Nociones generales sobre R

Nociones generales sobre R

1. Nociones sobre el sistema R

1. 1. Breve descripción de R

Software constituido por un conjunto de métodos estadísticos que están en continuo desarrollo y disponibles libremente. Contiene la mayoría de las técnicas estadísticas clásicas estándar así como las últimas metodologías en uso. Lo emplearemos para el análisis de datos. El sistema base está disponible en forma precompilada para Windows. Las ultimas versiones del paquete **base** vienen apoyadas por paquetes adicionales conectados con prioridad alta para implementar las tareas estadísticas más usadas en la práctica. Entre los múltiples paquetes que lo integran nos valdremos de los siguientes:

```
"package:methods"   "package:stats"
"package:graphics"  "package:grDevices" "package:utils"
"package:datasets"  "Autoloads"         "package:base"
```

Estos servirán para los fines propuestos en este cuatrimestre para el tratamiento estadístico de datos.
El paquete **base** se considera parte del código fuente de R y se carga automáticamente cuando se instala R. Contiene las funciones básicas para trabajar con R.
Puede descargar a su ordenador con sistema Windows conectando a

http://CRAN.R-project.org/bin/Windows/base/release.htm

Descargue el fichero denominado rwnum.exe y ejecútelo. Donde num indica alguna de las diferentes versiones que se van actualizando

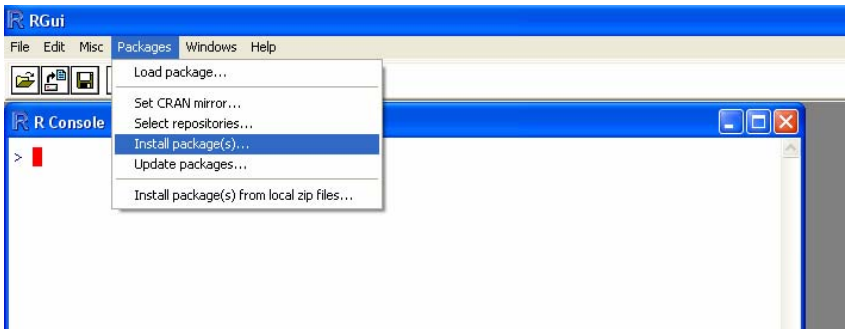
Se pueden instalar directamente desde R otros paquetes que no se incluyen en la lista adicional del base. Para ello puede invocarlos desde el menú. O bien mediante la orden **install.package("nombre")**, donde nombre se refiere al nombre del package a instalar.
Hay en torno a unos mil paquetes distribuidos en Internet, correspondientes a aportaciones realizadas a través de tiempo de diferentes autores, que diversifican, flexibilizan y amplían el espectro de los usuarios.
Cada paquete añadido a la librería (carpeta library) estará disponible para cargarlo y ejecutarlo.
library("nombre")

Vea la lista de paquetes disponibles en

http://CRAN.R-project.org/src/contrib/PACKAGES.html

Bajo Windows, se descargan e instalan las versiones precompiladas de estos packages.

Una vez instalado el package base, puede acceder a cargar o a instalar nuevos packages desde las opciones de menú de **R.** Accediendo a la dirección de Internet o bien al directorio en que se encuentren en versión **zip**



Existen diferencias importantes entre R y otros sistemas estadísticos como por ejemplo SPSS o Statgraphics. Todas estas entidades que maneja R se denominan **objetos**. **R** presenta los datos, resultados y funciones como **objetos** que pueden ser posteriormente tratados.

R trabaja manteniendo en memoria copias de estos objetos, en lo que se denomina área de trabajo o "**workspace**".

Puede ver un listado de todo lo que R tiene disponible y accesible de modo directo desde el área de trabajo y que carga automáticamente, si tras arrancar el sistema escribe:
> ls()

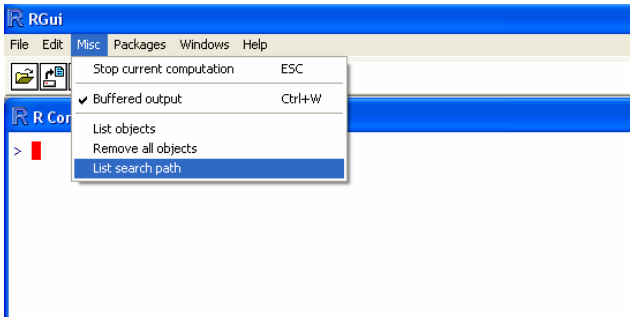
```
character(0)
```

> search()

```
[1] ".GlobalEnv"        "package:methods"   "package:stats"
[4] "package:graphics"  "package:grDevices" "package:utils"
[7] "package:datasets"  "Autoloads"         "package:base"
```

También puede acceder desde la ventana principal en la barra de menú a esta información mediante:

Misc…list objects
(Para un listado de los objetos disponibles)
Misc…list search path
(Para ver un listado de paquetes que ha cargado el sistema o a los que puede acceder directamente)



Puede ejecutarse interactivamente y proporcionando el archivo que contiene el código o instrucciones a ejecutar.

1. 2.El sistema de AYUDA de R

Para requerir ayuda en línea sobre los objetos use el símbolo ? seguido del nombre de la función u objeto para el que necesita información. También puede usar help() . Por ejemplo:

>help(lm)

es equivalente a

>?lm

Puede usar help.search("nombre") para buscar dentro de las páginas del manual

help.start()
Presenta el sistema de ayuda de **R** disponible.

Para ver iformación sobre un paquete escriba:

>**help**(package="*nombre*")

Hay packages que vienen con información adicional denominadas vignette. Para visualizarla escriba:

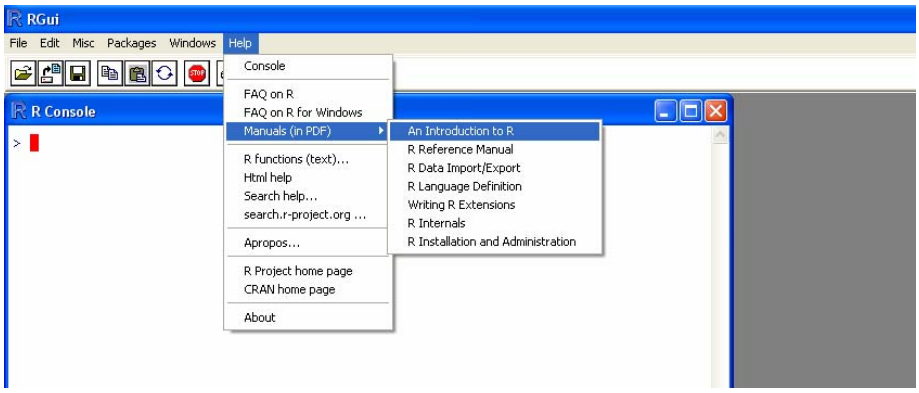
vignette("nombre paquete")

Los manuales completos se encuentran en la dirección:

http://CRAN.R-project.org/manuals.html

Las preguntas más frecuentes se encuentran en

http://CRAN.R-project.org/faqs.html



1.3. El entorno de Trabajo de R

1.3.1. Guardar objetos para futuras sesiones

Cuando salga de la aplicación R, (exit) tiene la opción de guardar el entorno de trabajo para futuras sesiones.

Los objetos creados en una sesión de R se guardan en el directorio corriente. Pueden guardarse al acabar la sesión automáticamente en un archivo denominado R por defecto (con extensión '.RData'), en el directorio corriente. También se pueden guardar las lineas de los comandos usados en la sesión, en el archivo '.Rhistory'.

