena de caracteres para fijar su posición: "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" y "center" .
- El color y el formato de cada punto en la legenda se especifica con los parámetros ***pch*** y ***col***.

```
> plot(cars)
```

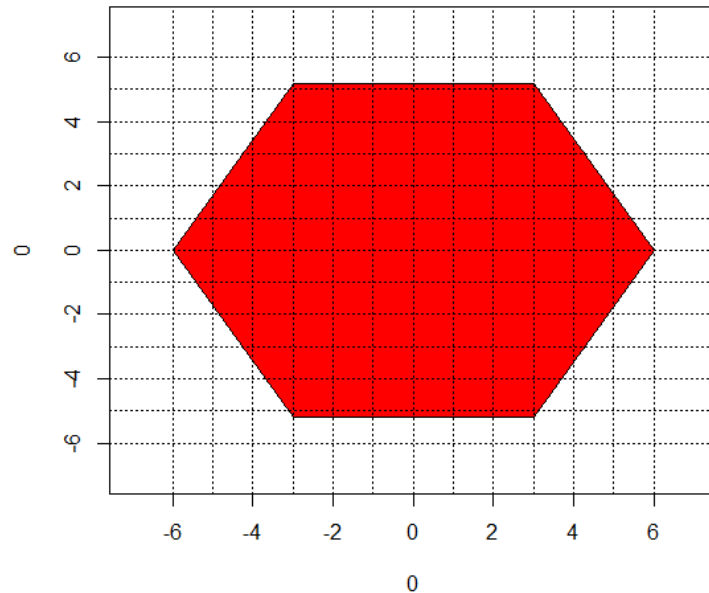


Gráficos en R base

```
x=1:10  
y = x*x  
plot(x,y)  
lines(x,y)  
for(i in 1:10)  
  abline(h=i*i,col=i,lty=2)  
for(i in 1:10)  
  abline(v=i,col=i,lty=3)
```



```
plot(0,0,type="n",xlim=c(-7,7),ylim=c(-7,7))  
x <-c(6,3,-3,-6,-3,3)  
y <-c(0,3*(3^.5),3*(3^.5),0,-3*(3^.5),-3*(3^.5))  
polygon(x,y,col="red",border="black")  
abline(v=-6:6,h=-6:6,lty=3)
```



ggplot2

Librería de R que permiten mejorar los gráficos obtenidos con las librerías base y lattice.

Componentes:

- Data: Datos para representar
- Aesthetic mapping: Características estéticas
- Geom: Objetos geométricos (puntos, líneas, polígonos, áreas...)
- Stat: Transformaciones estadísticas
- Scale: Escalas
- Coord: El sistema de coordenadas
- Faceting: Condicionamiento

Con **ggplot2** los gráficos se crean capa a capa:

- Se definen los datos a utilizar a través de data.frame

```
p <- ggplot(diamonds, aes(carat, price, colour = cut))
```

- Se define la estética asociada a los datos
- Se definen distintas geometrías a utilizar

```
p <- p + layer(geom = "point")
```

qplot(x, y = NULL, ..., data, facets = NULL, margins = FALSE, geom = "auto", stat = list(NULL), position = list(NULL), xlim = c(NA, NA), ylim = c(NA, NA), log = "", main = NULL, xlab = deparse(substitute(x)), ylab = deparse(substitute(y)), asp = NA)

x	x values
y	y values
...	other aesthetics passed for each layer
data	data frame to use (optional). If not specified, will create one, extracting vectors from the current environment.
facets	faceting formula to use. Picks facet_wrap or facet_grid depending on whether the formula is one sided or two-sided
margins	whether or not margins will be displayed
geom	character vector specifying geom to use. Defaults to "point" if x and y are specified, and "histogram" if only x is specified.
stat	character vector specifying statistics to use
position	character vector giving position adjustment to use
xlim	limits for x axis
ylim	limits for y axis
log	which variables to log transform ("x", "y", or "xy")
main	character vector or expression for plot title
xlab	character vector or expression for x axis label
ylab	character vector or expression for y axis label
asp	the y/x aspect ratio

Gráficos en ggplot2 (geom)

<u>Name</u>	<u>Description</u>
abline	Line, specified by slope and intercept
area	Area plots
bar	Bars, rectangles with bases on y-axis
blank	Blank, draws nothing
boxplot	Box-and-whisker plot
contour	Display contours of a 3d surface in 2d
crossbar	Hollow bar with middle indicated by horizontal line
density	Display a smooth density estimate
density 2d	Contours from a 2d density estimate errorbar Error bars
histogram	Histogram hline Line, horizontal
interval	Base for all interval (range) geoms
jitter	Points, jittered to reduce overplotting
line	Connect observations, in order of x value
linerrange	An interval represented by a vertical line
path	Connect observations, in original order
point	Points, as for a scatterplot
pointrange	An interval represented by a vertical line, with a point in the middle
polygon	Polygon, a filled path
quantile	Add quantile lines from a quantile regression
ribbon	Ribbons, y range with continuous x values
rug	Marginal rug plots
segment	Single line segments
smooth	Add a smoothed condition mean
step	Connect observations by stairs
text	Textual annotations
tile	Tile plot as densely as possible, assuming that every tile is the same size
vline	Line, vertical

Gráficos en ggplot2 (aesthetics y statistics)

