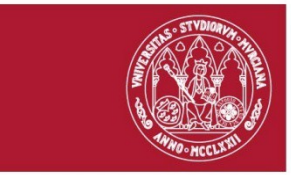


UNIVERSIDAD DE
MURCIA

# Taller de gráficos

Servicio de Apoyo a la Investigación
Universidad de Murcia.

Carlos Balsalobre (cbalsalobre@umh.es)
www.um.es/sai  www.um.es/ae

# CONTENIDO:

- **Realizando gráficos con R**
- **Funciones gráficas principales**
- **Comandos para la graficación de "bajo nivel"**
- **Función Par()**
- **Manejo de gráficos**
- **Ejemplos prácticos**
- **Bibliografía**

**Elementos de un gráfico:**

- Titulo principal
- Título secundario o Subtítulo
- Descripción del gráfico
- Región de datos y Símbolos
- Eje horizontal y Escala
- Eje Vertical y Escala
- Apuntadores
- Descripción de señales y marcas

REALIZANDO GRÁFICAS CON R

Si utilizamos el comando demo(graphics) podemos ver muchas de las posibilidades que nos ofrece R en lo que se refiere a la generación de gráficos.

Cuando realizamos una gráfica, esta es enviada a un dispositivo gráfico, el cual no es otra cosa que una ventana gráfica o un archivo.

Podemos diferenciar entre dos tipos de funciones gráficas: funciones de alto nivel que crean una nueva gráfica y las funciones de bajo nivel, que añaden elementos a una gráfica ya existente.

Vamos a comenzar con los gráficos más simples, iremos describiendo las funciones que utilizamos y poco a poco iremos realizando gráficos más complejos. Para ello vamos a crear tres vectores (coches, motos y camiones) que pasan por un punto señalizado.

```
> coches <- c(1, 3, 5, 5, 4, 9, 7)
> camiones <- c(2, 4, 4, 3, 5, 5, 11)
> motos <- c(1, 3, 3, 4, 3, 9, 14)
```

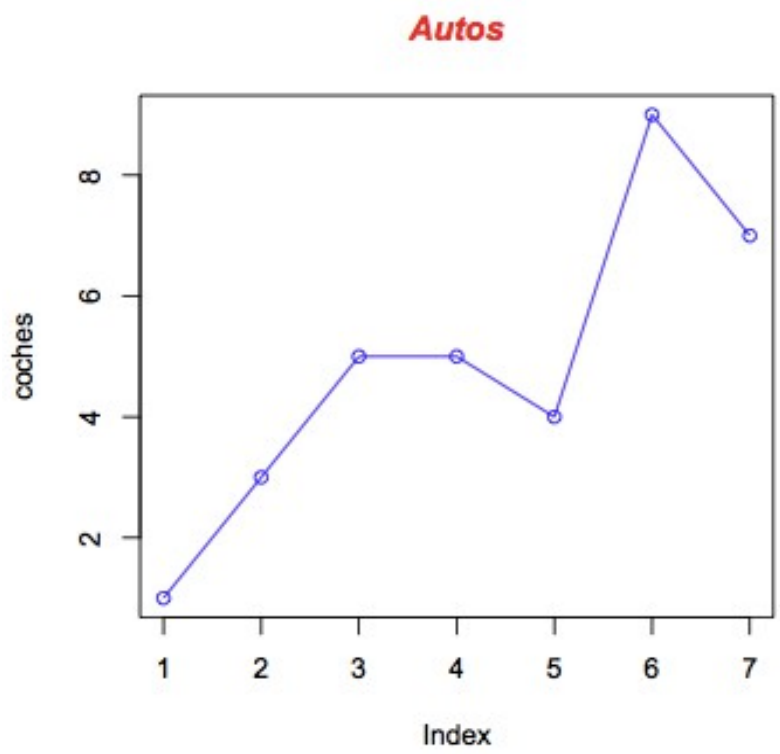
Las opciones y argumentos principales que podemos añadirle a cada gráfico (estas pueden consultarse en la ayuda de R)

- add= FALSE (defecto): Si es TRUE superpone el gráfico en el ya existente.
- axes= TRUE (defecto): si es FALSE no dibuja los ejes ni la caja del gráfico.
- type="p" (defecto): especifica el tipo de gráfico por ejemplo p= puntos, l= lineas, b= puntos conectados por lineas, h= lineas verticales.
- xlim=, ylim=: especifica los límites inferiores y superiores de los ejes; por ejemplo con xlim=c(1,5) o xlim=range(x)
- xlab=" ", ylab=" ": añade los títulos a los ejes.
- main=" ": añade el título principal.
- sub=" ": añade un subtitulo(letra más pequeña)
- col=: elegimos el color que deseamos para cada objeto.

```
> colors()
```