(Vea icono R en el directorio de trabajo. Este archivo contiene los objetos creados en la sesión y se carga automáticamente).
Puede cambiarle el nombre si no desea que se cargue automáticamente, sino sólo cuando lo solicite: Save workspace e introduzca un nombre de archivo que contendrá el entorno de trabajo.
La ventaja de guardarlo es que puede seguir trabajando en la sesión corriente con los objetos creados en una sesión previa: Load workspace e introduzca el nombre de archivo

Recuerde que en cualquier momento puede mostrar qué objetos están cargados en memoria:

Con la opción **ls()** se listan los objetos que están presentes (en memoria) en la sesión de trabajo (workspace).

Con la opción **rm()** se borran los objetos en el entorno de trabajo
Estas opciones están también disponibles desde el menú de R: Misc…

Para evitar confusión con los nombres de los objetos creados en distintos análisis se aconseja cambiar de directorio de trabajo.

File…Change dir

2. Manipulación de datos

2.1. Introducción y lectura de datos para tratamiento estadístico

Los datos para un análisis estadístico se caracterizan por establecerse en formato de matrices (filas y columnas). Generalmente representan **variables** medidas sobre un

conjunto de **individuos**. Cada variable se representa en una columna cuya primera fila contiene el nombre. A esta estructura de información se denomina en **R** objeto **data.frame**.

La manipulación, acceso, creación y/o lectura de la información o los datos para el análisis puede llegar desde muy diversas vías.

-Mediante lectura de archivos existentes, presentados en diversos formatos, creados por otras aplicaciones.
-Mediante uso del propio editor **R** de archivos **script**. Permite crear y modificar información así como guardarla en **archivos**.
-Mediante uso de ventana del editor de data.frame edit(). Permite editar, crear y modificar **objetos data.frame.**
-Acceso a archivos disponibles en **packages** de **R**

2.1.1. Lectura de archivos de datos

Aunque hay modos de importar y exportar con R, es usual preparar los archivos de datos de otros paquetes estadísticos para importarlos como objetos data.frame con R mediante **read.table()**[1]**.**

Uno de los archivos más usados para lectura será el de extensión **.txt** (creado con un editor de texto) o del tipo generado con utilidades como **Excel** o programa **SPSS** (guardado con delimitador **tab**[2]). O bien, generado desde **Excel** y guardado como delimitado por punto y coma (**.csv**), que se puede leer con **read.cvs2()**.

En este caso el archivo de datos presenta formato de tabla de filas y columnas, con la primera fila generalmente con los nombres de las variables. Cada fila siguiente contiene etiqueta de fila (nº de fila por ejemplo) y a continuación, los datos (numéricos, categóricos,…).
A veces no interesa etiquetar las filas. En este caso, se puede omitir la etiqueta de la fila (el sistema usa por defecto, la numeración).

Ejemplo

Dado el archivo de texto con la información:

```
  a b c
1 1 1
2 2 2
3 3 3
1 4 5
2 3 3
2 1 1
3 7 7
```

Puede leerse mediante:

```
> read.table("eje1.txt",header=T)
```

[1] Que genera un objeto data.frame.
[2] Cuando el archivo original por ejemplo en SPSS presenta datos missing con blancos, el archivo generado guardado con formato delimitador tabulación puede dar problemas. Versiones últimas de R permiten editar el archivo y sustituir los blancos por caracteres missing como NA. Edit…Replace.

```
  a b c
1 1 1 1
2 2 2 2
3 3 3 3
4 1 4 5
5 2 3 3
6 2 1 1
7 3 7 7
```

Notas:
-use **?read.table()** para más información.
-Vea que la opción **sep** permite indicar el tipo de separador de columna que se ha usado en el archivo. Esto evitará problemas al leerlo.
-Vea que puede indicar el separador decimal que se ha usado para los datos. Es necesario indicarlo si se ha utilizado la coma (**dec**=",")
- Para más información vea los ejemplos realizados en las prácticas iniciales en la carpeta **1.2.Practicas iniciales**.

2.1.2. Introducción de datos por teclado mediante scan()

Con la opción **scan()** se puede introducir vectores por teclado[3]. Se usa si es poca información a registrar.

Ejemplo

Para introducir el conjunto de datos 1, 2, 3, 7, 9 , en un vector de nombre **a**
1. Escriba a<- scan()
2. pulse intro
3. introduzca los datos (separados por espacio o por intro)
4. finalice pulsando las teclas **intro** y **ctrol+D**

```
> a <- scan()

1: 1
2: 2
3: 3
4: 7
5: 9
6:
Read 5 items

> a
[1] 1 2 3 7 9
```

2.1.3. Carga de datos de otros paquetes: data()

Escriba data, seguido de los argumentos: nombre del data.frame y el nombre del paquete (este último entre comillas)

>**data**(*nombredataframe*, package="*nombrepackage*")

Ejemplo:

Para cargar el data.frame Puromycin del package denominado datasets

```
>data(Puromycin, package="datasets")
```

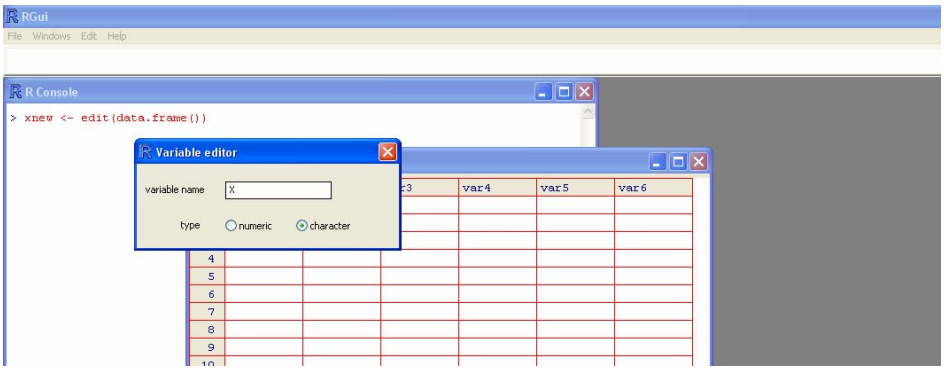
[3] El problema es que una vez aceptado el dato no se puede modificar, borrar o corregir.

2.1.4. Introducción de datos mediante el editor de data.frame

Ejemplo

Para introducir datos en un data.frame denominado xnew, escriba:

```
>xnew <- edit(data.frame())
```



Introduzca nombres para variables e introduzca datos.

```
> xnew
        X    Y    Z
1      23    22    1
2      22    34    2
3      14    56    3
```

Para posteriores modificaciones escriba:

```
 xnew=edit(xnew)
```

2.1.5. Introducción de datos desde el Editor R

Desde la barra de menú seleccione:

Archivo….Nuevo Scrit

Desde esta opción de menú abrirá una ventana similar a un procesador de texto para crear un nuevo archivo cuyos datos podrán introducirse directamente desde teclado.

