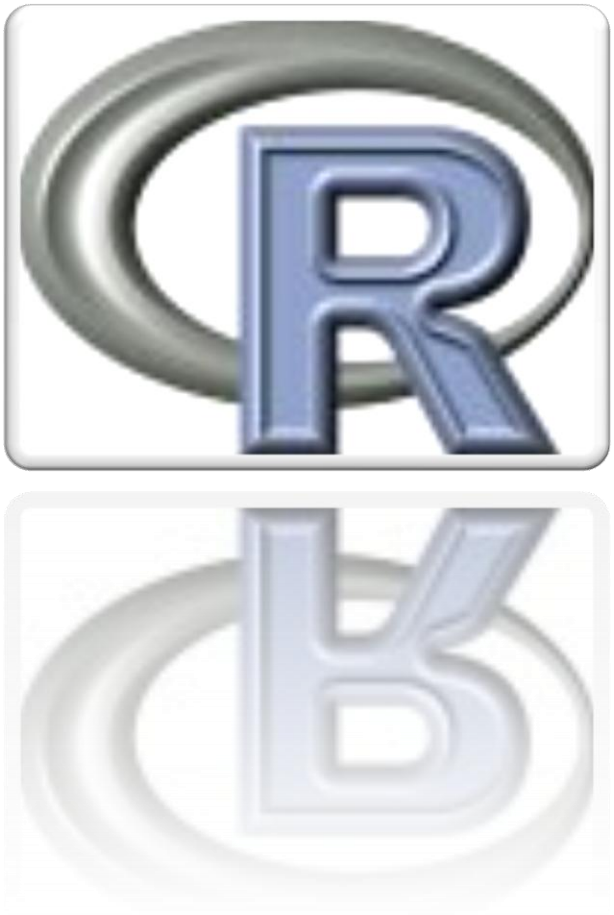


[R pa tontos]

por

[José Enrique Martín García
UP Gimialcón © 2014]

1 TABLA DE CONTENIDO

1. INTRODUCCION

R es un lenguaje de programación muy potente, aporta un entorno de trabajo orientado a resolver principalmente problemas de Estadística y representación gráfica aunque últimamente las posibilidades se están extendiendo a otras áreas del cálculo numérico. A estos sistemas se les denomina a veces PSE, Problem Solving Environments, como Matlab, Mathematica, etc.).

La principal ventajea de R es que es de uso libre en el dominio público, y resulta del esfuerzo cooperativo de personas e instituciones académicas relevantes relacionadas con la Estadística y la Computación en todo el mundo.

El lenguaje utiliza elementos clásicos de programación, con funciones, instrucciones, expresiones aritméticas, lógicas, estructuras for, if, etc. Se puede trabajar tanto en modo programa, con un conjunto de instrucciones, como en modo de comandos interactivos. Ofrece un entorno básico interactivo a través de una ventana de diálogo llamada R Console, y otro paquete, R Commander, que ofrece más interactividad en sus menús para las aplicaciones estadísticas.

La forma de trabajar y la potencialidad de R es muy parecida a otros lenguajes de programación de pago como puede ser Matlab. Da la posibilidad de trabajar en un entorno de trabajo más amigable con el usuario que otros lenguajes como son Fortran o C++.

Se puede utilizar R directamente o combinado con otros programas que ofrecen la interface como es RStudio (que también es gratuito).

Si no has trabajado nunca con lenguajes de programación y no has programado nunca, no te preocupes R es la solución para comenzar a trabajar. En este documento se dan las nociones básicas para descargarlo, instalarlo y dar los primeros pasos. En muchos sitios dicen “Aprende Ingles en 10 días”, “Aprende programación de APPs para móviles en 100 h” etc. etc. que por lo general no funciona ninguna. Aquí te aseguro que en 2 o 3 h y sin tener idea de programación puedes hacer tus primeros “pinitos” en R.

2. COMENCEMOS:

2.1 INSTALACION DE R:

Estos son los pasos que se deben dar

1.- Entrar en la página web: http://www.r-project.org/

2.- Pulsar Download para conectar con CRAN. Aparecen un conjunto de direcciones web en el mundo (mirrors) donde está disponibles copias del software para ser descargadas. Por proximidad se puede elegir la de España, asociada al CSIC: http://cran.es.r-project.org/

3.- En el recuadro Download and Install R, seleccionar Windows, si es el caso, para descargar una versión precompilada binaria del sistema R. Resulta una página con el título R for Windows

4.- En la ventana anterior pulsar base, para descargar el paquete básico. Resulta una página con el título R-3-2-1 for Windows (32/64 bit) (depende de la fecha en que te bajes el programa). En ella, elegir el hipervínculo R-3-2.1-win.exe,

Elegimos Guardar, es el programa ejecutable (.exe) instalador del sistema R básico. Lo descargamos en la carpeta que nos convenga, por ejemplo en el Escritorio. Lo ejecutamos con doble clic, y en la ventana de diálogo que resulta elegimos Ejecutar.

5.- Ejecuta el programa R, una vez instalado. Lo haremos pulsando un icono que es una letra R grande que al instalar nos habrá generado en nuestro escritorio. También eligiendo el programa por la vía >>Inicio>Programas>R>R 3.2.1

2.2 INSTALACION DE RSTUDIO:

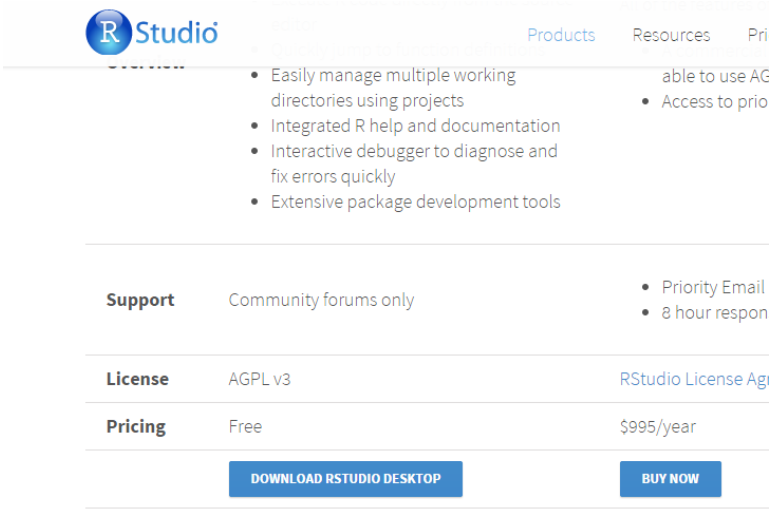
RStudio es un entorno de desarrollo integrado (IDE) para R. Es software libre con licencia GPLv3 y se puede ejecutar sobre distintas plataformas (Windows, Mac, or Linux) o incluso desde la web usando RStudio Server.

Estos son los pasos que se deben dar para instalarlo:

1.- Entrar en la página web: http://www.rstudio.org/

2.- Pulsar el botón Download RStudio

3.- Nos saldrá una pantalla como la de la figura y pulsar el botón Download RStudio Desktop



4.- Existen instalaciones para diversas plataformas. Nosotros elegimos

Rstudio 0.98.1091.Windows XP/Vista/7/8

Installers for ALL Platforms

Installers	Size	Date	MD5
RStudio 0.98.1091 - Windows XP/Vista/7/8	45 MB	2014-11-06	910fba345c0555597bda498cad1302b0
RStudio 0.98.1091 - Mac OS X 10.6+ (64-bit)	38.4 MB	2014-11-06	9c7d2cea702cf478a4a774b79134b3ee
RStudio 0.98.1091 - Debian 6+/Ubuntu 10.04+ (32-bit)	53 MB	2014-11-06	0bc579cbee43a514e3fb4569959a0ada
RStudio 0.98.1091 - Debian 6+/Ubuntu 10.04+ (64-bit)	54.9 MB	2014-11-06	1e88e6775993daa8cf7d4d89f76af7e0
RStudio 0.98.1091 - Fedora 13+/openSUSE 11.4+ (32-bit)	53.4 MB	2014-11-06	3ae5923956166f90ecc1cb721b02f90f
RStudio 0.98.1091 - Fedora 13+/openSUSE 11.4+ (64-bit)	55 MB	2014-11-06	6d1ac08ceed731f5750f3de9a911511b

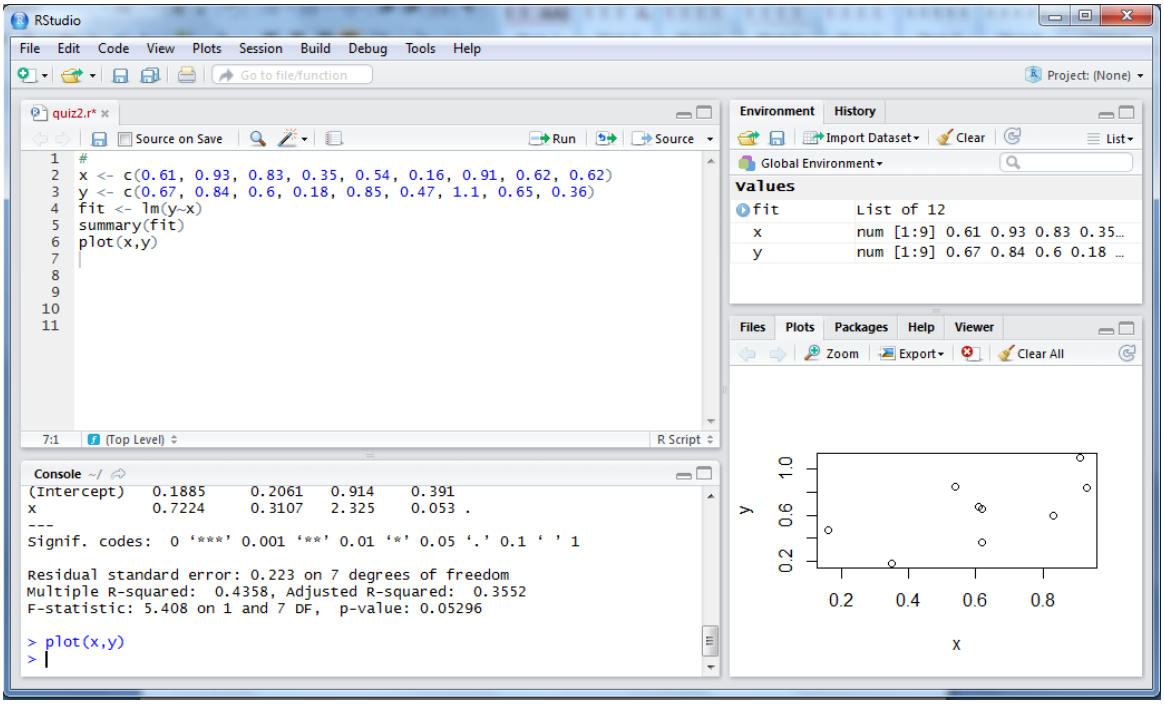
Zip/Tarballs

Zip/tar archives	Size	Date	MD5
RStudio 0.98.1091 - Windows XP/Vista/7/8	63 MB	2014-11-06	a7ed01c134009993db181deaa0e10438

5.-guardamos el ejecutable en el directorio que queramos por ejemplo en el escritorio y lo ejecutamos.