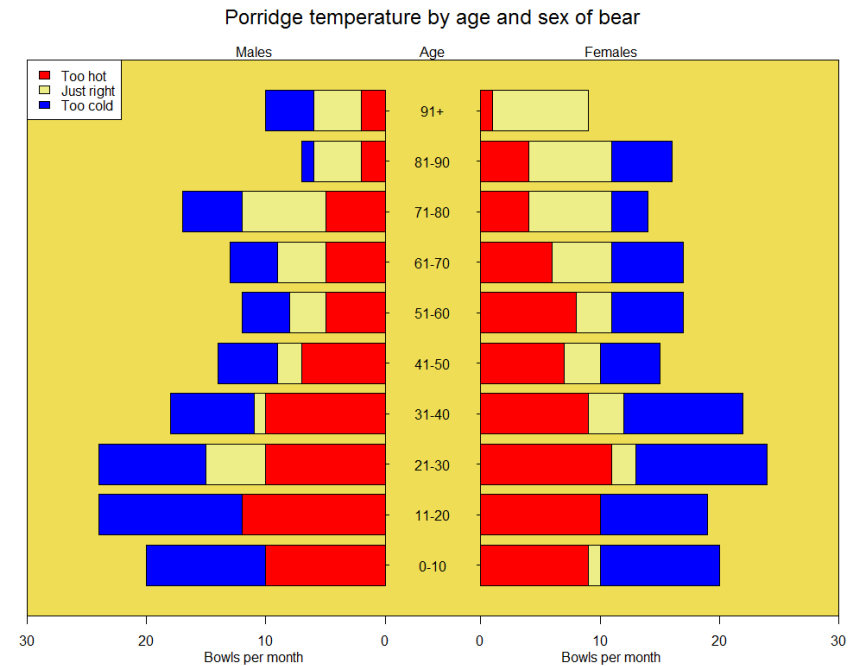
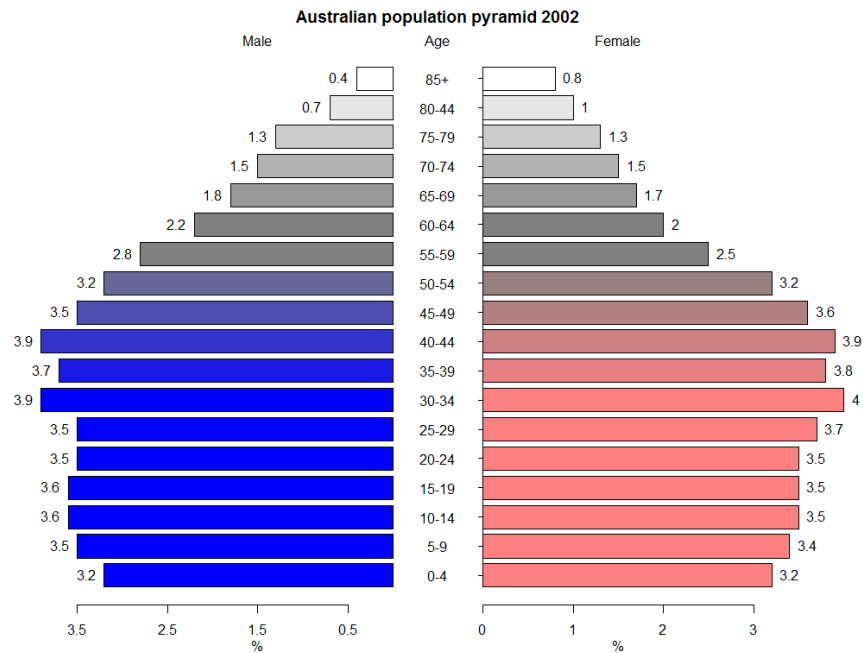
Name	Default stat	Aesthetics
abline	abline	colour, linetype, size
area	identity	colour, fill, linetype, size, x, y
bar	bin	colour, fill, linetype, size, weight, x
bin2d	bin2d	colour, fill, linetype, size, weight, xmax, xmin, ymax, ymin
blank	identity	
boxplot	boxplot	colour, fill, lower, middle, size, upper, weight, x, ymax, ymin
contour	contour	colour, linetype, size, weight, x, y
crossbar	identity	colour, fill, linetype, size, x, y, ymax, ymin
density	density	colour, fill, linetype, size, weight, x, y
density2d	density2d	colour, linetype, size, weight, x, y
errorbar	identity	colour, linetype, size, width, x, ymax, ymin
freqpoly	bin	colour, linetype, size hex, binhex, colour, fill, size, x, y
histogram	bin	colour, fill, linetype, size, weight, x
hline	hline	colour, linetype, size
jitter	identity	colour, fill, shape, size, x, y
line	identity	colour, linetype, size, x, y
linerrange	identity	colour, linetype, size, x, ymax, ymin
path	identity	colour, linetype, size, x, y
point	identity	colour, fill, shape, size, x, y
pointrange	identity	colour, fill, linetype, shape, size, x, y, ymax, ymin
polygon	identity	colour, fill, linetype, size, x, y
quantile	quantile	colour, linetype, size, weight, x, y
rect	identity	colour, fill, linetype, size, xmax, xmin, ymax, ymin
ribbon	identity	colour, fill, linetype, size, x, ymax, ymin
rug	identity	colour, linetype, size
segment	identity	colour, linetype, size, x, xend, y, yend
smooth	smooth	alpha, colour, fill, linetype, size, weight, x, y
step	identity	colour, linetype, size, x, y
text	identity	angle, colour, hjust, label, size, vjust, x, y
tile	identity	colour, fill, linetype, size, x, y
vline	vline	colour, linetype, size