2.2. Exportar archivos para recuperar con Excel: write.table

Puede guardar data.frames en archivos para utilizar con la aplicación Excel.
La función **write.table** permite guardar datos de un data.frame en un archivo que puede abrirse con Excel

Ejemplo: genere datos con edit en un data frame de nombre x

```
> x<-edit(data.frame())
```

Introduzca los datos y visualice en pantalla

```
> x

     x1   x2
1   1.3  2.6
2  1.56 3.89
3 567.4  4.6
```

Escriba

```
> write.csv2(x,"filex")  #utiliza como separador de columna punto y coma. Y usa como delimitador decimal la coma.
```

Se generará un archivo de nombre **filex** que puede abrirse con Excel

2.3. Guardar objetos en archivos

Permite guardar (save) de modo permanente objetos que pueden cargarse (load) en otras sesiones

Ejemplo:

Guarde los objetos de nombres objeto1 y objeto2 en un archivo de nombre nombrefile

save(*objeto1*, *objeto2*, file="*nombrefile*.rda")

El archivo se guarda en el directorio corriente. Aunque cierre la aplicación, puede recuperar los objetos y cargarlos para sesiones futuras mediante:

load(file="*nombrefile*.rda")

2.4. Tipo de extensiones para archivos de diversa naturaleza soportados por R

1. Archivos con extensión '.R' o '.r' se denominan fuente source()
2. Archivos con extensión '.RData' o '.rda' se cargan al comenzar una sesión load()
3. Archivos con extensión '.tab', '.txt' o '.TXT' usados para leer con read.table(..., header = TRUE), y generar un data frame.
4. Archivos con extension '.csv' o '.CSV' para leer con read.table(..., header = TRUE, sep = ";"), y generar un data frame.

3. Los gráficos en R

La mayoría de los gráficos se exploran mejor en su contexto estadístico, no obstante, veremos algunas de las posibilidades que ofrece R para describir gráficamente la información.

Los paquetes: lattice y rgl están diseñados para manejar gráficos especiales en R

Enumeramos algunas opciones gráficas, para las que puede obtener ayuda adicional del sistema. Vea también los ejemplos prácticos realizados.

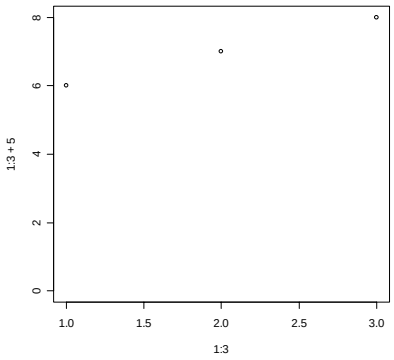
Algunas opciones gráficas usuales útiles son:

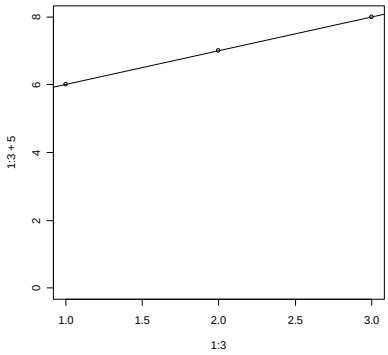
abline(*intercep***,** *pendiente***)**

Añade una **línea recta** a un gráfico existente, dando la intercept y la pendiente

Ejemplo:

```
> plot(1:3,1:3+5)
> abline(5,1) # añade una linea al gráfico de puntos corriente con intercep 5 y pendiente 1.
```

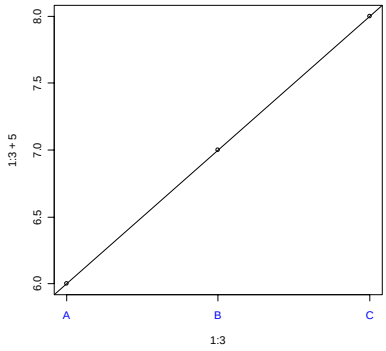




axis añade un eje al gráfico corriente

Ejemplo

```
plot(1:3,1:3+5,axes=FALSE) #quita los ejes por defecto
abline(5,1) # añade una linea al gráfico de puntos corriente con intercep 5 y pendiente 1.
axis(1, 1:3, LETTERS[1:3], col.axis = "blue") #añade eje abscisa y coloca letras
axis(2) #añade el eje vertical por defecto
```



barplot gráfico de barras

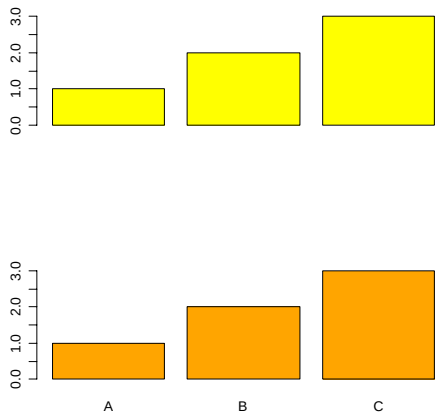
barplot(*vector_o_matriz*,…)

Los valores del argumento *vector_o_matriz* describen las alturas de las barras de una o varias series

Ejemplo 1: Un vector de datos

```
>x=1:3
> class(x)
```

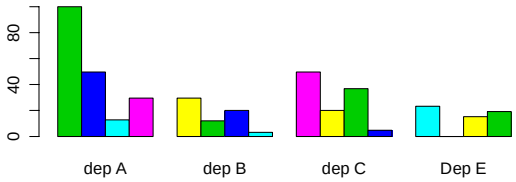
```
[1] "integer"
> x
[1] 1 2 3
>
> barplot(x)
> barplot(x,names.arg=c("A","B","C"))
> barplot(x,col="yellow")
> barplot(x,names.arg=c("A","B","C"),col="orange")
>
```

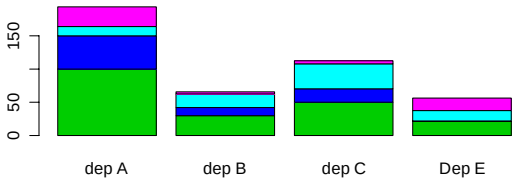


Ejemplo 2:

Los datos muestran una matriz de gasto medio en 4 departamentos (columnas) en 4 partidas (filas)

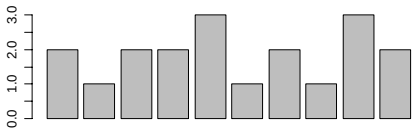
```
> gasto
            dep A dep B dep C Dep E
nominas       100    30    50    23
importacion    50    12    20     0
publicidad     13    20    37    15
varios         30     3     5    19

> class(gasto)
[1] "matrix"

> rownames(gasto)
[1] "nominas"     "importacion" "publicidad"  "varios"
> colnames(gasto)
[1] "dep A" "dep B" "dep C" "Dep E"

> barplot(gasto,beside=T,col=3:7)
> barplot(gasto,beside=F,col=3:7)
>
```

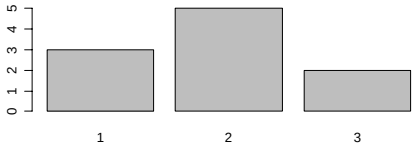




Ejemplo 3:

```
>  x=rpois(10,2)
> x
 [1] 2 1 2 2 3 1 2 1 3 2
>  x=rpois(10,2)
> x
 [1] 2 1 2 2 3 1 2 1 3 2
> table(x)
x
1 2 3
3 5 2
> barplot(x)
> barplot(table(x))
```





boxplot gráfico caja.

boxplot(*formula*, data = *datos*, ..., subset=*operadorlógico*, col="*color*")

Los argumentos más usuales son fórmula, data, subset, col
formula expresa la **variable continua** para la que se construye el gráfico y una variable categórica que permite establecer subgrupos para los que se realizan cajas separadas. Al mismo tiempo se puede estratificar o subdividir en subgrupos mediante subset.