2.3 ENTORNO DE TRABAJO CON RSTUDIO:

Primeramente vamos a comenzar por describir someramente lo que tenemos en pantalla. Aparte de la línea de menús en pantalla aparecen cuatro ventanas: Ficheros abiertos (superior izquierda), Workspace-History (superior derecha), Console o ventana de comandos (inferior izquierda) y Files-Plots-Packages-Help (inferior derecha).



Nosotros vamos a introducir los comandos en Console y los resultados, bien se obtendrán en la misma ventana bien en la que está justo encima, a no ser que sea un gráfico que aparecerá en la sección Plot de la ventana inferior derecha.

Si necesitamos ayuda podemos ir directamente a la sección de ayuda (Help) del menú de opciones para resolver cuestiones sobre R o Rstudio y en la pestaña Help de la ventana inferior derecha es específica de R. Otra forma muy rápida consiste es ecribir ?*plot()* en la Consola y nos saca la ayuda de la función plot en la ventana inferior derecha.

2.4 DIRECTORIO DE TRABAJO

Se trata del directorio donde habitualmente vamos a trabajar. Cuando llamemos a R para abrir cierto fichero lo intentará buscar en ese directorio de trabajo. Yo recomiendo que cada proyecto tenga su propio directorio de trabajo. Por ejemplo para este curso de introducción a R podemos crearnos nuestro propio directorio y llamarlo “C:/R/R pa tontos”.

Antes de comenzar a trabajar con R ponemos el siguiente comando en la ventana de comandos:

```
> Setwd(“C:/R/R pa tontos”)
```

Y me da un error. ¡SO TONTO! ten cuidado con la mayúsculas y las minúsculas 

Prueba de nuevo con:

```
> setwd(“C:/R/R pa tontos”)
```

Si directamente lo hacemos en RStudio, nos vamos a Tools/Set working directory.

Si queremos conocer en que directorio estamos tecleamos:

```
> getwd()
```

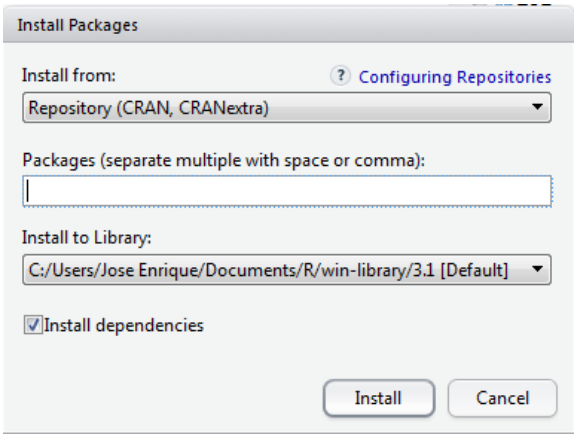
2.5 LIBRERIAS

R está preparado para realizar infinidad de análisis por lo que trabaja con librerías que se cargan a gusto del programador. En la actualidad (diciembre 2014) hay más de 6000 librerías “packages” y esto sigue creciendo.

Existen una serie de librerías que ya están cargadas por defecto y son las de uso mas común. Es posible que a la hora de cargar una librería nos de algún error porque dependa de otra que aún no esté cargada. Se carga la que falte y listo.

Por ejemplo vamos a cargar la librería “geometry”

En la línea de menús vamos a “Tools”->”install packages” y nos sale el siguiente menú:

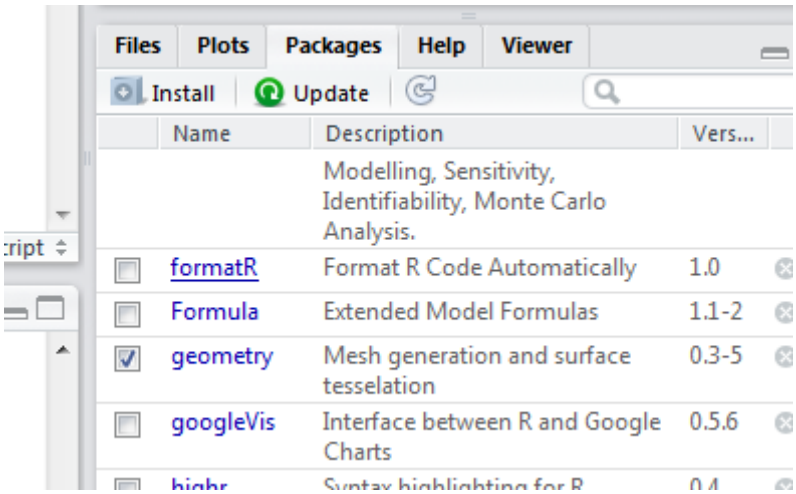


Tecleamos el nombre “geometry” en packages y pulsamos el botón “Install”.

Para cargar la librería o bien tecleamos en la consola:

```
> library("geometry")
```

O nos vamos a la ventana “Files-Plots-Packages-Help” y en la pestaña “packages” buscamos la librería y marcamos



3. PRIMEROS EJEMPLOS CON COMANDOS DE R

Vamos a comenzar escribiendo el típico “Hola Mundo” de todos los inicios de programación con un lenguaje nuevo. Para ello vamos a Console y escribimos:

```
> hola <- function() {cat("¡Hola Mundo!\n")}
> hola()
¡Hola Mundo!
```

Con esto (la primera línea) lo que hacemos es crear un nuevo objeto “hola” que es de tipo función y al que le asignamos una tarea (lo que hay ente llaves) y al ejecutarlo (segunda línea) se le dice a R que realice la tarea asociada a dicha función, es decir, que escriba ¡Hola Mundo!

3.1 CALCULADORA

Vamos a ver cómo usar R como una calculadora básica. En la ventana de comandos podemos utilizar R de la siguiente forma:

```
> 10^2 + 7
[1] 107
```

3.2 ESPACIO DE TRABAJO “WORKSPACE”

R trabaja principalmente con variables que pueden ser utilizadas más tarde la forma de asignación es a través de los simbolos “=” , “<-“ o “->” veamos varios ejemplos

```
> a<-1
> a
[1] 1
> b=2
> b
[1] 2
> 3->c
> c
[1] 3
```

Algunas operaciones simples con variables

```
> a=1
> b=2
> c=3; d=4
> a+b+c+d
[1] 10
> a-d
[1] -3
> c/d
[1] 0.75
> a
[1] 1
> b
[1] 2
> 2 * pi
[1] 6.283185
```

Para borrar objetos (variables) del espacio de trabajo (rm : Iniciales de remove) basta con escribir en la consola o ventana de comandos:

```
> rm(list=ls())
> a
Error: object 'a' not found
```

Si luego tecleamos el nombre de una variable nos da un error

Para ver los nombres de los objetos presentes en el espacio de trabajo R

```
>ls()
```

o bien

```
>objects()
```

Se pueden recuperar líneas de instrucciones introducidas anteriormente pulsando la tecla con la flecha ascendente del teclado, a fin de reejecutarlas o modificarlas.

Para introducir comentarios se utiliza el símbolo #. Desde el carácter # al fin de línea es un comentario.

Se puede poner más de una instrucción en una única línea poniendo el carácter ";"

3.3 ESCALARES VECTORES Y MATRICES

Lo mismo que otros programa R Organiza los números en escalares (un simple número 0-Dimensional) vectores (una fila de números, dimensión 1) o en matrices (2 dimensiones o mas)

Se pueden construir vectores. Construyamos un vector de nombre v y escribimos su valor en la ventana de R Console. La expresión c() significa el conjunto de valores dados entre paréntesis.

```
> v=c(1,2,3,4,5,6);v
[1] 1 2 3 4 5 6
```

En ocasiones dentro de una secuencia de datos ocurre que hay huecos, es decir, que faltan algunos y se desea reflejar dicha ausencia. El entorno R lo hace con el texto <NA> (Non Available, No Accesible). Y tiene mecanismos para gestionar su presencia. Citemos dos y su propio uso explica el funcionamiento:

```
> v1=c(1,2,NA,4,5,6);v1
[1] 1 2 NA 4 5 6
> is.na(v1) # función is.na
[1] FALSE FALSE  TRUE FALSE FALSE FALSE
```

Para ver la longitud del vector x

```
> length(v1)
[1] 6
```

Para construir una matriz de 8 números, en 2 filas y 4 columnas:

```
> M=array(c(1,2,3,4,5,6,7,8),dim=c(2,4)) ; M # la matriz se va
llenando por columnas
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
```

```
[2,]    2    4    6    8
```

Para acceder al elemento sub 2-3 de M, que vale 1:

```
> M[2,3]
[1] 6
> M[2,3:4]
[1] 6 8
> M[1,c(1,3,4)]
[1] 1 5 7
```

La fila 1:

```
> M[1,]
[1] 1 3 5 7
```

La columna 4:

```
> M[,4]
[1] 7 8
> M[2,c(2,4)]
[1] 4 8
```

```
> M[2,c(2,4)]=c(17,18);M
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2   17    6   18
```

Número de filas:

```
> nrow(M)
[1] 2
```

Número de columnas:

```
> ncol(M)
[1] 4
```

Para ver las dimensiones de una matriz:

```
> dim(M)
[1] 2 4
```

Para definir una matriz de ‘ceros’ y otra de con un mismo valor, 1, por ejemplo:

```
> Z=array(0,c(4,3));Z;U=array(1,c(4,3));U
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
[3,]    0    0    0
[4,]    0    0    0
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
[3,]    1    1    1
[4,]    1    1    1
```

4 AYUDA Y DOCUMENTACIÓN

R dispone de una ayuda muy completa sobre todas las funciones, procedimientos y elementos que configuran el lenguaje.