Pirámides de población en PLOTrix

```
pyramid.plot(lx,rx,labels=NA,top.labels=c("Male","Age","Female"),  
main="",laxlab=NULL,raxlab=NULL,unit="%",lxcol,rxcol,gap=1,space=0.2,  
ppmar=c(4,2,4,2),labelcex=1,add=FALSE,xlim,show.values=FALSE,ndig=1, do.first=NULL)
```

lx,rx	Vectors or a matrix or data frame (see Details) which should be of equal length.
labels	Labels for the categories represented by each pair of bars. There should be a label for each lx or rx value, even if empty. If labels is a matrix or data frame, the first two columns will be used for the left and right category labels respectively.
top.labels	The two categories represented on the left and right sides of the plot and a heading for the labels in the center.
main	Optional title for the plot.
laxlab	Optional labels for the left x axis ticks.
raxlab	Optional labels for the right x axis ticks.
unit	The label for the units of the plot.
lxcol,rxcol	Color(s) for the left and right sets of bars. Both of these default to rainbow(length(labels)).
gap	One half of the space between the two sets of bars for the labels in user units.
space	Space between the bars. Should be $0 \leq \text{space} < 1$.
ppmar	Margins for the plot (see Details).
labelcex	Expansion for the category labels.
add	Whether to add bars to an existing plot. Usually this involves overplotting a second set of bars, perhaps transparent.
xlim	Optional x limit for the plot (see Details).
show.values	Whether to display lx and rx at the ends of the bars.
ndig	The number of digits to round the values if displayed.
do.first	Optional expression to evaluate before displaying anything.

Piramides de población en PLOTRIX



Dispositivos Gráficos:

El comando `?Devices` proporciona una lista de los dispositivos gráficos soportados, así como los comandos relacionados:

[windows](#) The graphics driver for Windows (on screen, to printer and to Windows metafile).

[postscript](#) Writes PostScript graphics commands to a file

[pdf](#) Write PDF graphics commands to a file

[pictex](#) Writes LaTeX/PicTeX graphics commands to a file

[png](#) PNG bitmap device

[jpeg](#) JPEG bitmap device

[bmp](#) BMP bitmap device

[xfig](#) Device for XFIG graphics file format

[bitmap](#) bitmap pseudo-device via GhostScript (if available).

La orden `graphics.off()` cierra todos los dispositivos gráficos que estén abiertos.

También relacionado: `dev.list()`, que lista los dispositivos abiertos, y `dev.off()` que cierra el dispositivo actual o el número indicado.

Existen muchas funciones y parámetros gráficos disponibles en R, algunas se han visto en transparencias anteriores.

Gráficos básicos:

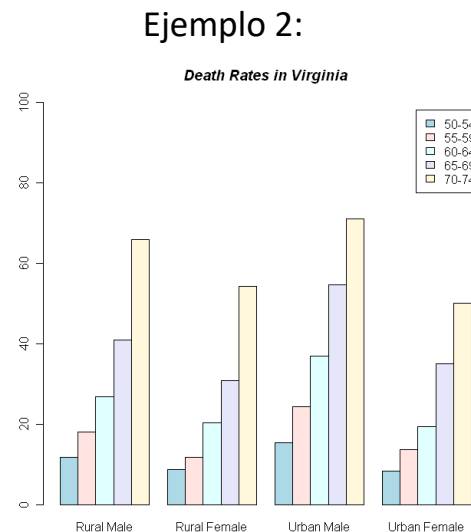
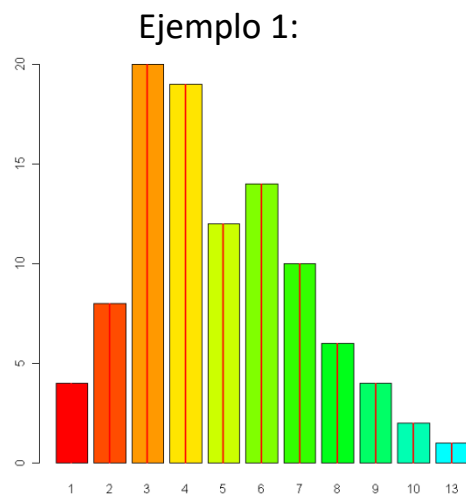
Barplot: Ejemplos extraídos de *help(barplot)*:

Ejemplo 1:

```
tN <- table(Ni <- rpois(100, lambda=5))  
r <- barplot(tN, col=rainbow(20)) #20 colores “consecutivos”  
lines(r, tN, type='h', col='red', lwd=2)
```

Ejemplo 2:

```
barplot(VADeaths, beside = TRUE, col = c("lightblue", "mistyrose", "lightcyan", "lavender", "cornsilk"),  
legend = rownames(VADeaths), ylim = c(0, 100))  
title(main = "Death Rates in Virginia", font.main = 4)
```



Existen muchas funciones y parámetros gráficos disponibles en R, algunas se han ido viendo en transparencias anteriores.