Ejemplos:

boxplot(y, col=”red”)
boxplot(y~x, data=misdatos, subset=sexo==”h”, col=”blue”)

Nota Importante: Si solo se utiliza una variable de agrupamiento, esta debe emplearse en la fórmula. No en subset. Sólo se utiliza subset cuando interviene más de una variable de agrupamiento y hemos usado una en la fórmula.
Por ejemplo, use:
boxplot(y~sexo,col="red")
pero, No use:
boxplot(y,subset=sexo==”h”)
boxplot(y,subset=sexo==”m”)
porque el sistema ignora la clasificación y representa todos los casos.

hist histograma
R permite que el usuario introduzca un número de clases (al menos aproximado, ya que si el número no es adecuado lo sustituye por otro más conveniente) mediante el argumento nclass=nº, o que introduzca los extremos de los intervalos (argumento br=vector con los extremos de intervalo)

hist(*Var*, nclass= *num*,...)
hist(*Var*, br= *num*,...)

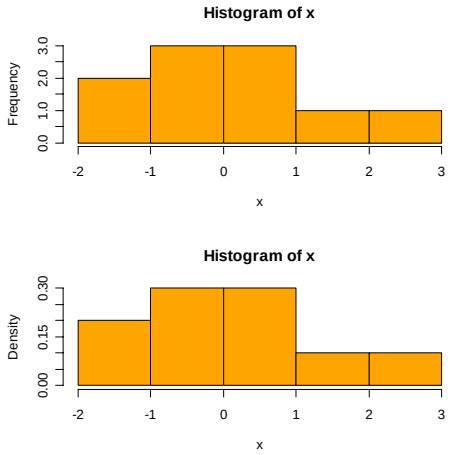
Ejemplo 1

```
> x=rnorm(100,2,0.5)
> hist(x,nclass=4)
> hist(x,br=c(-1,1,2,3,4,5))
```

Ejemplo 2

```
x=rnorm(10)
> hist(x,plot=F)

$breaks
[1] -2 -1  0  1  2  3
$counts
[1] 2 3 3 1 1
$intensities
[1] 0.2000000 0.3000000 0.3000000 0.1000000 0.1000000
$density
[1] 0.2000000 0.3000000 0.3000000 0.1000000 0.1000000
$mids
[1] -1.5 -0.5  0.5  1.5  2.5
$xname
[1] "x"
$equidist
[1] TRUE
attr(,"class")
[1] "histogram"
```

hist(x,col=”orange”)
hist(x,freq=F,col=”orange”)



layout (*matriz,…***)**
En ayuda se especifican los parámetros de la función que permiten modificar el ouput gráfico. Por ejemplo, una opción muy usada, es la que permite dividir la pantalla gráfica en secciones para colocar varios gráficos juntos.

Ejemplo:

layout (matriz(1:4,nrow=2))
Divide el espacio gráfico en 2x2 partes (2 filas y 2 columnas) para colocar los gráficos

Reserva el espacio dividido donde ubicarán los siguientes 4 gráficos generados tras esa orden en el espacio organizado en 4 partes

legend añade etiqueta al gráfico

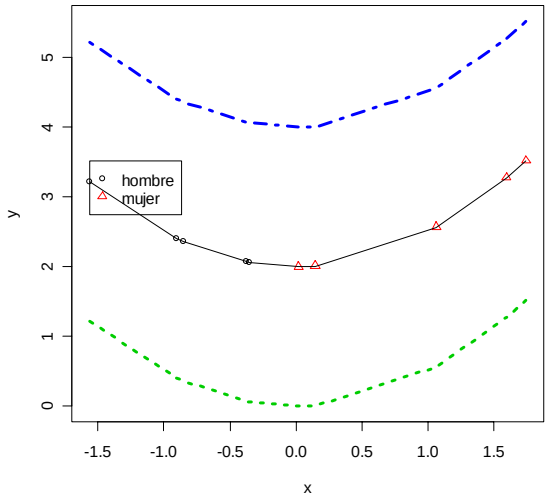
legend(*coordx*, *coordy* , *leyenda*, …)

lines añade lineas al gráfico

lines(*coordx*,*coordy*, lty=*nºtipolinea*, lwd=*nºgrueso*, ...)

Ejemplo:

```
x=rnorm(10)
x=sort(x)
y=x^2*0.5+2
t1=y-2
t2=y+2
z=c(rep(1,5),rep(2,5))

plot(x, y, ylim=c(min(t1),max(t2)), col=z, pch=z)

lines(x,y,lty=1,col=1)
lines(x,t1,lty=3,col=3,lwd=3)
lines(x,t2,lty=4,col=4,lwd=3)

legend(min(x),max(y), c("hombre","mujer"), pch=c(1,2), col=c(1,2))
```



Matplot varios plot para las columnas de una matriz

matplot(*vect_o_matrizx*, *vect_o_matrizy*, type = "*p*", lty = *1:5*, lwd = *1*, pch = *NULL*,
col = *1:6*, … xlab = *NULL*, ylab = *NULL*, xlim = *NULL*, ylim =)

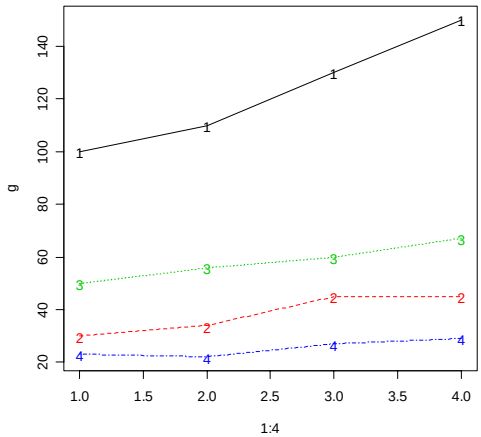
matpoints(*vect_o_matrizx*, *vect_o_matrizy*, type = "*p*", lty = 1:5, lwd = *1*, pch = *NULL*,
col = 1:6, ...)

matlines (*vect_o_matrizx*, *vect_o_matrizy*, type = "*l*", lty = *1:5*, … col = *1:6*, ...)

Ejemplo:

```
> g
     trim1 trim2 trim3 trim4
2000   100    30    50    23
2001   110    34    56    22
2002   130    45    60    27
2003   150    45    67    29
```

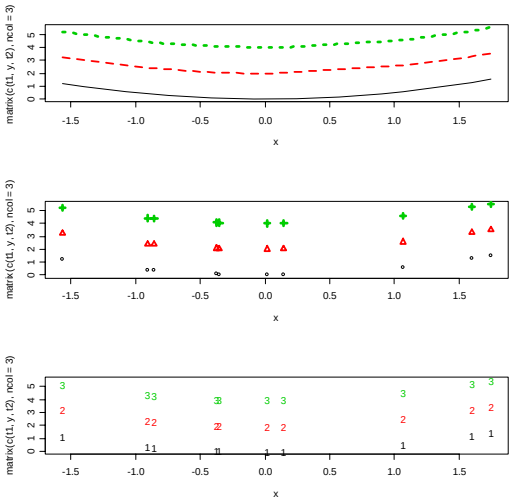
> matplot(1:4,g)
> matlines(1:4,g,lty=1:4)



Ejemplo 2

> layout(1:3)
> matplot(x, matrix(c(t1,y,t2),ncol=3),col=1:3, type="l",lty=1:3,lwd=1:3)
> matplot(x, matrix(c(t1,y,t2),ncol=3),col=1:3, type="p",lty=1:3,lwd=1:3,pch=1:3)
> matplot(x, matrix(c(t1,y,t2),ncol=3),col=1:3, type="p",lty=1:3,lwd=1:3)

>



mtext añade texto en los márgenes

mtext("*text*", side = *nºde 1 a 4*, ...)