El entorno R ofrece ayuda al usuario mediante el menú Ayuda en la barra de la ventana R Console o en otras ventanas específicas. También se puede buscar ayuda mediante comandos:

```
> ?help
> help(rnorm)
> help(package="deSolve")
> help("for") # Para instrucciones especiales como for se ponen
comillas
> ?summary
> ?mean
```

A continuación presento una serie de links muy interesantes para ayuda en R:

• http://cran.r-project.org/doc/manuals/.

• http://cran.r-project.org/doc/contrib/

• http://zoonek2.free.fr/UNIX/48_R/all.html

• http://rwiki.sciviews.org/doku.php.

• http://www.statmethods.net/.

• http://mathesaurus.sourceforge.net/

Otros libros

- Chambers (2008). *Software for Data Analysis*, Springer.
- Chambers (1998). *Programming with Data*, Springer.
- Venables & Ripley (2002). *Modern Applied Statistics with S*, Springer.
- Venables & Ripley (2000). *S Programming*, Springer.
- Pinheiro & Bates (2000). *Mixed-Effects Models in S and S-PLUS*, Springer.
- Murrell (2005). *R Graphics*, Chapman & Hall/CRC Press.
- Springer has a series of books called *Use R!*.

Un listado más extenso de libros lo podemos encontrar en

http://www.r-project.org/doc/bib/R-books.html

y en español:

- "R para Principiantes", Jorge A. Ahumada
- "Introducción a R" Andrés González and Silvia González
- "Gráficos Estadísticos con R" J. C. Correa, Nelfi González
- "Cartas sobre Estadística de la Revista Argentina de Bioingeniería" Marcelo R. Risk.
- "Introducción al uso y programación del sistema estadístico R" Ramón Díaz-Uriarte
- "Generacion automática de reportes con R y LaTeX" Mario Alfonso Morales Rivera.
- "Metodos Estadisticos con R y R Commander" Antonio Jose Saez Castillo.
- "Optimización Matemática con R: Volumen I" E. G. Baquela A. Redchuk
- "Introducción al uso de R y R Commander para el análisis estadístico de datos en ciencias sociales" by Rosario Collatón Chicana.

5 MÓDULO DE PROGRAMA O "SCRIPT"

Un conjunto de instrucciones del lenguaje R se pueden integrar en un archivo texto para formar un módulo de programa (script), que se puede abrir, editar y ejecutar utilizando el menú Archivo en R Console y también en RStudio. R reconoce los archivos texto con la extensión .R.

Para operaciones que requieran varias instrucciones consecutivas, resulta útil el trabajar con un fichero de comandos editable (script)

R proporciona por defecto la posibilidad de trabajar con scripts como ventanas del propio programa.

Podemos abrir un script nuevo des de el menú Archivo/Nuevo script. Desde ellos podemos ejecutar con CTRL + r los comandos de la línea en la que estamos, o el bloque de conamdos que tengamos seleccionado.

Si hemos generado un script que le hemos denominado “prueba.r” se puede ejecutar todo el srcipt poniendo:

```
> source("prueba.r")
```

Operadores:

Aritméticos	Comparativos	Lógicos
+ Suma	== igualdad	& Y lógico
- Resta	!= Diferente de	! No lógico
* Multiplicación	< Menor que	| O lógico
/ División	> Mayor que	
^ Potencia	<= Menor o igual	
%/% División entera	>= Mayor o igual	

Funciones:

Raíz cuadrada de x	*sqrt(x)*	Media	*mean(x)*
Exponencial de x	*exp(x)*	Desv. Típica	*sd(x)*
Logaritmo neperiano	*log(x)*	Varianza	*var(x)*
Nº de elementos de un vector x	*length(x)*	Mediana	*median(x)*
Suma los elementos del vector x	*sum(x)*	Quantiles	*quantile(x,p)*
Seno de x	*sin(x)*	Máximo y Mínimo	*range(x)*
Coseno de x	*cos(x)*	Ordenación	*sort(x)*
Tangente de x	*tan(x)*	Resumen de todos	*summary*

6 ESTRUCTURAS DE DATOS

Anteriormente hemos introducido lo que son los vectores y las matrices vamos a profundizar un poco más en estas estructuras y a introducir alguna nueva.

6.1 OPERACIONES ALGEBRAICAS CON MATRICES:

Vamos a ver cómo funcionan algunas operaciones con matrices

```
> M=array(c(1,2,3,4,5,6,7,8),dim=c(2,4))
> M
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
> M+3
     [,1] [,2] [,3] [,4]
[1,]    4    6    8   10
[2,]    5    7    9   11
> M*2
     [,1] [,2] [,3] [,4]
[1,]    2    6   10   14
[2,]    4    8   12   16
> Mt=t(M);Mt
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
> M1=array(2.5,c(2,4));Suma=M+M1;Suma
     [,1] [,2] [,3] [,4]
[1,]  3.5  5.5  7.5  9.5
[2,]  4.5  6.5  8.5 10.5
```

Producto de matrices (operador %*% para producto):

```
> Producto=M1%*%array(1,c(4,2));Producto
     [,1] [,2]
[1,]   10   10
[2,]   10   10
> y=c(3,2,1);A=array(c(1,2,3,4,5,6,7,8,9),dim=c(3,3));h=A%*%y;A;h;y
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
     [,1]
[1,]   18
```

```
[2,]   24
[3,]   30
[1] 3 2 1
```

Solución de un sistema de ecuaciones lineales

```
>y=c(3,2,1);A=array(c(1,2,3.5,4,5.8,6,7,8,9),dim=c(3,3));h=A%*%y;A;h;
y
     [,1] [,2] [,3]
[1,]  1.0  4.0    7
[2,]  2.0  5.8    8
[3,]  3.5  6.0    9
     [,1]
[1,] 18.0
[2,] 25.6
[3,] 31.5
[1] 3 2 1
> sol=solve(A,h);sol
     [,1]
[1,]    3
[2,]    2
[3,]    1
> solve(A);solve(A)%*%A
           [,1]       [,2]       [,3]
[1,] -0.3021583 -0.4316547  0.6187050
[2,] -0.7194245  1.1151079 -0.4316547
[3,]  0.5971223 -0.5755396  0.1582734
             [,1]          [,2]          [,3]
[1,] 1.000000e+00 -1.332268e-15 -1.776357e-15
[2,] 2.220446e-16  1.000000e+00  0.000000e+00
[3,] 1.110223e-16  4.440892e-16  1.000000e+00
```

6.2 LA ESTRUCTURA DATA.FRAME (MARCO DE DATOS)

Es una lista cuyas componentes deben ser vectores (numéricos, carácter, lógicos), factores, matrices numéricas, listas u otros data.frames. Si son vectores, deben ser todos de igual longitud, y si matrices, con el mismo número de filas.

Es frecuente estructurar un data.frame como un conjunto de vectores columna de igual longitud. Podemos imaginar el data.frame como una matriz rectangular, cuyas columnas son las variables del data.frame, que pueden ser de diferente tipo, numéricas, factores, lógicas.

Para construir una variable de tipo data.frame se puede utilizar la función data.frame(). Veamos un ejemplo de data.frame con 2 columnas, a partir de un vector numérico v, que será una columna de nombre valor y otro vector que será otra columna de nombre caso, formada con datos cualitativos.

```
> v=c(6.4,3.5,8.4,5.5,8.3);v
[1] 6.4 3.5 8.4 5.5 8.3
> BBB=data.frame(valor=v,abm=c("alto","alto","bajo","medio","medio"))
> BBB
  valor   abm
1   6.4  alto
2   3.5  alto
3   8.4  bajo
4   5.5 medio
5   8.3 medio
```

Nos referimos separadamente a cada columna separando con el signo $ el nombre del data frame y el de la columna (que es como la denominación de la variable asociada).

```
> BBB$valor;BBB$abm
[1] 6.4 3.5 8.4 5.5 8.3
[1] alto  alto  bajo  medio medio
Levels: alto bajo medio
```

Para ver la dimensión de un objeto, en este caso el data.frame BBB, se puede emplear la función dim:

```
> dim(BBB)
[1] 5 2
```

La función de R summary() aplicada al data frame nos da el resumen estadístico de sus variables (columnas)

```
> summary(BBB)
     valor          abm
 Min.   :3.50   alto :2
 1st Qu.:5.50   bajo :1
 Median :6.40   medio:2
 Mean   :6.42
 3rd Qu.:8.30
 Max.   :8.40
```

Veamos algunas instrucciones para acceder a la información del data.frame. Se puede entender que se trabaja con el data.frame como si fuese una matriz con filas y columnas:

```
> BBB[1,2]
[1] alto
Levels: alto bajo medio
> BBB[0,]
[1] valor abm
<0 rows> (or 0-length row.names)
> BBB[1,]
  valor  abm
1   6.4 alto
> DatosEstruc[1:3,]
  valor  caso
1   2.1  alto
2   1.4  bajo
3   6.0 medio
> DatosEstruc[1:3,2]
[1] alto  bajo  medio
Levels: alto bajo medio
```

Ordenamos en orden creciente según los valores de la variable/columna BBB$valor, con la función sort.list() que devuelve un vector de índices de los datos ordenados.

```
> BBB_orden=sort.list(BBB$valor);BBB_orden
[1] 2 4 1 5 3
```

Construimos un data.frame a partir del anterior , ordenado según los valores crecientes obtenidos para la columna valor. Simplemente se aplica al data.frame BBB el vector de índices BBB_nuevo

```
> BBB_Nuevo=BBB[BBB_orden,];BBB_Nuevo
  valor   abm
2   3.5  alto
4   5.5 medio
1   6.4  alto
5   8.3 medio
3   8.4  bajo
```

Ordenación con un vector de números: Se define un vector de números

```
> v=c(2.1,1.4,6,3.5,8);v
[1] 2.1 1.4 6.0 3.5 8.0
```

7 GRAFICOS

Anteriormente hemos introducido lo que son los vectores y las matrices vamos a profundizar un poco más en estas estructuras y a introducir alguna nueva.