Visualización de algunas opciones gráficas:

#Ejemplo

```
plot(c(0,31),c(1,22),type="n",ylab="",axes=FALSE,xlab="")
points(1:26,rep(21,26),pch=0:25)
text(0:26,rep(22,26),c("pch",paste(0:25)),srt=90,cex=0.65)
```

```
for (i in 1:8)
{
  par(font=i)
  ypos<- 21-i
  points(0:31,rep(ypos,32), pch=64:95)
  axis(2,at=ypos, labels=paste("font =",i), las=1)
}
```

```
par(font=1)
for (i in 1:6)
{
  lines(c(0,31), c(i,i),lty=i)
  axis(2,at=i,labels=paste("lty =", i), las=1)
}
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	
	□	◇	△	+	×	◇	▽	◇	※	米	⊕	+	⊗	田	×	⊗	■	◆	▲	◆	●	○	□	◇	△	▽	
font=1 -	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
font=2 -	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
font=3 -	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
font=4 -	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
font=5 -	≡	A	B	X	Δ	B	Φ	Γ	H	I	9	K	A	M	N	O	Π	⊕	P	S	T	Y	ς	Ω	Ξ	Ψ	Z
font=6 -	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
font=7 -	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
font=8 -	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

lty = 6	-----
lty = 5	-----
lty = 4	-----
lty = 3
lty = 2	-----
lty = 1	_____

Pie: Ejemplo del *help(pie)*.

```
pie.sales <- c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)
```

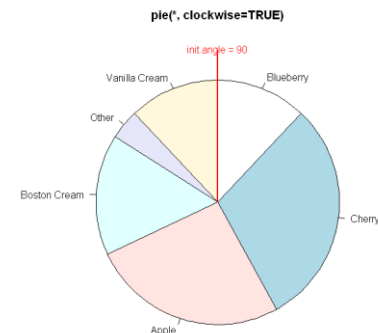
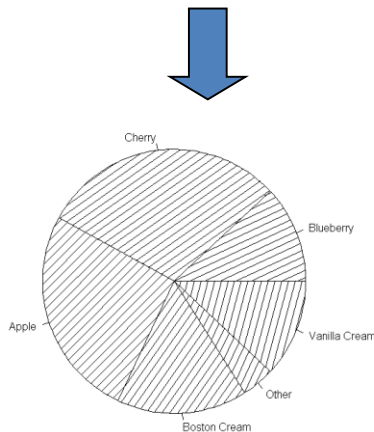
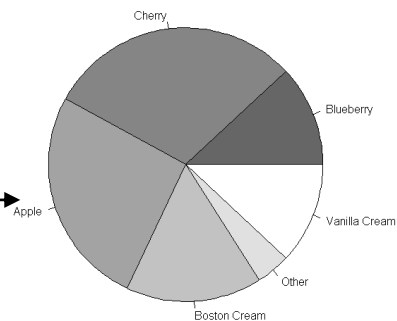
```
names(pie.sales) <- c("Blueberry", "Cherry", "Apple", "Boston Cream", "Other", "Vanilla Cream")
```

```
pie(pie.sales) # Colores por defecto
```

```
pie(pie.sales, col = c("purple", "violetred1", "green3", "cornsilk", "cyan", "white"))
```

```
pie(pie.sales, col = gray(seq(0.4,1.0,length=6)))
```

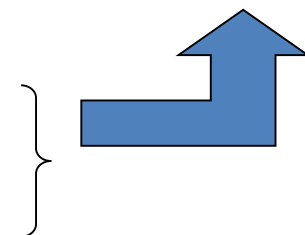
```
pie(pie.sales, density = 10, angle = 15 + 10 * 1:6) #Grosor y pendiente del sombreado
```



```
pie(pie.sales, clockwise=TRUE, main="pie(*, clockwise=TRUE)")
```

```
segments(0,0, 0,1, col= "red", lwd = 2)
```

```
text(0,1, "init.angle = 90", col= "red")
```



Caras de Chernoff:

```
library(TeachingDemos)
data(longley)
faces(longley[1:9,])
```

```
set.seed(17)
faces(matrix(sample(1:1000,128,),16,8),
main="random faces")
```



1947



1948



1949



1950



1951



1952



1953



1954



1955



random faces

1



2



3



4



5



6



7



8



9



10



11



12



13



14



15



16

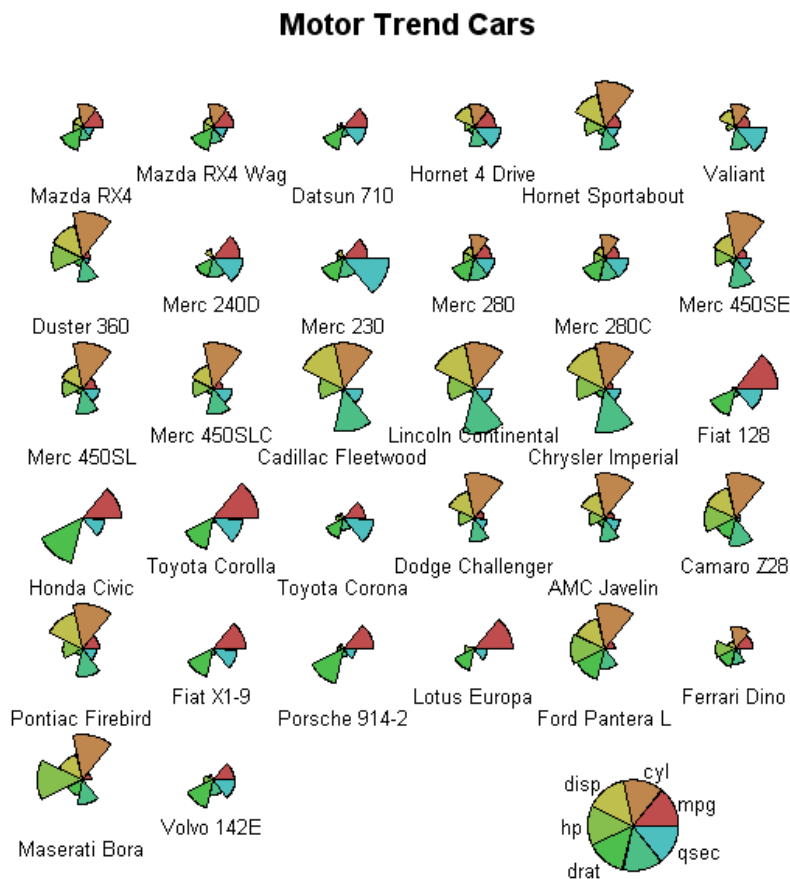


La función *stars* también permite obtener representaciones gráficas de datos multidimensionales.

#Ejemplo:

```
palette(rainbow(12, s = 0.6, v = 0.75))
```

```
stars(mtcars[, 1:7], len = 0.8, key.loc = c(12, 1.5), main = "Motor Trend Cars",  
draw.segments = TRUE)
```



Gráficos Trellis:

La librería *lattice* dispone de una implementación de gráficos *Trellis*, un entorno de Visualización desarrollado en los laboratorios Bell en los años 90s.

help(lattice) permite acceder a la ayuda y ejemplos. Algunas secuencias de instrucciones extraídas de esta ayuda: `library(lattice)`

#Ejemplo 1

```
wireframe(volcano, shade = TRUE, aspect = c(61/87, 0.4), light.source = c(10,0,10))
```

#Ejemplo 2

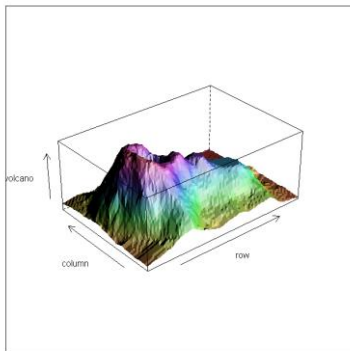
```
cloud(prop.table(Titanic, margin = 1:3), type = c("p", "h"), strip = strip.custom(strip.names = TRUE),  
scales = list(arrows = FALSE, distance = 2), panel.aspect = 0.7, zlab = "Proportion")[, 1]
```

#Ejemplo 3

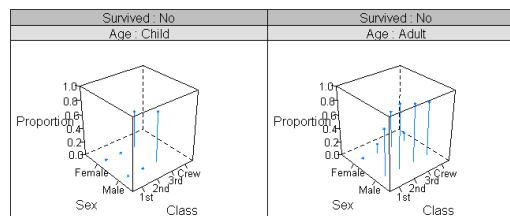
```
Depth <- equal.count(quakes$depth, number=8, overlap=.1)
```

```
xyplot(lat ~ long | Depth, data = quakes)
```

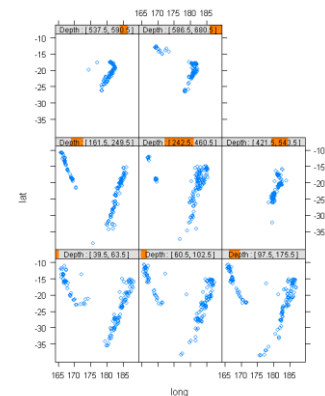
```
update(trellis.last.object(), strip = strip.custom(strip.names = TRUE, strip.levels = TRUE), par.strip.text = list(cex =  
0.75), aspect = "iso")
```



Ejemplo 1



Ejemplo 2



Ejemplo 3

#Ejemplo: Dibujo de imágenes

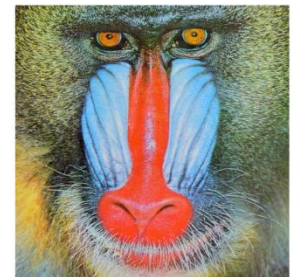
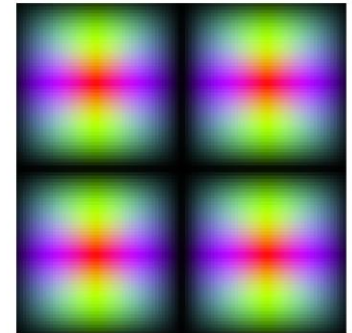
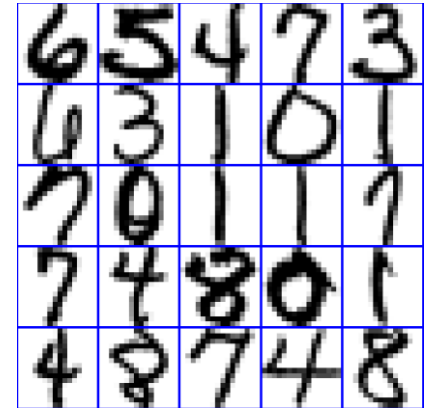
```
library(pixmap)
##Dibujo de dígitos escaneados
datos<-as.matrix(read.table("zip.dat",row.names=1,header=TRUE))
summary(datos) #Cada pixel toma valores en [-1,1]
paranti<- par(no.readonly=TRUE)
par(mfrow=c(5,5),mar=c(0,0,0,0)+0.15,bg="blue")
for (i in 1:25)
{
  matriz<- matrix(-datos[i,2:257],16,16,byrow=T)
  x<-pixmapGrey(matriz)
  plot(x)
}
par(paranti)
```

#RGB