Ejemplo:
> mtext("CURVAS", side=3,col="red")
> mtext("tres curvas", side=2,col="red")
>

Par
Permite presentar múltiples figuras en un plot.

Ejemplo:
op=par(mfrow=c(2,3))
#los 6 gráficos que realice se estructuran en un 2x3 espacio de dos filas y 3 columnas
par(op) #restaura la pantalla.

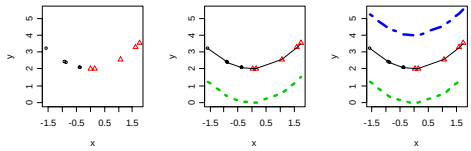
Ejemplo:
grf <- par(mfrow = c(1, 3), pty = "s") #3 gráficos en regiones cuadradas,

plot(x, y, ylim=c(min(t1),max(t2)), col=z, pch=z) #primer gráfico sin lineas

plot(x, y, ylim=c(min(t1),max(t2)), col=z, pch=z) #segundo gráfico con líneas
lines(x, y,lty=1,col=1)
lines(x, t1,lty=3,col=3,lwd=3)

plot(x, y, ylim=c(min(t1),max(t2)), col=z, pch=z) #tercer gráfico con 3 líneas
lines(x, y,lty=1,col=1)
lines(x, t1,lty=3,col=3,lwd=3)
lines(x, t2,lty=4,col=4,lwd=3)

Restaura:
par(grf)

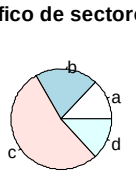


pie gráfico de sectores

pie(*vectorareasector*, labels = names(*vectorareasector*), …,main = “titulo del gráfico”, ...)

Ejemplo

```
> layout(1:2)
> px=c(2,3,8,2)
> modalid=c("a","b","c","d")
> pie(rpx, labels=modalid, main="grafico de sectores")
> names(px)= c("na","nb","nc","nd")
> pie(rpx,  labels=names(px),main="grafico de sectores",col=1:4)
>
```



plot función genérica. (vea ejemplos anteriores y pulse ayuda par información adicional)

points añade puntos a un gráfico. Presenta una estructura similar a **lines** (ayuda para más información)

qqnorm cuantiles-cuantil y normal q-qplot (se verá más adelante en **carpeta de regresión**)

El gráfico qqnorm pone los valores de la variable de la muestra frente a los esperados bajo un modelo normal. Permite detectar si la muestra se comporta aproximadamente como un modelo normal, si la forma de las colas de la distribución difiere o no de la normal, o incluso, si hay puntos anómalos. Se suele añadir una recta que pasa por primer y tercer cuartil del correspondiente modelo normal, de modo que se refleje mejor la concordancia o discordancia con el modelo normal. Puntos próximos a la recta son indicio de comportamiento próximo a la normal.
qqnorm(*var*); **qqline**(*var*)

qqplot permite comparar las distribuciones de dos variables colocando los cuantiles de una de ellas frente a los de la otra, **qqplot**(*var1, var2*).

rect añade rectángulos (vea ayuda para detalles)

symbols dibuja símbolos en un gráfico (vea ayuda para detalles)

text añade texto al corriente gráfico (plot)

text(v*coordx*,v*coordy*., v*texto*, ...)

Los argumentos coordx, coordy representan las coordenadas en eje x e y respectivamete, indican la posición donde colocar el texto. Si hay más de uno se indican con vectores. El vector de texto de caracteres contiene el texto a añadir al gráfico.
Ejemplo

```
>  plot(1:3,2+1:3)
> text(1,4,"falta la recta",col="blue",cex=0.8)
> text(1.2,4.5,"falta la recta",col="red",cex=1)
> text(1:3,3:5,LETTERS[1:3],col=3,cex=2)
```

title añade título a un gráfico
Ejemplo

```
>  px=c(2,3,8,2)
>  modalid=c("a","b","c","d")
>  pie(rpx, labels=modalid)
> title("grafico de sectores")
```

4. Forma de combinar, modificar y resumir data.frames

Los objetos data.frame representan conjuntos de información expresada en columnas y filas. Cada columna representa una variable que puede ser de tipo cualitativo o cuantitativo.

4.1. Declarar códigos numéricos como niveles de una variable factor

```
> a
  var1 var2
1    0   23
2    1   34
3    1   55
4    0   23
5    0    2
> a$var1=factor(a$var1)
> a
  var1 var2
1    0   23
2    1   34
3    1   55
4    0   23
```

```
5    0    2
> levels(a$var1)[1]<-"bajo"
> levels(a$var1)[2]<-"alto"
> a
  var1 var2
1 bajo   23
2 alto   34
3 alto   55
4 bajo   23
5 bajo    2
```

4.2. Recodificación de los niveles de los factores

```
> a
  var1 var2 var3 var4
1 bajo   23    1    b
2 alto   34    2    a
3 alto   55    2    a
4 bajo   23    1    c
5 bajo    2    1    a
> a$var4=factor(a$var4)
> a$var4
[1] b a a c a
Levels: a b c
> levels(a$var4)=c("bajo","medio","alto")
> a
  var1 var2 var3  var4
1 bajo   23    1 medio
2 alto   34    2  bajo
3 alto   55    2  bajo
4 bajo   23    1  alto
5 bajo    2    1  bajo
```

4.3. Combinación de data.frames mediante cbind y rbind.

Combinan objetos data.frame por columnas y por filas
cbind(a,b) #combina por columnas los data.frames a y b
rbind(a,b) #combina por filas los data.frames a y b

4.4. Operaciones con las columnas o variables de una data.frame

Operaciones como suma, resta, producto, …de variables
Operaciones como logaritmos, exponencial, …de variables
Ejemplo:
a$nueva=a[,3]/a[,4] #añade una columna de nombre nueva resultado de dividir las variables (columnas) 3 y 4

4.5. La función attach

Permite referenciar los nombres de las columnas de los data.frames, sin necesidad de especificar el nombre del data.frame precedido del símbolo $, lo que agiliza su manipulación.
La función **detach** lo desactiva.

4.6. La función apply

Opera sobre arrays (todas las variables son cuantitativas o todas cualitativas)
En la matriz m formada por filas y columnas realiza una función sobre las filas, o sobre las columnas o sobre ambas a la vez.

apply(*matriz, n, función*)

Ejemplo

> m #matriz de datos

```
     [,1] [,2]
[1,]    3    1
[2,]    2    2
[3,]    4    3
[4,]    4    4
```

> apply(m,1,sum) #realiza la suma de las filas

```
[1] 4 4 7 8
```

> apply(m,2,mean) calcula la media en las columnas

```
[1] 3.25 2.50
```

> apply(m,c(1,2),log)

```
          [,1]      [,2]
[1,] 1.0986123 0.0000000
[2,] 0.6931472 0.6931472
[3,] 1.3862944 1.0986123
[4,] 1.3862944 1.3862944
```

4.7. División del data.frame con la función split

Podemos considerar grupos de filas dividiendo el data frame según uno o más factores.
Division mediante split
split(*dataframe*, *factor*)

Ejemplo:

> a

```
  x1 x2
1  1  a
2  2  a
3  3  a
4 30  b
5 40  b
6 45  b
```

> split(a,a$x2)

```
$a
  x1 x2
1  1  a
2  2  a
3  3  a

$b
  x1 x2
4 30  b
5 40  b
6 45  b
```