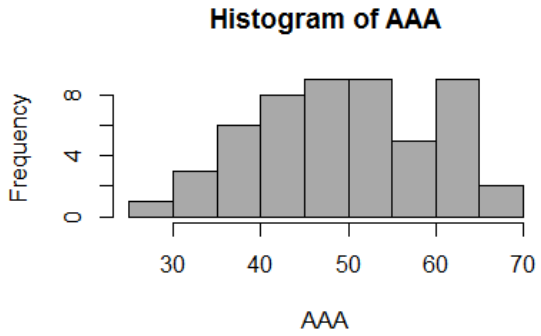
7.1 HISTOGRAMA

En la gráfica cuantitativa el número rectángulos se puede fijar o dejar al programa de forma automática. El eje Y cuenta las frecuencias, porcentajes, densidades, observadas en la muestra

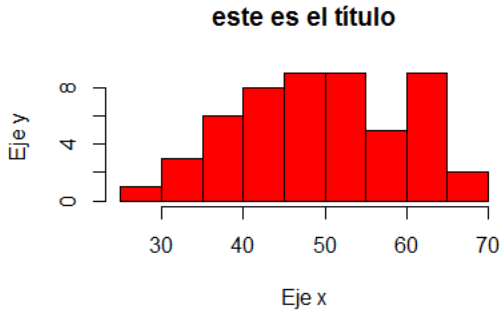
```
> AAA <-c(60.2, 43.3, 51.2, 46.6, 32.5, 41.8, 45.9, 60.6, 32.3,
       31.7, 39.4, 41.2, 60.2, 49.0, 40.5, 58.3, 42.7, 61.4,
       26.0, 53.3, 58.7, 46.4, 39.1, 63.9, 51.5, 53.3, 41.6,
       54.9, 55.2, 60.2, 47.3, 39.8, 46.8, 64.4, 57.9, 39.1,
       44.8, 65.3, 69.7, 50.4, 54.2, 39.4, 46.6, 55.8, 53.6,
       61.8, 44.3, 48.5, 53.9, 61.4, 38.1, 47.8)
```

La instrucción R:

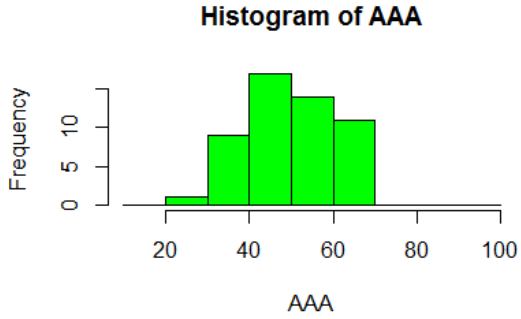
```
>hist(AAA, scale="density",breaks="Sturges",col="darkgray")
```




```
>hist(AAA,col="red", xlab="Eje x",  ylab="Eje y",main="este es el título")
```



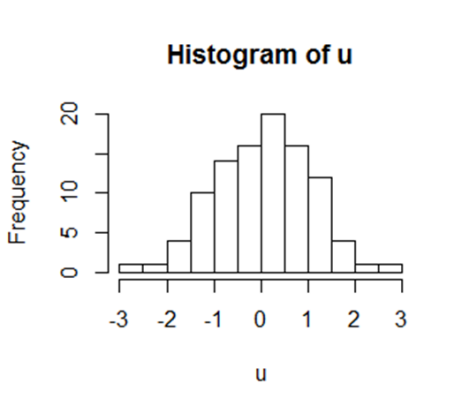
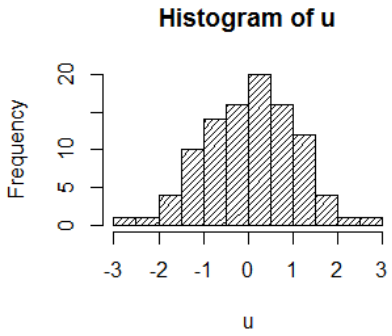
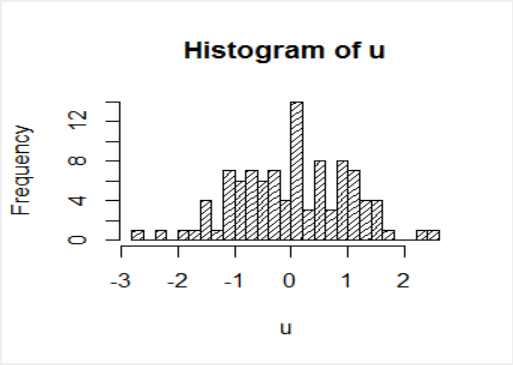

```
>hist(AAA,
scale="density",breaks=seq(from=10,to=100,by=10),col="green")
```

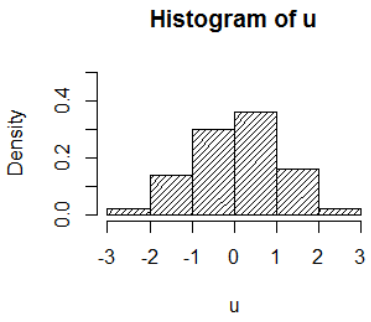


Veamos más ejemplos sobre Histogramas

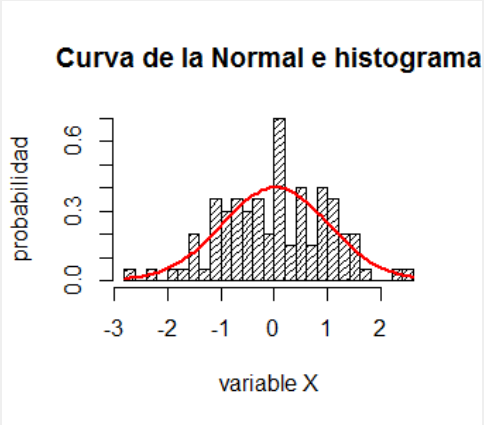
```
set.seed(121343)
```

```r
u <- rnorm(100)
```

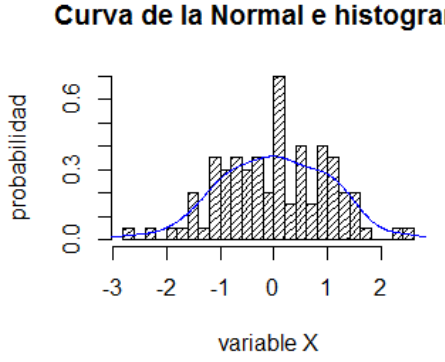
	`#histograma por defecto` `hist(u)`
	`#cambiando en relleno` `hist(u, density=20)`
	`#Identificando el n° de columnas` `hist(u, density=20, breaks=20)`



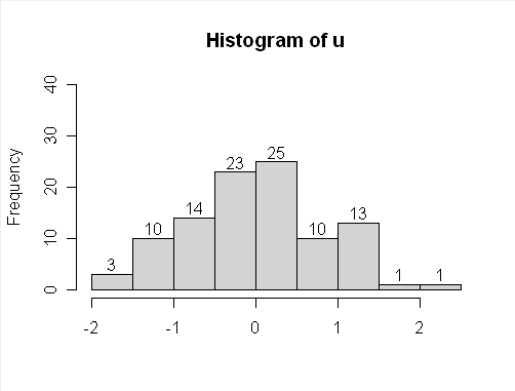

```
hist(u, density=20, breaks=-3:3,
     ylim=c(0,.5), prob=TRUE)
```




```
m<-mean(u)
std<-sqrt(var(u))
hist(u, density=20, breaks=20,
prob=TRUE, xlab="variable X",
ylab="probabilidad",ylim=c(0, 0.7),
main="Curva de la Normal e
histograma")
curve(dnorm(x, mean=m,sd=std),
col="red", lwd=2, add=TRUE)
```




```
lines(density(u), col = "blue")
```




```
# en este caso escribe el n° de
# datos que hay en cada intervalo
aa<-hist(u)
plot(aa, ylim=c(0, 40),
col="lightgray",
    xlab="", main="Histogram of u")
text(aa$mids, aa$counts+2,
label=c(aa$counts))
```

7.2 GRÁFICA DE TALLOS Y HOJAS...

Permite la descripción de los datos agrupados en filas y columnas donde recuenta la frecuencia hasta la fila donde se encuentra la mediana, señalada por medio de paréntesis ():

Instrucción R y resultado:

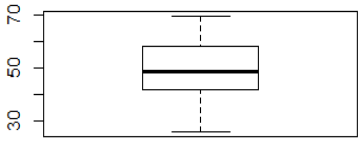
```
> stem.leaf(AAA)
1 | 2: represents 12
 leaf unit: 1
            n: 52
   1    2. | 6
   4    3* | 122
  10    3. | 899999
  18    4* | 01112344
  (9)   4. | 566667789
  25    5* | 011333344
  16    5. | 55788
  11    6* | 000011134
   2    6. | 59
```

7.3 DIAGRAMA DE CAJAS

El diagrama de caja (box plot) consiste en una caja cuyos bordes inferior y superior son los cuartiles 1º y 3º y la línea central representa la mediana. Los bigotes desde la caja indican el rango de los datos. Además de elegir una variable cuantitativa, numérica, permite considerar un factor para comparar la primera respecto de los niveles del factor.

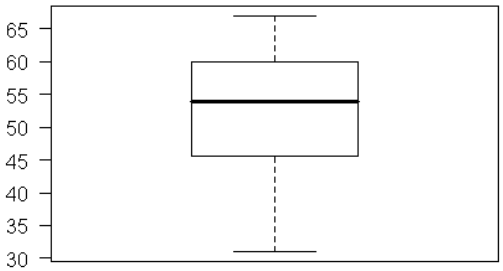
La instrucción R:

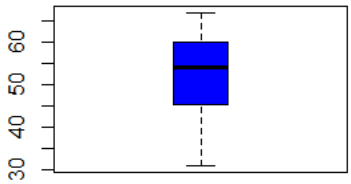
```
> boxplot(AAA)
```



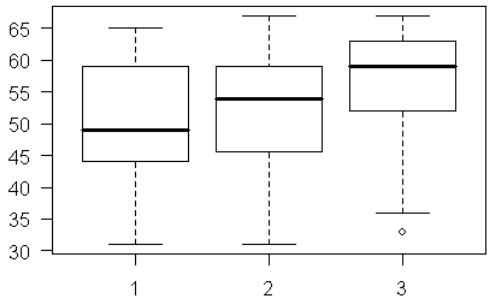
Veamos más ejemplos de este tipo de datos

```
BP <- read.table('http://www.ats.ucla.edu/stat/r/modules/hsb2.csv',
header=T, sep=",")
```