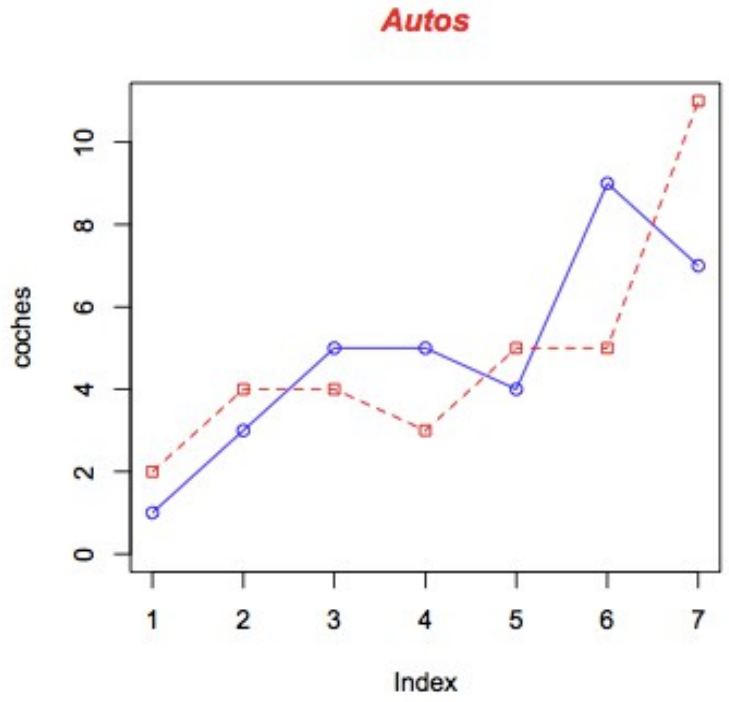
## Diagrama de líneas:

```
> plot(coches)
> plot(coches, type="o", col="blue")#Añadimos el "tipo de puntos" y color.
> title(main="Autos", col.main="red", font.main=4) #Añadimos un título en color rojo y
letra cursiva/negrita.
```



Ahora introduciremos al gráfico una línea roja para los camiones y le asignaremos el rango al eje y:

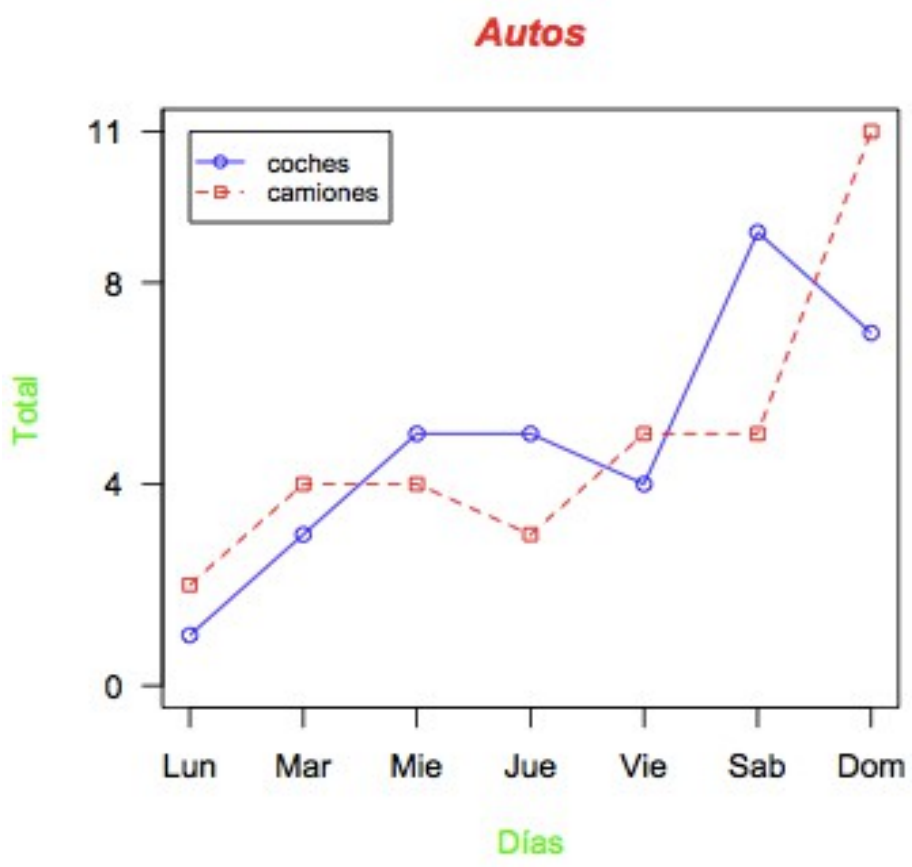
```
> plot(coches, type="o", col="blue", ylim=c(0, 11)) # Dibujamos coches añadiendole el rango del eje y (0,11)
> lines(camiones, type="o", pch=22, lty=2, col="red") # Dibujamos camiones con una linea intermitente roja y puntos cuadrados.
> title(main="Autos", col.main="red", font.main=4) #Creamos el título rojo, negrita y cursiva
```