4.8. División del data.frame con la función by

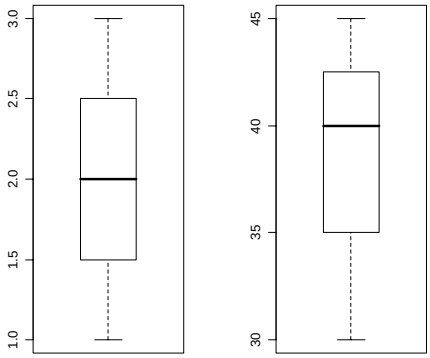
by(*dataframe***,** *factor o lista de factores*, *función***)**

Ejemplo:

> by(a[,1],list(a$x2),summary)

```
: a
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0     1.5     2.0     2.0     2.5     3.0
------------------------------------------------------------
: b
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  30.00   35.00   40.00   38.33   42.50   45.00
```

>par(mfrow=c(1,2))
>by(a[,1],list(a$x2),boxplot)
>par(op)



4.9. División del data.frame mediante aggregate

aggregate(*dataframe***,** *factor o lista de factores, función***)**

aggregate es similar a **by** pero el output para el subgrupo es un solo resultado, en vez de una lista
Ejemplo:

```
> aggregate(a[,1],by=list(a$x2),median)

  Group.1  x
1       a  2
2       b 40
```

4.10. Mezcla de data.frames

Mezcla dos data.frames con una variable como clave que permite relacionar los casos de ambos.

Ejemplo:

```
> a
  x1 x2 clave
1  1  a     1
2  2  a     2
3  3  a     3
4 30  b     4
5 40  b     5
6 45  b     6
> b
  x1 x2 clave
1  1  a     8
2  2  a     2
3  3  a     3
4 30  b     4
5 40  b     5
6 45  b     6

> merge(a,b,by.x="clave",by.y="clave") # incluye sólo los elementos comunes en ambos data.frames

  clave x1.x x2.x x1.y x2.y
1     2    2    a    2    a
2     3    3    a    3    a
3     4   30    b   30    b
4     5   40    b   40    b
5     6   45    b   45    b
>
```

> merge(a,b,by.x="clave",by.y="clave",all=T) #incluye todos los casos y coloca missing en las columnas correspondientes para los casos que no hay información.

```
  clave x1.x x2.x x1.y x2.y
1     1    1    a   NA <NA>
2     2    2    a    2    a
3     3    3    a    3    a
4     4   30    b   30    b
5     5   40    b   40    b
6     6   45    b   45    b
7     8   NA <NA>    1    a
>
```

Cambio de la estructura o formato de presentación de data.frame para variables que contienen información repetida para cada sujeto: reshape

Ejemplos típicos de este tipo de presentación de datos son mediciones de cierta variable a través del tiempo para cada sujeto.
Vea **?reshape** para más información

4.11. Extracción de información de un data.frame mediante subset

Ejemplo:

> a

```
  sexo edad
1    1   23
2    2   22
3    2   24
4    1   22
5    1   23
6    2   23
7    1   24
```

> subset(a, sexo==1, select=edad)

```
  edad
1   23
4   22
5   23
7   24
```

> subset(a, sexo==1&edad>22, select=edad)

```
  edad
1   23
5   23
7   24
```

> subset(a, sexo==1&edad>22)

```
  sexo edad
1    1   23
5    1   23
7    1   24
```

4.12. Acceso a componentes de un vector o data.frame

A[i] muestra el valor de la componente i-ésima del vector A

> a

```
  sexo edad
1    1   23
2    2   22
3    2   24
4    1   22
5    1   23
6    2   23
7    1   24
```

> a[,1]

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

> a[1]

```
  sexo
1    1
2    2
3    2
```

```
4    1
5    1
6    2
7    1
```

4.13. Acceso a una componente de una lista

A[[i]] muestra el componente i-ésimo de la lista A
A$c muestra el componente de nombre c de la lista A

> a[[1]]

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

> a$sexo

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

4.14. Ordenación de datos

La ordenación de un vector es inmediata.

Ejemplo

```
xx=c(8,1,4,6,3,2,1,2,4,5)
sort(xx)
```

```
[1] 1 1 2 2 3 4 4 5 6 8
```

La ordenación de un data.frame, requiere obtener la permutación adecuada que generará el data.frame ordenado según una o más variables, mediante la función **order**

Ordenación del data.frame a

```
> a
   var1 var2 var3
1     3    4    8
2     4    2    1
3     2    6    3
4     1    9    5
5     2    1    7
6     7    2    4
7     1    5    9
8     2    4    8
9     8    3    2
10    3    1    3
```

> rbind(a$var1,a$var2,a$var3)[,order(a$var1,a$var2)] #las 3 variables del data.frame se ordenan en función de las variables var1 y var2, respectivamente. Par presentar el resultado con variables estructuradas en columnas se realiza la transposición del resultado (la función t(*matriz*) cambia filas por columnas).

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    1    2    2    2    3    3    4    7     8
[2,]    5    9    1    4    6    1    4    2    2     3
[3,]    9    5    7    8    3    3    8    1    4     2
```

> t(rbind(a$var1,a$var2,a$var3)[,order(a$var1,a$var2)])

```
      [,1] [,2] [,3]
 [1,]    1    5    9
 [2,]    1    9    5
 [3,]    2    1    7
 [4,]    2    4    8
 [5,]    2    6    3
 [6,]    3    1    3
 [7,]    3    4    8
 [8,]    4    2    1
 [9,]    7    2    4
[10,]    8    3    2
```

> t(rbind(a$var1,a$var2,a$var3)[,order(a$var3)]) #Ordena los casos según sus valores en var3

```
      [,1] [,2] [,3]
 [1,]    4    2    1
 [2,]    8    3    2
 [3,]    2    6    3
 [4,]    3    1    3
 [5,]    7    2    4
 [6,]    1    9    5
 [7,]    2    1    7
 [8,]    3    4    8
 [9,]    2    4    8
[10,]    1    5    9
```

5. Tabulación

A veces la información presenta variables cualitativas o factores con pocas modalidades e interesa resumirla en tablas de frecuencia de una, dos o más dimensiones.
Otras veces interesa poner una tabla de contingencia en formato de data.frame
Para más información vea carpeta 2.Tabulación de variables categóricas

5.1. Funciones relacionadas con la tabulación

ftable()

ftable(*dtf_o_table*, row.vars=*nºonombr1e*, col.vars= *nºonombre2* , ...)

El argumento *dtf_o_table* puede ser un data.frame formado por 2 ó más variables cualitativas o factores (es decir, variables con pocas modalidades, para que tenga sentido tabular) o un objeto que representa ya una tabla de contingencia(es decir, puede ser el output de **table** o **ftable**). No puede ser un vector, dado que manipula tablas de contingencia.
Los valores de row.vars y col.vars pueden ser números o nombres entre comillas que hacen referencia a las variables de la tabla.

Ejemplo 1 del manual de R
El input es ya una tabla de contingencia: Titanic.con 4 variables categóricas: Age,Survived,Class,Sex

> Titanic #visualiza la tabla de contingencia

```
, , Age = Child, Survived = No

      Sex
Class  Male Female
  1st     0      0
  2nd     0      0
  3rd    35     17
  Crew    0      0

, , Age = Adult, Survived = No

      Sex
Class  Male Female
  1st   118      4
  2nd   154     13
  3rd   387     89
  Crew  670      3