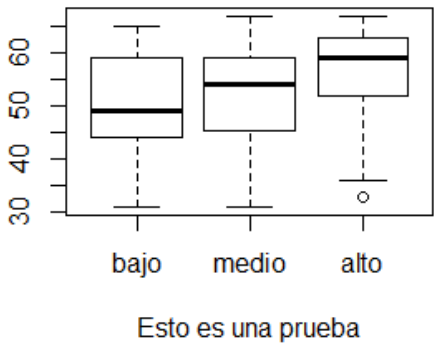
	`boxplot(write)`



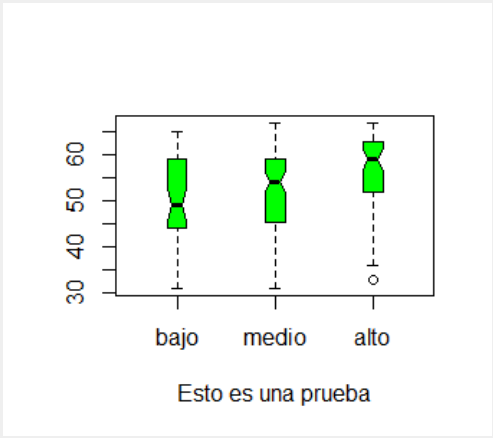

```
boxplot(write,xlab="aaa", boxwex=.4,
+          col="blue")
```



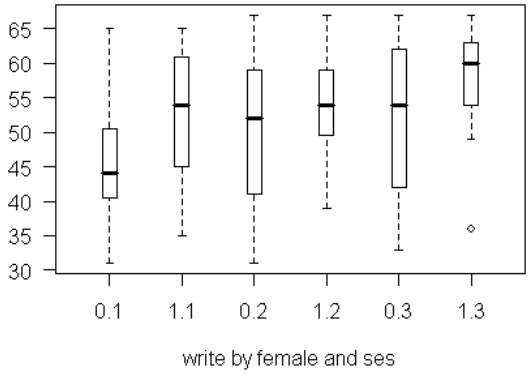

```
boxplot(write ~ ses)
```




```
nom<-as.vector(c("bajo","medio",
"alto"))
sesf<-factor(ses, label=nom)
boxplot(write ~ sesf, xlab="Esto
es una prueba")
```




```
boxplot(write ~ sesf, xlab="Esto
es una prueba",
+     boxwex=.2, notch = TRUE,
+     col = "green")
```




```
boxplot(write ~ female + ses,
xlab="write by female and ses",
boxwex=.2)
```

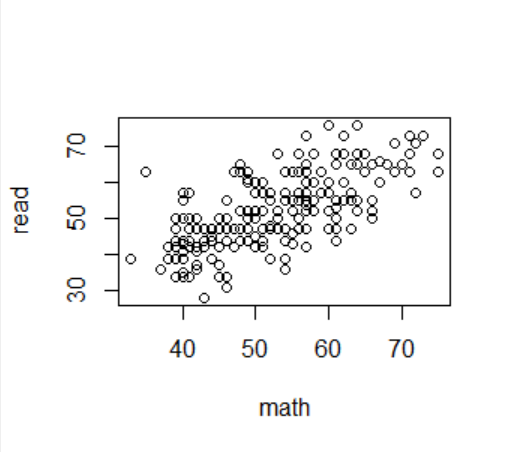
7.4 DIAGRAMA DE DISPERSIÓN

Muestra conjuntamente datos de dos variables (en X y en Y) para ver su correlación, y permite considerar grupos (niveles de un factor)

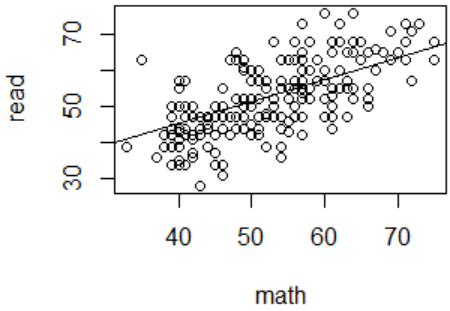
Para hacer las pruebas vamos a utilizar datos genéricos provenientes de “University of California, Los Angeles”

'http://www.ats.ucla.edu/stat/r/modules/hsb2.csv'

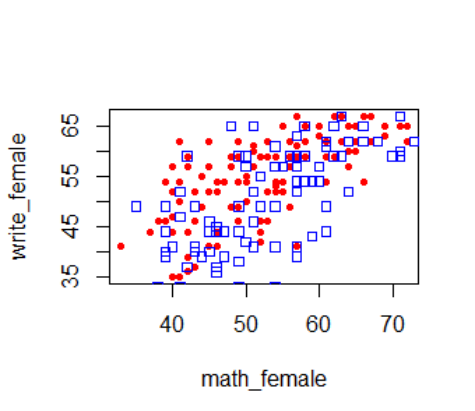
```
hsb2 <- read.table('http://www.ats.ucla.edu/stat/r/modules/hsb2.csv',
header=T, sep=",")
attach(hsb2)
```



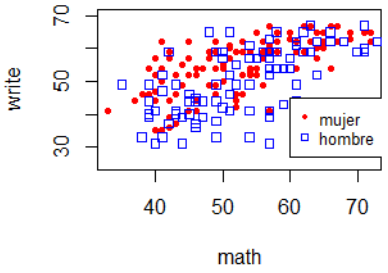

```
plot(math, read)
```



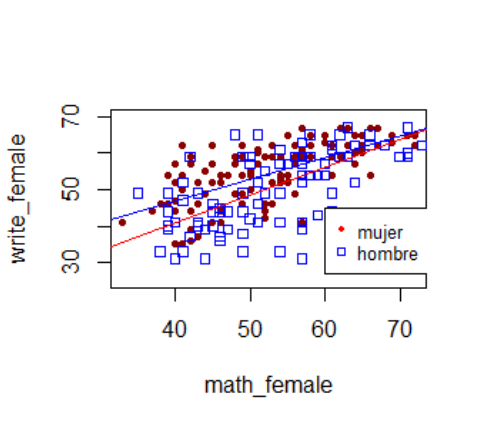

```
plot(math, read)

abline(lsfit(read, math))
```




```
math_male<-hsb2$math[female==0]
write_male<-hsb2$write[female==0]
math_female<-hsb2$math[female==1]
write_female<-hsb2$write[female==1]

plot(math_female, write_female,
type="p",
     pch=20, col="red")
points(math_male, write_male,
pch=22, col="blue")
```




```
Añadiendo leyenda


hsb2_female<-hsb2[female==1,]
> hsb2_male<-hsb2[female==0,]
> with(hsb2_female, plot(math, write,
pch=20,
+ col="red", ylim=c(25, 70)))
> with(hsb2_male, points(math, write,
pch=22,
+ col="blue"))
> legend(60, 45, c("mujer", "hombre"),
 pch=c(20, 22),
+ cex=.8, col=c("red", "blue"))
```




```
math_male<-hsb2$math[female==0]
> write_male<-hsb2$write[female==0]
> math_female<-hsb2$math[female==1]
> write_female<-hsb2$write[female==1]
> plot(math_female, write_female, type
="p", pch=20,
+ col="darkred", ylim=c(25, 70))
> points(math_male, write_male, pch=22
, col="blue")
> abline(lsfit(write_female, math_fema
le), col="red")
> abline(lsfit(write_male, math_male),
 col="blue")
> legend(60, 45, c("mujer", "hombre"),
 pch=c(20, 22),
+ cex=.8, col=c("red", "blue"))
```

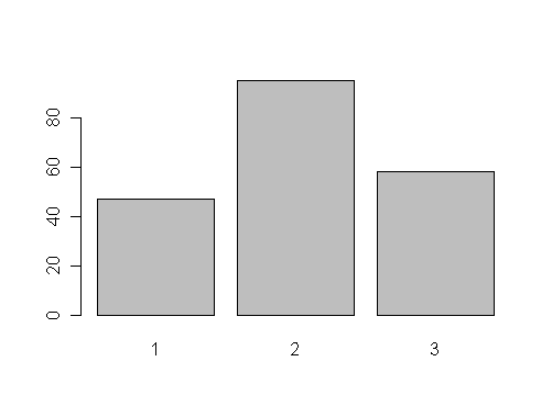
7.5 GRÁFICA DE BARRAS

Gráfico usado para recoger las frecuencias de los niveles en las variables cualitativas (factores)

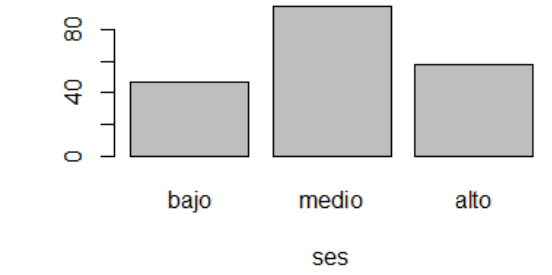
Lo mismo que en el caso anterior para hacer las pruebas vamos a utilizar datos genéricos provenientes de “University of California, Los Angeles”

'http://www.ats.ucla.edu/stat/r/modules/hsb2.csv'

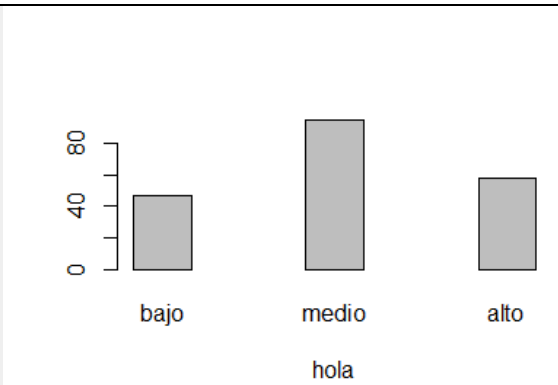
```
barras <-
read.table('http://www.ats.ucla.edu/stat/r/modules/hsb2.csv',
header=T, sep=",")

attach(barras)
```



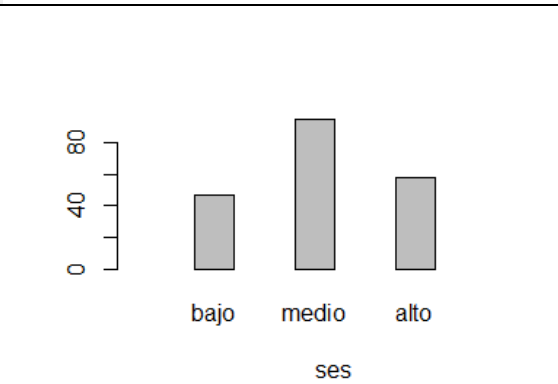

```
a<-table(ses)

barplot(a)
```



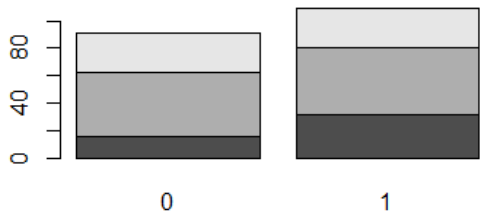

```
seslab<-
as.vector(c("bajo","medio",
"alto"))

fses<-factor(ses, label=seslab)

sesstat<-table(fses)

barplot(sesstat, xlab="ses")
```



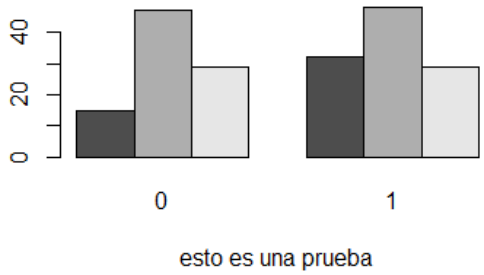

```
barplot(sesstat, space=2,
xlab="hola")
```




```
barplot(sesstat, xlim=c(0,5),
space=1.5, width=.5, xlab="ses")
```




```
barras2<-table(ses, female)
barplot(barras2)
```




```
b<-table(ses, female)
barplot(b, beside=T, xlab="esto
es una prueba")
```

7.6 GRÁFICA DE SECTORES

Se trata de un diagrama en forma de círculo dividido en tantos sectores como datos distintos haya, en el que el ángulo de cada sector es proporcional a la frecuencia relativa del correspondiente dato.