```
x <- seq(-3,3,length=100)
z1 <- outer(x,x,function(x,y) abs(sin(x)*sin(y)))
z2 <- outer(x,x,function(x,y) abs(sin(2*x)*sin(y)))
z3 <- outer(x,x,function(x,y) abs(sin(x)*sin(2*y)))
z <- pixmapRGB(c(z1,z2,z3), 100, 100, bbox=c(-1,-1,1,1))
plot(z, axes=TRUE)
```

#Lectura y dibujo de una imagen JPEG

```
library(jpeg)
x <- readJPEG("mandril.jpg")
z <- pixmapRGB(c(x[,1],x[,2],x[,3]), 512,512, bbox=c(-1,-1,1,1))
plot(z)
```



Otras funciones destacables: rect y polygon, para dibujar rectángulos y polígonos, ver help() o example().

Expresiones matemáticas en las anotaciones:

Se puede hacer con la ayuda de la función *expression*, su sintaxis es similar a la de LATEX.

#Ejemplo (de help(legend)):

```
x <- rexp(100, rate = .5)
```

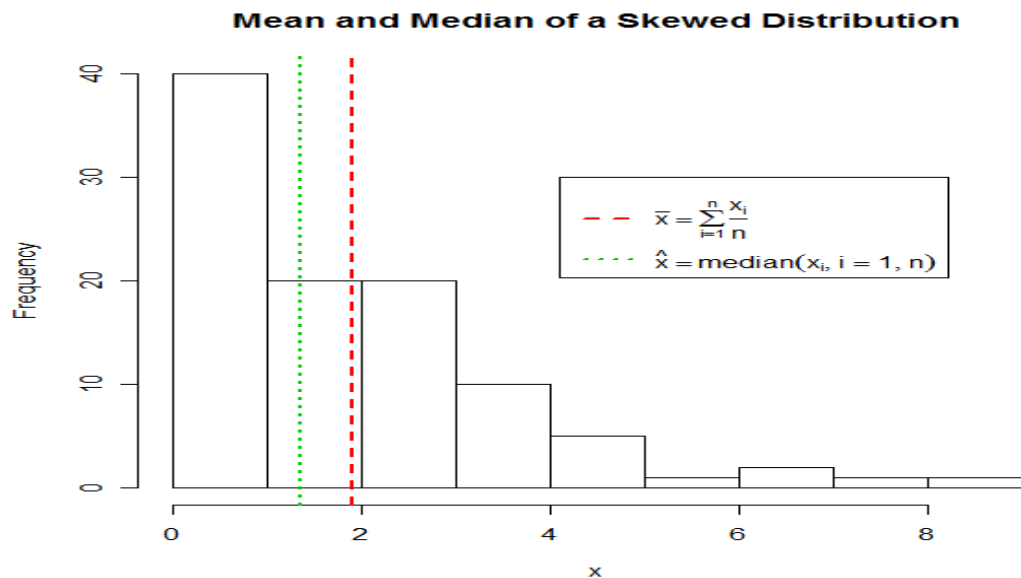
```
hist(x, main = "Mean and Median of a Skewed Distribution")
```

```
abline(v = mean(x), col=2, lty=2, lwd=2)
```

```
abline(v = median(x), col=3, lty=3, lwd=2)
```

```
ex12 <- expression(bar(x) == sum(over(x[i], n), i==1, n),  
                    hat(x) == median(x[i], i==1,n))
```

```
str(legend(4.1, 30, ex12, col = 2:3, lty=2:3, lwd=2))
```



Dibujos 3D:

Función **persp**:

```
library(mvtnorm)
```

```
x<-c(-30:30)/10
```

```
y<-c(-30:30)/10
```

```
media<-c(0,0)
```

```
rho<- 0.7
```

```
matriz<-array(c(1,rho,rho,1),c(2,2))
```

```
nx<-length(x)
```

```
ny<-length(y)
```

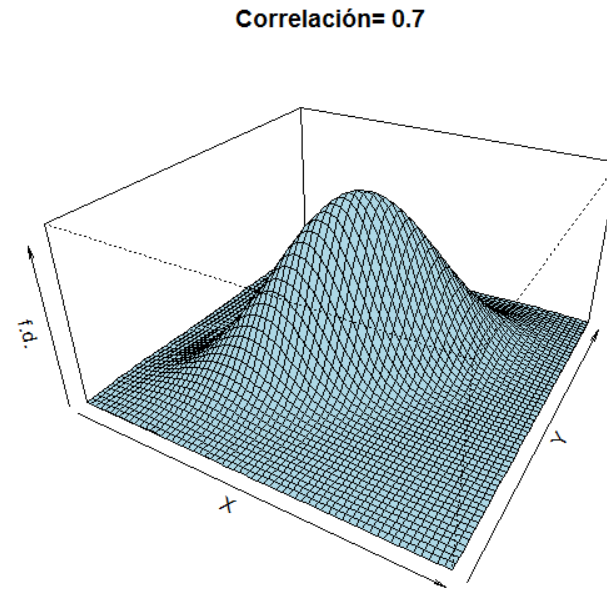
```
z<-array(0,c(nx,ny))
```