, , Age = Child, Survived = Yes

      Sex
Class  Male Female
  1st     5      1
  2nd    11     13
  3rd    13     14
  Crew    0      0

, , Age = Adult, Survived = Yes
```

```
      Sex
Class  Male Female
  1st    57    140
  2nd    14     80
  3rd    75     76
  Crew  192     20
```

> ftable(Titanic, row.vars = 1:3) #Coloca las modalidades de 3 variables en filas. Las modalidades de la cuarta variable Survived aparece en columnas:

```
                    Survived  No Yes
Class Sex    Age
1st   Male   Child            0   5
             Adult          118  57
      Female Child            0   1
             Adult            4 140
2nd   Male   Child            0  11
             Adult          154  14
      Female Child            0  13
             Adult           13  80
3rd   Male   Child           35  13
             Adult          387  75
      Female Child           17  14
             Adult           89  76
Crew  Male   Child            0   0
             Adult          670 192
      Female Child            0   0
             Adult            3  20
>
```

ftable(Titanic, row.vars = 1:2, col.vars = "Survived")
> ftable(Titanic, row.vars = 1:2, col.vars = "Survived") #Reagrupa o colapsa una variable. Tabla marginal con solo 3 variables. Dos en filas y una en columnas:

```
             Survived  No Yes
Class Sex
1st   Male            118  62
      Female            4 141
2nd   Male            154  25
      Female           13  93
3rd   Male            422  88
      Female          106  90
Crew  Male            670 192
      Female            3  20
```

>ftable(Titanic, row.vars = 2:1, col.vars = "Survived")

> ftable(Titanic, row.vars = 2:1, col.vars = "Survived") #El mismo resultado anterior cambiando el orden de presentación de las variables ubicadas por filas

```
             Survived  No Yes
Sex    Class
Male   1st            118  62
       2nd            154  25
       3rd            422  88
       Crew           670 192
Female 1st              4 141
       2nd             13  93
       3rd            106  90
       Crew             3  20
```

Ejemplo 2 con input un data.frame

> data(quine,package="MASS")
> quine

```
    Eth Sex Age Lrn Days
1     A   M  F0  SL    2
2     A   M  F0  SL   11
3     A   M  F0  SL   14
4     A   M  F0  AL    5
….

144   N   F  F3  AL   18
145   N   F  F3  AL   22
146   N   F  F3  AL   37
```

```
> ftable(quine$Sex, quine$Age,row.vars=2)

     F  M

F0  10 17
F1  32 14
F2  19 21
F3  19 14

> tquine=ftable(quine$Sex, quine$Age,row.vars=2)
> ftable(tquine,row.vars=2)

   F0 F1 F2 F3

F  10 32 19 19
M  17 14 21 14

> ftable(tquine,row.vars=1:2)

F0 F  10
   M  17
F1 F  32
   M  14
F2 F  19
   M  21
F3 F  19
   M  14
```

table(dataframeovector,…)

Ejemplo con los datos del data.frame quine del package MASS

```
> data(quine,package="MASS")
> quine

    Eth Sex Age Lrn Days
1     A   M  F0  SL    2
2     A   M  F0  SL   11
3     A   M  F0  SL   14
4     A   M  F0  AL    5
….

144   N   F  F3  AL   18
145   N   F  F3  AL   22
146   N   F  F3  AL   37

> table(quine$Sex) #con el vector Sex del data.frame quine realiza una distribución de frecuencias

 F  M
80 66
```

Nota: Observe que **ftable** no acepta 1 sólo vector, ya que construye tablas de contingencia, por tanto, debe haber al menos dos variables cualitativas.

```
> tabla=ftable(quine$Sex, quine$Age,row.vars=2)
> tabla

     F  M
F0  10 17
F1  32 14
F2  19 21
F3  19 14

> as.data.frame(tabla)  #Convierte en un data.frame la tabla. Las combinaciones de las modalidades de las variables cualitativas y una nueva columna con las frecuencias.

  Var1 Var2 Freq
1   F0    F   10
2   F1    F   32
3   F2    F   19
4   F3    F   19
```

```
5   F0    M   17
6   F1    M   14
7   F2    M   21
8   F3    M   14
```

xtabs(), summary(), chisq.test(), se verán en carpeta 2 sobre tabulación

5.2. Resúmenes sobre tablas

tapply(*vectorparacálculos*, *listadefactores*, *funciónaplicada*) Realiza el cálculo de una función para subgrupos de datos. Los definidos mediante las combinaciones de los valores de los factores.

Ejemplo con los datos del data.frame quine

En el ejemplo se calcula en el vector Days la media para cada grupo de edad.

tapply(Days,Age,mean)

> quine

```
    Eth Sex Age Lrn Days
1     A   M  F0  SL    2
2     A   M  F0  SL   11
3     A   M  F0  SL   14
…

…
144   N   F  F3  AL   18
145   N   F  F3  AL   22
146   N   F  F3  AL   37
```

> quine[sapply(quine,is.factor)]

```
    Eth Sex Age Lrn
1     A   M  F0  SL
2     A   M  F0  SL
3     A   M  F0  SL
…
…
142   N   F  F3  AL
143   N   F  F3  AL
144   N   F  F3  AL
145   N   F  F3  AL
146   N   F  F3  AL
```

> table(quine[sapply(quine,is.factor)])

```
, , Age = F0, Lrn = AL

   Sex
Eth  F  M
  A  4  5
  N  4  6

, , Age = F1, Lrn = AL

   Sex
Eth  F  M
  A  5  2
  N  6  2

, , Age = F2, Lrn = AL

   Sex
Eth  F  M
…..
```

…

6. Funciones estadísticas

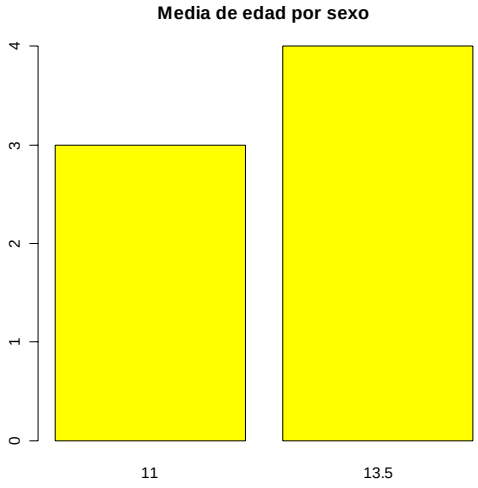
6.1. Resúmenes numéricos

ave(), cor(), median(), mean(), var(), sd(), range(), quartiles(), summary()

ave(*var*, *factor1*, *factor2*, …, FUN= *función*) genera una variable de la misma longitud que var que contiene el resultado de calcular una función, como por ejemplo media, mediana, mín, máx, etc., para la variable var, en los subgrupos formados por las combinaciones de los factores.

Ejemplo

```
> a
  edad sexo
1   12    h
2   14    h
3   13    h
4   15    h
5   10    m
6   11    m
7   12    m

> ave(a$edad,a$sexo,FUN=mean)
[1] 13.5 13.5 13.5 13.5 11.0 11.0 11.0

> media=ave(a$edad,a$sexo,FUN=mean)
> barplot(table(media),col="yellow",main="Media de edad por sexo")
```



cor() Determina la correlación entre variables.Argumento: dos vectores o una matriz o data frame de 2 ó más columnas.

Ejemplo