Esta representación gráfica se denomina diagrama de sectores o diagrama de tarta. También puede usarse para datos cuantitativos agrupados en clases, y en tales casos, cada sector corresponde a una clase.

La función para diseñar diagramas de sectores en R es: pie()

Por ejemplo, la encuesta de población activa elaborada por el Instituto Nacional de Estadística referente al cuarto trimestre de 1970, presenta para el número de empleados por rama de actividad los siguientes datos:

Rama de Actividad	Miles de Empleados
Agricultura, caza y pesca	3706.3
Fabriles	3437.8
Construcción	1096.3

Comercio	1388.3
Transporte	648.7
Otros servicios	2454.8

Para almacenarlos en R:

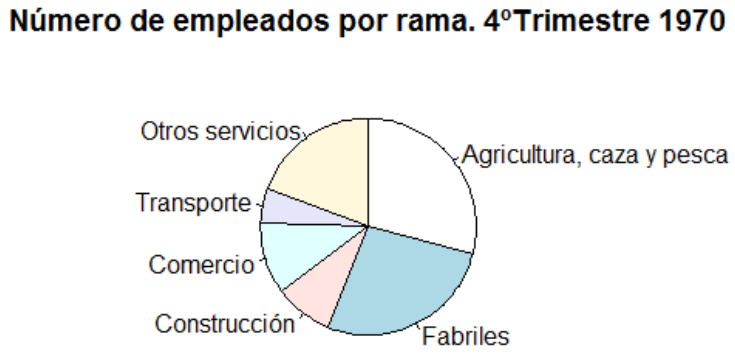
```
> Sector <- c(3706.3, 3437.8, 1096.3, 1388.3, 648.7, 2454.8)
```

Y posteriormente, le asignaremos la Rama de Actividad al vector Sector mediante la función names():

```
>names(Sector) <- c("Agricultura, caza y pesca", "Fabriles",
"Construcción", "Comercio", "Transporte", "Otros servicios")
```

Si usamos ahora la función pie() con los datos anteriores obtenemos:

```
pie(Sector, clockwise=TRUE, main="Número de empleados por rama.
4°Trimestre 1970", col=c(2,3,4,5,6,7))
```



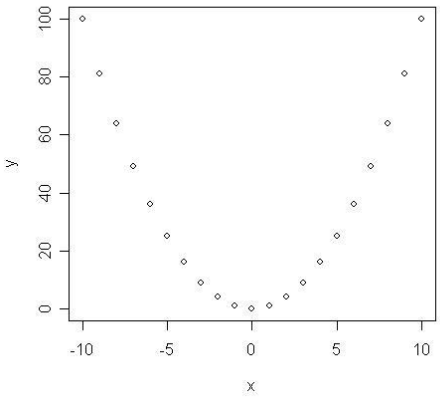
7.7 GRÁFICA XY:

Permite comparar datos de dos variables cuantitativas

La función plot()

El comando plot se utiliza para crear una nueva figura.

```
> x = seq(-10,10)        # Generamos los números -10, -9,....,9, 10
> y = x^2                # Generamos los cuadrados de dichos números
> plot(x,y)              # Graficamos
```

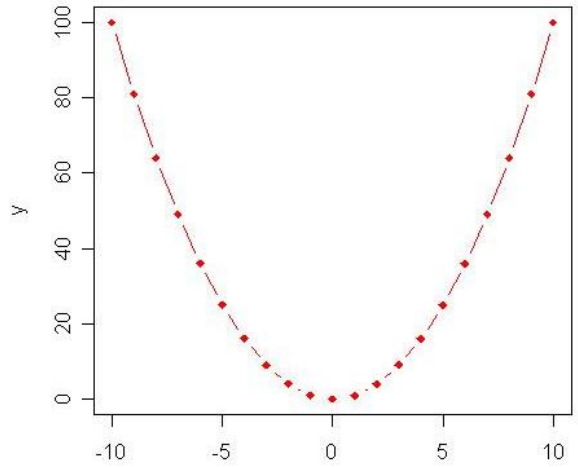


Dicha función admite bastantes argumentos. Vamos a destacar los más importantes:

axes=F	**Suprime la generación de los ejes**
log="x" **log="y"** **log="xy"**	**Hace que alguno de los ejes se tome en escala logarítmica**
type="p" **type="l"** **type="b"** **type="o"** **type="h"** **type="s"** **type="S"** **type="n"**	**Dibuja puntos individuales (opción por defecto)** **Dibuja líneas** **Dibuja puntos y líneas** **Dibuja puntos atravesados por líneas** **Dibuja con líneas verticales** **Dibuja a base de funciones escalera** **Casi lo mismo** **No dibuja nada. Pero deja marcados los puntos para manejos posteriores**
xlab="cadena" **ylab="cadena"** **main="cadena"** **sub="cadena"**	**Etiqueta para el eje de las x** **Etiqueta para el eje de las y** **Título del gráfico** **Subtítulo del gráfico**
pch="simbolo" **pch=18** **pch="x"** **pch="P"**	**Se dibuja con el simbolo especificado. Por ejemplo:**
col= numero entero **col=2** **col=3**	**Color para dibujar** **Color rojo** **Color verde**

Algunos ejemplos:

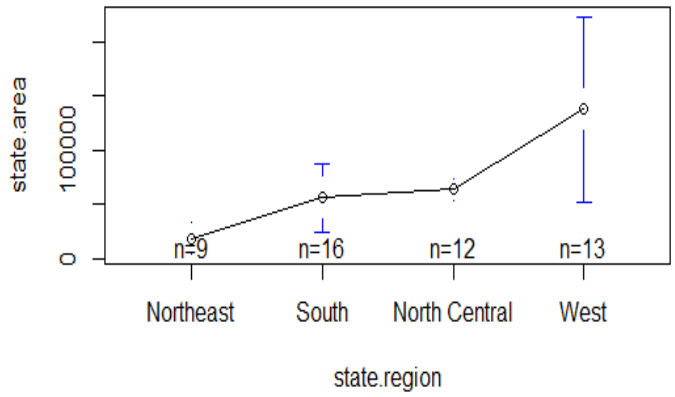
```
> x = seq(-10,10)
> y = x^2
> plot(x,y,type="l",xlab="eje de las x",ylab="eje de las y",
main="Parabola")
> plot(x,y,type="h",xlab="eje de las x",ylab="eje de las y",
main="Parabola",axes=F)
> plot(x,y,pch=18,col=2,type="b")
```



7.8 GRÁFICA DE LAS MEDIAS:

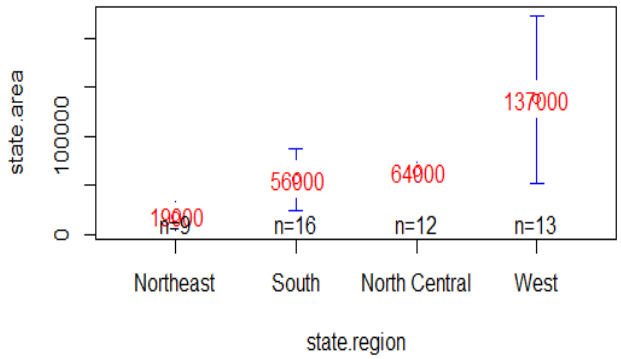
Permite comparar el efecto de los niveles de uno o dos factores en el comportamiento de una variable cuantitativa. Junto a las medias se añade a cada lado una desviación típica muestral, que se ha elegido en las opciones.

```
> data(state)
> plotmeans(state.area ~ state.region)
```



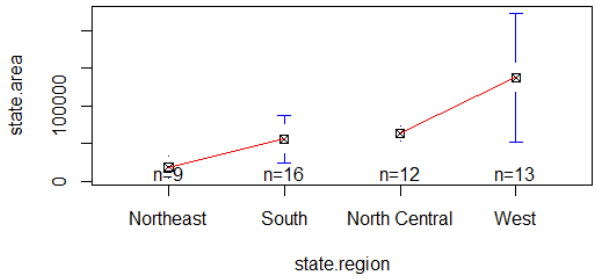
Muestra el valor medio en otro color

```
> plotmeans(state.area ~ state.region, mean.labels=TRUE, digits=-3,
col="red", connect=FALSE)
```