```
for (i in 1:nx)
```

```
  for (j in 1:ny)
```

```
    z[i,j]<-dmvnorm(c(x[i],y[j]),mean=media,sigma=matriz)
```

```
persp(x,y,z, theta = 30, phi = 30, expand = 0.5, col = "lightblue", main=paste("Correlación=",rho), xlab="X",  
ylab="Y", zlab="f.d.")
```



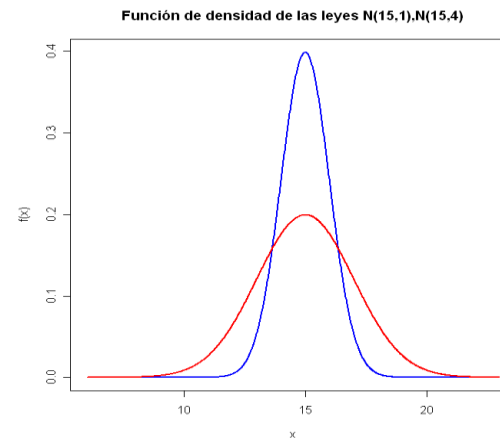
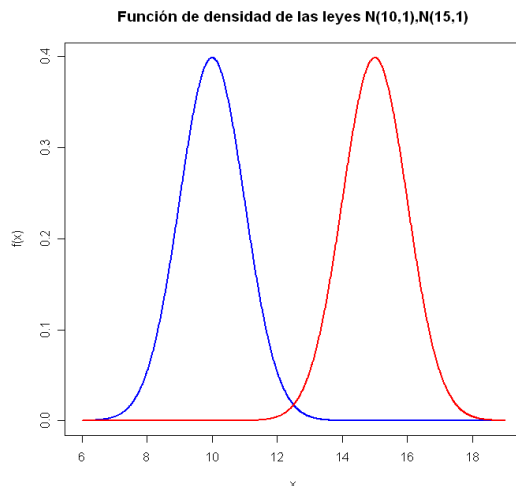
También: función *scatterplot3d* en la librería del mismo nombre.

#Visualización de distribuciones estadísticas

```
curve(dnorm(x),-4,4,1000,lwd=2,col="blue",  
main="Función de densidad de la ley N(0,1)",ylab="fd N(0,1)")  
curve(pnorm(x),-4,4,1000,lwd=2,col="blue",main="Función de distribución de la ley N(0,1)",  
ylab="FD N(0,1)")
```

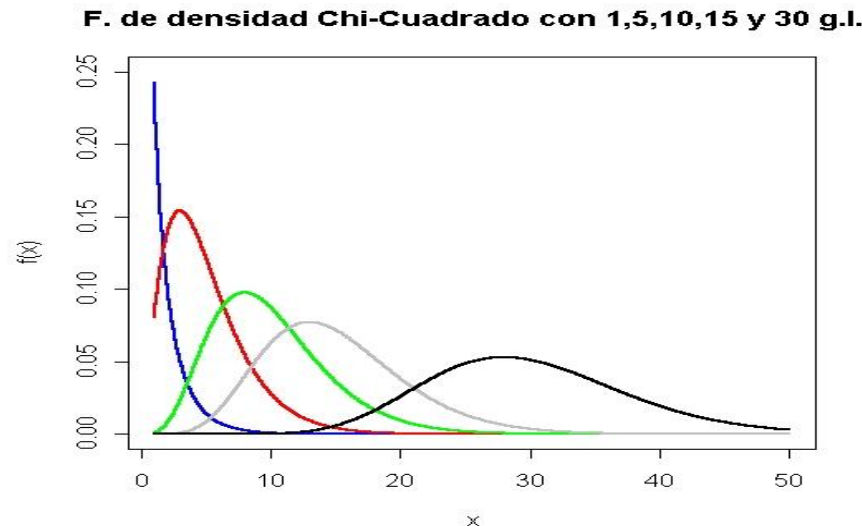
```
curve(dnorm(x,10),6,19,1000,lwd=2,col="blue",  
main="Función de densidad de las leyes N(10,1),N(15,1)",ylab="f(x)")  
curve(dnorm(x,15),6,19,1000,lwd=2,col="red",add=TRUE)
```

```
curve(dnorm(x,15,1),6,23,1000,lwd=2,col="blue",  
main="Función de densidad de las leyes N(15,1),N(15,4)",ylab="f(x)")  
curve(dnorm(x,15,2),6,23,1000,lwd=2,col="red",add=TRUE)  
qnorm(0.975) #Cuantiles de la ley normal  
qnorm(0.95)
```



#Densidad y Distribución Chi-cuadrado