```
> x=1:5
> y=rnorm(5,0,1)
> z=6:2
> m=data.frame(x,y,z)
> cor(m)
          x         y         z
x  1.000000  0.773032 -1.000000
y  0.773032  1.000000 -0.773032
z -1.000000 -0.773032  1.000000
>
> cor(x,y)
[1] 0.773032
> cor(x,z)
[1] -1
>
```

IQR(*x*) Calcula el rango intercuartílico del vector x

mea() Calcula la media de un vector o para varias variables de un data frame

Ejemplo

```
> x
[1] 1 2 3 4 5
> m
  x         y z
1 1 -0.67125928 6
2 2 -0.25744404 5
3 3  1.11590375 4
4 4  0.09495986 3
5 5  1.16346747 2
> mean(x)
[1] 3
> mean(m)
       x        y        z
3.0000000 0.2891256 4.0000000
>
```

median() Calcula la mediana de un vector o para varias variables de un data frame

quantile() Calcula los cuatiles de una variable para un vector de órdenes. Si se omite, determina cuartiles.

Ejemplo

```
> x=rnorm(100,30,5)
> quantile(x,probs=seq(0,1,.1)) #Cálculo de los deciles de la variable x, incluyendo mín y máx de x

      0%      10%      20%      30%      40%      50%      60%      70%
17.32247 24.15807 26.12608 27.85702 29.21221 30.23434 31.19030 32.32757
     80%      90%     100%
33.41947 36.21479 41.97044
```

var() Calcula la varianza de un vector o para varias variables de un data frame

weighted.mean() Calcula una media ponderada

weighted.mean(*medias*,*pesos*) Calcula una media ponderada para un vector de medias y otro de los correspondientes pesos o ponderaciones

Ejemplo

```
> medias=c(2,7,3) vector de medias
> grupos=c(100,4,60) vector de pesos
> weighted.mean(medias,grupos)
[1] 2.487805

> grupo=grupos/sum(grupos)
> grupo
[1] 0.60975610 0.02439024 0.36585366

> weighted.mean(medias,grupo)
[1] 2.487805
```

6.2. Distribuciones de modelos de probabilidad

Normal (norm), binomial (binom), poison (pois), uniforme (unif), exponencial (exp), t de Student (t), Chi-cuadrado (chisq), etc.

Un modelo de probabilidad de una variable aleatoria tendrá información sobre: función de densidad o probabilidad (dnormal, dbinom, dunif, dpois, …), función de distribución (pnormal, punif, …), cuantiles (qnormal, qpois, qunif, …), y generación aleatoria de muestras (rnomal, rpois, ...).
dnombremodelo función de densidad, f(x), para un valor x de la variable,
pnombremodelo permite obtener p[X < x] para un valor de x. Si el argumento lower.tail=F calcula la probabilidad del suceso contrario (cola a la derecha)
qnombremodelo cuantil de un orden p
rnombremodelo muestra aleatoria de tamaño n.
Para modelo binomial, por ejemplo, los argumentos size y prob son número de pruebas y probabilidad de éxito, respectivamente:
dbinom(*x, size, prob*,…)
pbinom(*q, size, prob*, lower.tail = TRUE,…)
qbinom(*p, size, prob*, lower.tail = TRUE,…)
rbinom(*n, p, size, prob*, lower.tail = TRUE,…)# El parámetro n es el tamaño de muestra

ANEXO. Algunas funciones prácticas usadas con frecuencia

A.1. Funciones que permiten conocer características de los objetos

class(), dim(), nrow(), ncol(), names(), length(), is.na(),complet.cases()

Ejemplos

> class(m) informa que el objeto es un data frame

```
[1] "data.frame"
```

> class(x) informa que el objeto es de clase numérica

```
[1] "numeric"
```

```
> b=as.matrix(m)
> class(b)
[1] "matrix"
> dim(m)
[1] 5 3
> dim(b)
[1] 5 3
> ncol(m)
[1] 3
> ncol(b)
[1] 3
> names(m)
[1] "x" "y" "z"
> names(b)
NULL
>
> length(m$z)
[1] 5
```

A.2. Generación de información

La información en R se estructura en objetos con nombres. El objeto más simple es un vector. Un vector es una serie de valores estructurados con un nombre.
Para crear un vector simplemente asignamos un nombre y establecemos la relación de valores que lo constituyen.

Ejemplos

v1=1:10 crea un vector de nombre v1 con los valores 1 a 10

```
> v1=1:10
> v1
 [1]  1  2  3  4  5  6  7  8  9 10
```

v1=c(1,2,3,4,5,6,7,8,9) Otro modo equivalente de crear v1

v2=1/v1 crea un vector de nombre v2 con los valores inversos de v1

```
> v2=1/v1
> v2
 [1] 1.0000000 0.5000000 0.3333333 0.2500000 0.2000000 0.1666667 0.1428571
 [8] 0.1250000 0.1111111 0.1000000
```

v3=c(v1,v2,4) crea un vector de nombre v3 con los valores de v1, los de v2 y el valor 4

```
> v3=c(v1,v2,4)
> v3
 [1]  1.0000000  2.0000000  3.0000000  4.0000000  5.0000000  6.0000000
 [7]  7.0000000  8.0000000  9.0000000 10.0000000  1.0000000  0.5000000
[13]  0.3333333  0.2500000  0.2000000  0.1666667  0.1428571  0.1250000
[19]  0.1111111  0.1000000  4.0000000
```

v1=sqrt(v1) sustituye los valores de v1 por las correspondientes raiz cuadrada

```
> v1=sqrt(v1)
> v1
 [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427
 [9] 3.000000 3.162278
>
```

v1=v1^2 recupera los valores originales elevando al cuadrado

```
> v1=v1^2
> v1
 [1]  1  2  3  4  5  6  7  8  9 10
```

Repetición:

rep(5,3) repite 3 veces el valor 5,

```
> rep(5,3)
[1] 5 5 5
>
> rep(v1,3)  repite 3 veces el vector v1
 [1] 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5
[26] 6 7 8 9 10
```

rep(v1, each=3) Repite cada valor del vector v1 3 veces

```
> rep(v1, each=3)
 [1]  1  1  1  2  2  2  3  3  3  4  4  4  5  5  5  6  6  6  7  7  7  8  8  8  9
[26]  9  9 10 10 10
```

Secuencias:
seq() generación de secuencias

seq(-1,1,0.3) genera valores desde -1 hasta un máximo de 1 a intervalos de 0.3

```
> seq(-1,1,0.3)
[1] -1.0 -0.7 -0.4 -0.1  0.2  0.5  0.8
```

A.3. Operadores aritméticos

En R se pueden realizar las operaciones aritméticas usuales de cualquier calculadora como suma (+), resta (-), división (/), multiplicación (*), potenciación (^), así como las funciones de exponenciación (exp), logaritmos neperianos (log), logaritmo en base 10 (log10), raíz cuadrada (sqrt), entre otras.

A.4. Operadores lógicos

Producen resultados de verdadero o falso.
Los operadores son <, <=, >, >=,== y != (distinto).
Así como las operaciones de intersección (&), unión (|) y negación o contrario (!) realizadas con expresiones lógicas (verdadero o falso)

Ejemplo

```
> X=c(1,2,3)
> Y=c(3,2,1)
> X==Y
[1] FALSE  TRUE FALSE

> X>=Y
[1] FALSE  TRUE  TRUE

> (X==Y) | (X>=Y)
[1] FALSE  TRUE  TRUE

> (X==Y) & (X>=Y)
[1] FALSE  TRUE FALSE
```