Conexión entre algunos de los valores medios

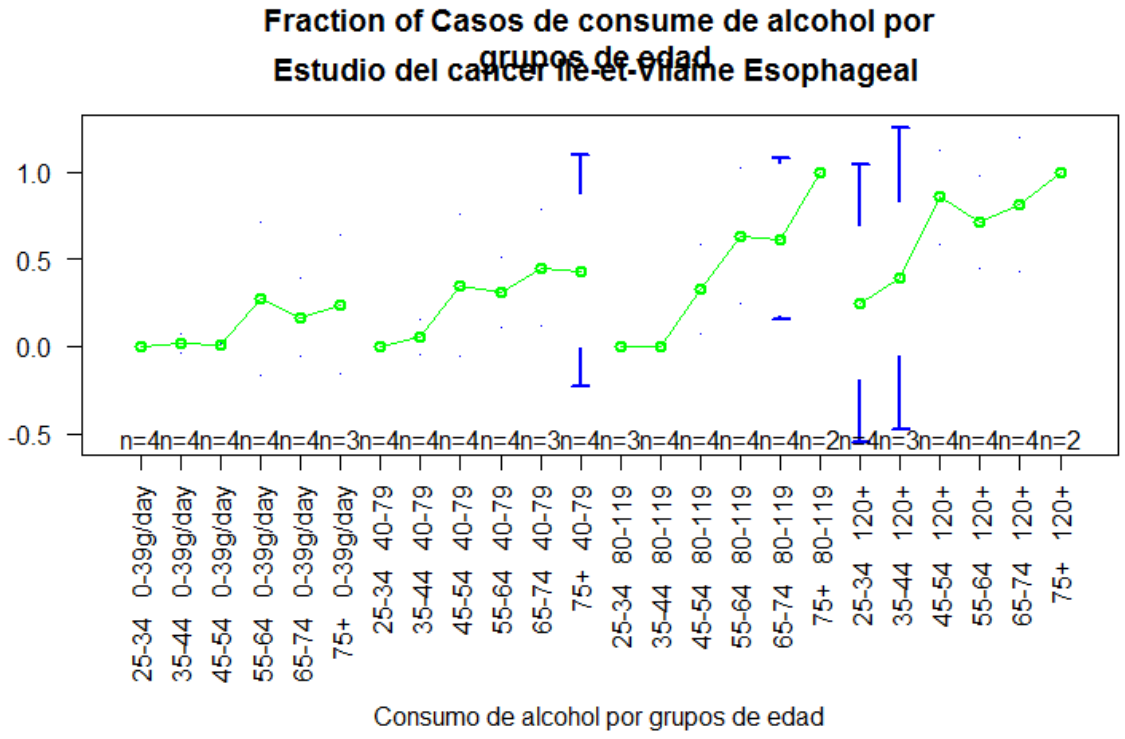
```
>  plotmeans(state.area ~ state.region, connect=list(1:2, 3:4),  ccol="red",
pch=7 )
```



Y un ejemplo más complicado

```
>  data(esoph)
>  par(las=2,                                  # use perpendicular axis
labels
      mar=c(10.1,4.1,4.1,2.1),       # create enough space for long x labels
      mgp=c(8,1,0)                          # move x axis legend down to avoid
overlap
      )
>  plotmeans(ncases/ncontrols ~ interaction(agegp , alcgp, sep ="   "),
```

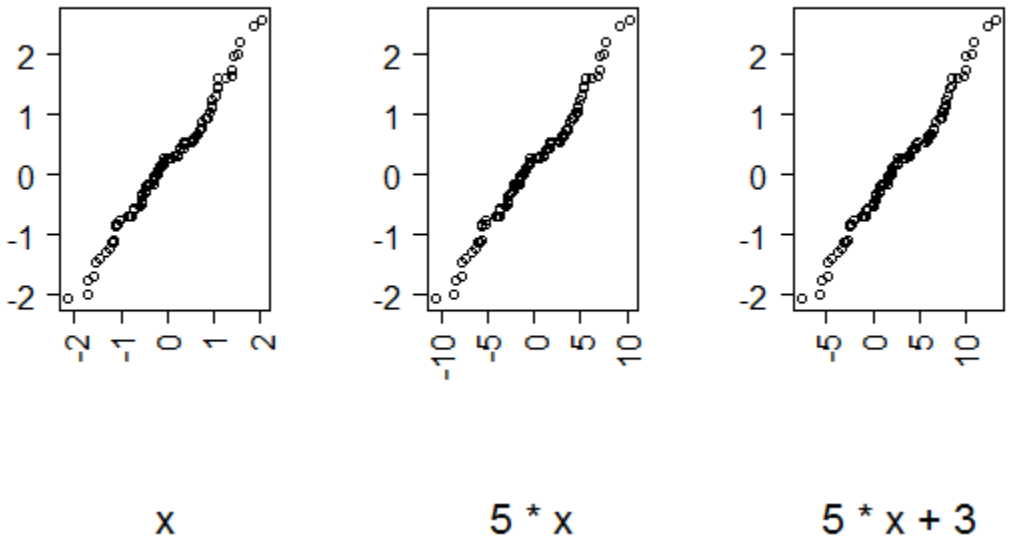
```
            connect=list(1:6,7:12,13:18,19:24),
            barwidth=2,
            col="green",
            data=esoph,
            xlab="Consumo de alcohol por grupos de edad",
            ylab="# Casos / # Controles",
            main=c("Fraction of Casos de consumo de alcohol por
                        grupos de edad ",
                   "Estudio del cancer Ile-et-Vilaine Esophageal ")
            )
abline(v=c(6.5, 12.5, 18.5), lty=2)
```



7.9 MATRIZ DE DIAGRAMA DE DISPERSIÓN (QQ):

En una matriz de gráficas representa por parejas los datos asociados a un conjunto de variables cuantitativas. Extiende los Diagramas de dispersión a más de 2 variables. Permite considerar un factor cualitativo asociado a las variables cuantitativas.

```
x <- rnorm(100)

y <- rnorm(100)

par(mfrow=c(1,3))

qqplot(x,y ,cex.lab=2,cex.axis=1.5)

qqplot(5*x,y,cex.lab=2,cex.axis=1.5 )

qqplot(5*x+3,y,cex.lab=2,cex.axis=1.5)
```

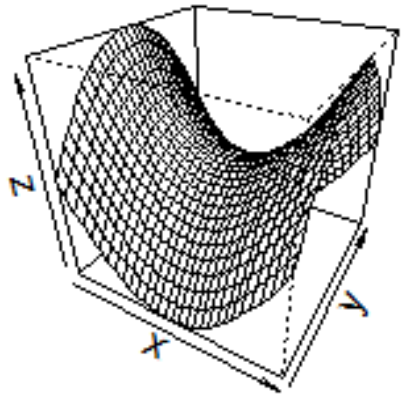


7.10 GRÁFICO 3D

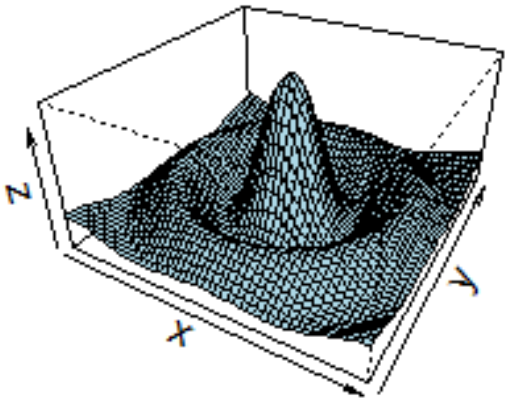
Representa en 3D el comportamiento de una variable explicada (eje vertical) a partir de los datos de otras 2 (ejes horizontales), tomadas como explicativas de la primera. Permite considerar la presencia de un factor. Y representar superficies de regresión, entre ellas el plano de regresión.

Ejemplo de dibujo de una función z=f(x,y):

```
> x = seq(-15,15,length=30)      # Generamos una malla de puntos (x,y)
> y = x
> f = function(x,y) { x^2 - y^2 }# Definimos la función que
dibujaremos
> z = outer(x,y,f)                 # La función outer evalua la
función f en cada punto(xi,yj)
> persp(x,y,z)                     # Un gráfico en perspectiva
> persp(x,y,z,theta=30,phi=30)
```



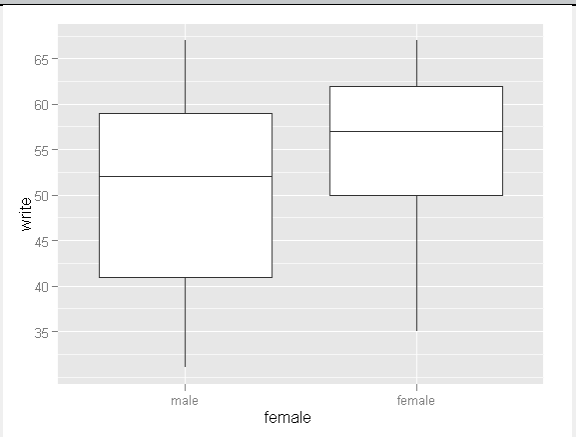

```
x <- seq(-10, 10, length.out = 50)
y <- x
rotsinc <- function(x,y)
{ sinc <- function(x) { y <- sin(x)/x ; y[is.na(y)] <- 1; y }
  10 * sinc( sqrt(x^2+y^2) )
}
sinc.exp <- expression(z == Sinc(sqrt(x^2 + y^2)))
z <- outer(x, y, rotsinc)
oldpar <- par(bg = "white")
persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue")
title(sub=".")## work around persp+plotmath bug
title(main = sinc.exp)
persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue",
      ltheta = 120, shade = 0.75, ticktype = "detailed",
      xlab = "X", ylab = "Y", zlab = "Z")
title(sub=".")## work around persp+plotmath bug
title(main = sinc.exp)
```



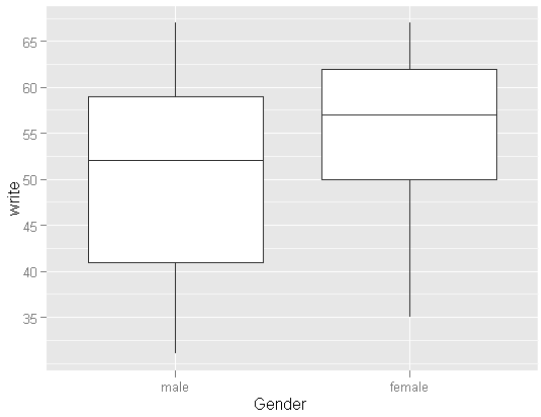
7.11 LIBRERIA GGPLOT2

Existen muchas librerías gráficas en R una de la más extendidas es ggplot2, veamos algunas de las posibilidades