```
curve(dchisq(x,1),1,50,1000,lwd=2,col="blue",
main="F. de densidad Chi-Cuadrado con 1,5,10,15 y 30 g.l.",
ylab="f(x)",ylim=c(0,0.25))
curve(dchisq(x,5),1,50,1000,lwd=2,col="red",add=TRUE)
curve(dchisq(x,10),1,50,1000,lwd=2,col="green",add=TRUE)
curve(dchisq(x,15),1,50,1000,lwd=2,col="grey",add=TRUE)
curve(dchisq(x,30),1,50,1000,lwd=2,col="black",add=TRUE)
curve(pchisq(x,1),0,50,1000,lwd=2,col="blue",
main="F. de distribución Chi-Cuadrado con 1,5,10,15 y 30 g.l.", ylab="f(x)",ylim=c(0,1))
curve(pchisq(x,5),0,50,1000,lwd=2,col="red",add=TRUE)
curve(pchisq(x,10),0,50,1000,lwd=2,col="green",add=TRUE)
curve(pchisq(x,15),0,50,1000,lwd=2,col="grey",add=TRUE)
curve(pchisq(x,30),0,50,1000,lwd=2,col="black",add=TRUE)
qchisq(0.975,10) #Cuantiles
```



5. Bucles, “if” y funciones

- **For:** para repetir un grupo de instrucciones un número de iteraciones previamente fijado.
- Si no se sabe ese número, debe usarse *while* o *repeat*.
- Los comando tipo *apply* suelen ser más eficientes que la instrucción *for*, son preferibles cuando sea posible su aplicación (por ejemplo, calcular la media de cada columna de una matriz).

#Ejemplo

#Ilustración del Teorema Central del Límite

M<-5000 #Núm. de muestras

vn<-c(3,5,15,30) #Tamaños de cada muestra

paranti<- par(no.readonly=T)

par(mfrow=c(2,2))

for (n in vn)

{ x<- matrix(rpois(M*n,1),nrow=M,ncol=n)

medias<- rowMeans(x)

hist(medias,freq=FALSE, col="blue", main=paste("n =",n), ylab=expression(f[i] /a[i]))

}

par(paranti)

while

```
eps <- 1; s <- 0; n <- 0
while(eps > .001)
{
  n <- n + 1
  x <- runif(1,-1,1)
  y <- runif(1,-1,1)
  if(x^2 + y^2 < 1) s <- s + 1
  piest<- 4*s/n
  eps = abs(piest- pi)
}
piest ; n
```

repeat

```
eps <- 1; s <- 0; n <- 0
repeat
{
  n <- n + 1
  x <- runif(1,-1,1)
  y <- runif(1,-1,1)
  if(x^2 + y^2 < 1) s <- s + 1
  piest<- 4*s/n
  eps = abs(piest- pi)
  if (eps <= 0.001) break
}
piest ; n
```

Funciones.

Ejemplos de la definición de nuevas funciones:

```
fahrenheit<-function(gc)  
{9/5*gc+32}
```

#El último valor no asignado es la salida

*#También se puede generar explícitamente un objeto de salida con
#return()*

```
celsios<-25:40  
fahrenheit(25:40)  
conversion<-data.frame("Grados Celsios"=celsios,"Grados  
Fahrenheit"=fahrenheit(celsios))  
str(conversion)  
print(conversion)
```

Funciones. Recursividad

Ejemplo: Para $m \geq 1$ se define el m -ésimo código de Reed-Muller de primer orden como el código lineal binario con matriz generadora $G(m)$, definida como sigue:

$G(0)=1$

Para $m \geq 1$:

$$G(m) = \begin{bmatrix} G(m-1) & G(m-1) \\ 0 \dots 0 & 1 \dots 1 \end{bmatrix}$$

```
CreaMatrizGene<-function (m)
{
  auxi <- array(NA,c(m+1,2^m))
  if (m==0) auxi=1 else
  if (m>=0)
  {
    B<-c(rep(0,2^(m-1)),rep(1,2^(m-1)))
    A<-cbind(CreaMatrizGene (m-1), CreaMatrizGene (m-1))
    auxi<-rbind(A,B)
  }
  row.names(auxi)<- NULL
  auxi
}
```

```
m<-3
G<- CreaMatrizGene(m)
s<- sample(c(0,1),m+1,rep=T)
S
s%*%G %%2
```

```
> G
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    1    1    1    1    1    1    1
[2,]    0    1    0    1    0    1    0    1
[3,]    0    0    1    1    0    0    1    1
[4,]    0    0    0    0    1    1    1    1
> s
[1] 1 0 1 1
> s%*%G %%2
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    1    0    0    0    0    1    1
```

if, ifelse

```
f10<-function(x)
{
  if(x>10) print("Mayor de 10")
  else print("No supera 10")
}
f10(1); f10(12); f10(10)
```

```
par.impar<-function(x)
{
  ifelse(x%%2==1, "Impar",
  "Par")
}
par.impar(32)
par.impar(89)
```

Funciones. Switch.

```
localiz<- function(x, tipo)
{switch(tipo,
        media=mean(x),
        mediana=median(x),
        "No aplicable")
}
```

```
localiz(rnorm(10), "media")
localiz(rnorm(10), "mediana")
localiz(rnorm(10), "recortada")
```