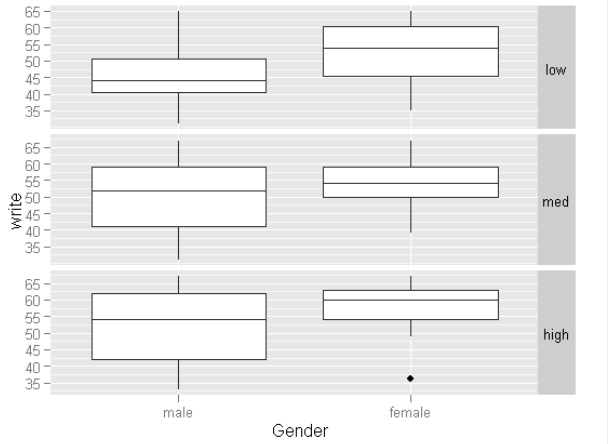
```
library(ggplot2)
hsb2 <- read.table('http://www.ats.ucla.edu/stat/r/modules/hsb2.csv', header=T, sep=",")
hsb2$female <- factor(hsb2$female, labels = c("male", "female"))
hsb2$ses<- factor(hsb2$ses, labels = c("low", "med", "high"))
```



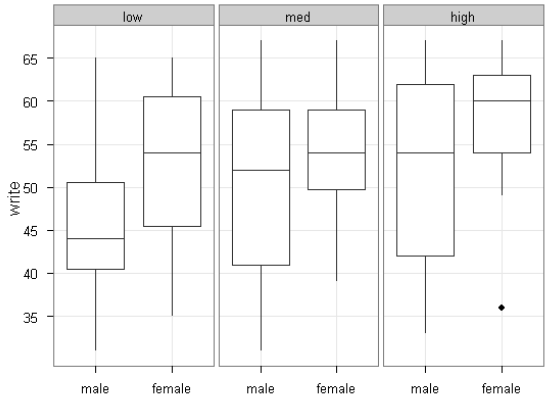

```
qplot(female, write, data = hsb2, geom="boxplot")
```



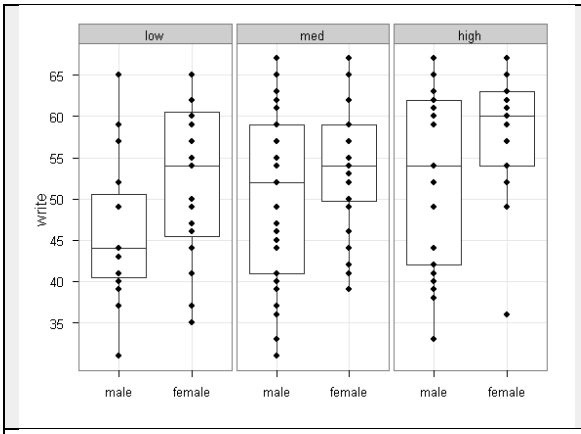

```
qplot(female, write, data = hsb2,
geom="boxplot")+xlab("Gender")
```



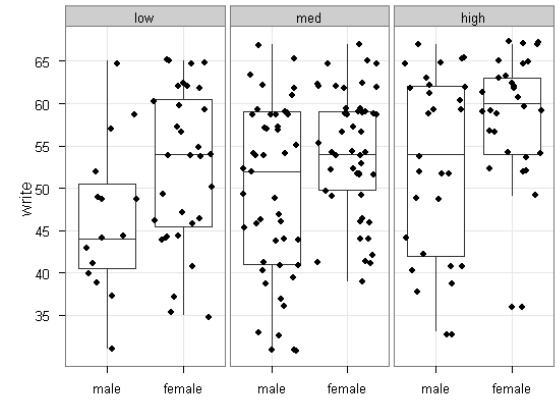

```
qplot(female, write, data = hsb2,
   geom="boxplot")+ xlab("Gender") +
facet_grid(ses~.,
   scales="free", space="free") +
opts(strip.text.y = theme_text())
```



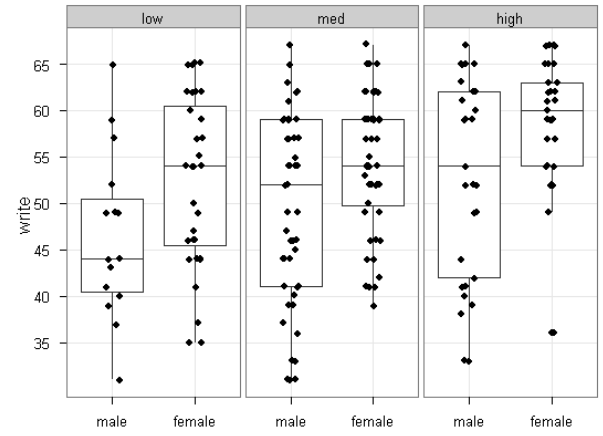

```
qplot(female, write, data = hsb2,
  geom="boxplot")+ xlab("") +
facet_grid(.~ses, scales="free",
  space="free") + theme_bw() +
opts(strip.text.y = theme_text())
```



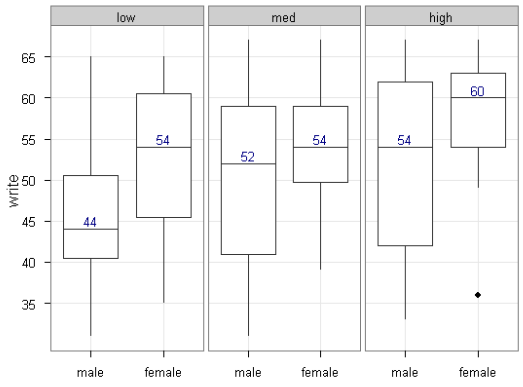

```
qplot(female, write, data = hsb2,
  geom="boxplot")+ geom_point() +
xlab("") + facet_grid(.~ses,
  scales="free",space="free") +
theme_bw() + opts(strip.text.y =
theme_text())
```




```
qplot(female, write, data = hsb2,
  geom="boxplot")+ geom_jitter() +
xlab("") + facet_grid(.~ses,
  scales="free",space="free") +
theme_bw() + opts(strip.text.y =
theme_text())
```




```
qplot(female, write, data = hsb2,
  geom="boxplot")+
geom_jitter(position=position_jitter(w=
0.1,  h=0.1)) + xlab("") +
facet_grid(.~ses, scales="free",
  space="free") + theme_bw()+
opts(strip.text.y = theme_text())
```




```
library(doBy)
a<-summaryBy(write ~ female + ses ,
hsb2, FUN=c(median))
qplot(female, write, data = hsb2,
  geom="boxplot") + xlab("") +
layer(data = a, mapping =
  aes(x = female, y= write.median+1,
label=round(a$write.median)),
  geom = "text", color="NavyBlue",
size=3.5) +facet_grid(.~ses,
  scales="free", space="free") +
theme_bw()+ opts(strip.text.y =
theme_text())
```

7.12 GRÁFICOS DE BAJO NIVEL

Los comandos de bajo nivel sirven para añadir información extra a los gráficos que producen los comandos de alto nivel. Por ejemplo, podríamos querer añadir texto a un gráfico, puntos extras, líneas, cosas así. Entre los más importantes podemos destacar:

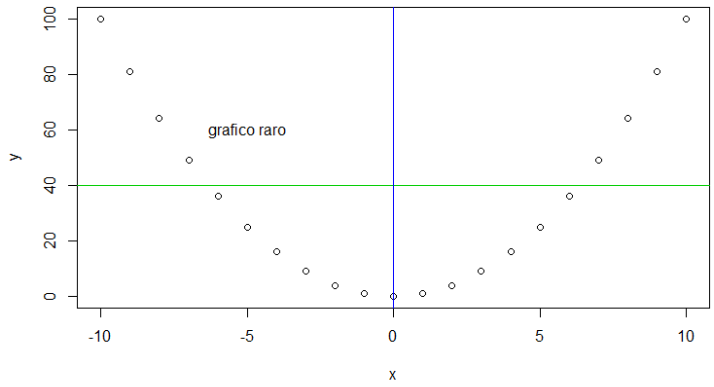
points(x,y) **lines(x,y)**	**Añade puntos o líneas conectadas al gráfico actual**
text(x,y,etiquetas)	**Añade texto al gráfico actual en la posición x,y**
abline(a,b) **abline(h=y)** **abline(v=x)**	**Añade una línea de pendiente a y que corta al origen en b** **Añade línea horizontal que corta al eje y en h=y** **Lo análogo para línea vertical**
polygon(x,y)	**Dibuja un polígono**
title(main,sub)	**Añade título y subtítulo al gráfico actual**
axis(side)	**Añade ejes al gráfico actual (de 1 a 4)**

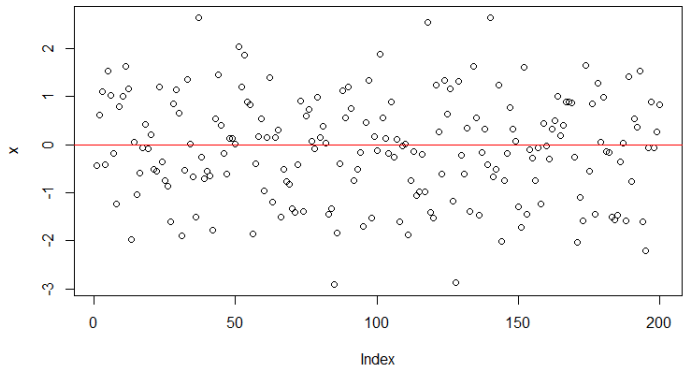
Algunos ejemplos:

```
> x = seq(-10,10)
> y = x^2
> plot(x,y,axes=F)
> axis(1)
```

```
> x = seq(-10,10)
> y = x^2
> plot(x,y)
> abline(h=40,col=3)
> abline(v=0,col=4)
> text(-5,60,"grafico raro")
```

```
> x = rnorm(200)
> plot(x)
> abline(h=0,col=2)
```





7.13 VARIOS GRAFICOS EN UNA MISMA VENTANA

Para conseguir esto se usa el comando par, con la opción mfrow.

Ejemplos:

par(mfrow=c(1,1)) **# un solo gráfico por ventana: la opción por defecto**
par(mfrow=c(2,1)) **# Dibuja una matriz de gráficos 2x1: un gráfico debajo de otro**
par(mfrow=c(2,3)) **# Matriz de gráficos 2 x 3 : dos filas por tres columnas**

Un ejemplo:

```
> x = rnorm(200)      # Se generan 200 valores de una normal
estandarizada
> par(mfrow=c(2,2))   # Se crea una matriz de gráficos 2 x 2
> plot(x)             # Dibujo de x frente al índice 1 a 200
> hist(x)             # Histograma de x
> boxplot(x)          # Diagrama de caja de x
> qqnorm(x)           # Gráfico cuantil-cuantil de x frente a la
distribución normal
```