A continuación vamos a cambiar las etiquetas de los ejes y añadiremos al gráfico una leyenda. También calcularemos los valores máximos de nuestro eje y y lo incorporaremos a nuestro gráfico.

```
> rango <- range(0, coches, camiones) #Calculamos el rango desde 0 al máximo valor de coches y camiones

> plot(coches, type="o", col="blue", ylim=rango, axes=FALSE, ann=FALSE) # Dibujamos Autos poniendo en el eje y el rango desde 0 hasta el máximo de vector coches o camiones. Además cerramos las etiquetas de los ejes y anotaciones para poner nosotros las que queramos.

> axis(1, at=1:7, lab=c("Lun", "Mar", "Mie", "Jue", "Vie", "Sab", "Dom")) # Introdocimos al eje x la etiqueta con los días de la semana.

> axis(2, las=1, at=rango[2])

> box() #creamos una caja alrededor del gráfico.

> lines(camiones, type="o", pch=22, lty=2, col="red") #Volvemos a dibujar camiones.

> title(main="Autos", col.main="red", font.main=4) #título

> #Etiquetamos los ejes y e y en color verde.

> title(xlab="Días", col.lab="green2")

> title(ylab="Total", col.lab="green2")

> #Vamos a añadir una leyenda al gráfico utilizando el tipo de linea y colores del gráfico.
```

```
> legend(1, rango[2], c("coches", "camiones"), cex=0.8, col=c("blue", "red"),
pch=21:22, lty=1:2)
```



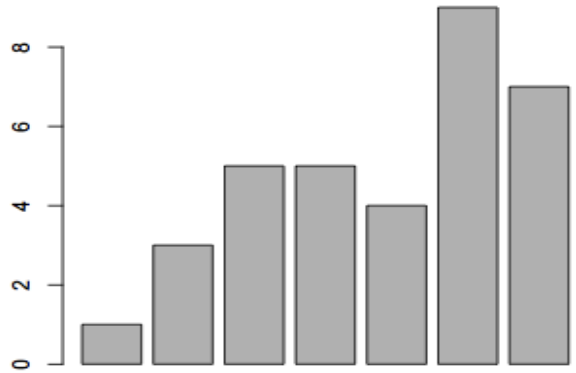
Diagramas de barras:

Seguimos utilizando los mismos vectores pero ahora vamos a crear un data frame:

```
> df <- data.frame(coches, camiones, motos)
> df
  coches camiones motos
1      1         2     1
2      3         4     3
3      5         4     3
4      5         3     4
5      4         5     3
6      9         5     9
7      7        11    14
```

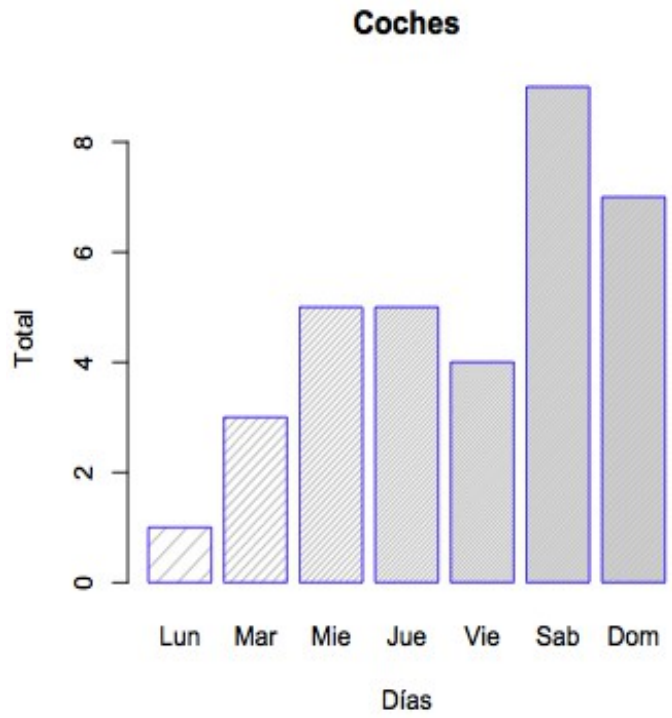
Realizamos un diagrama de barras para el vector coches:

```
> barplot(coches)
```



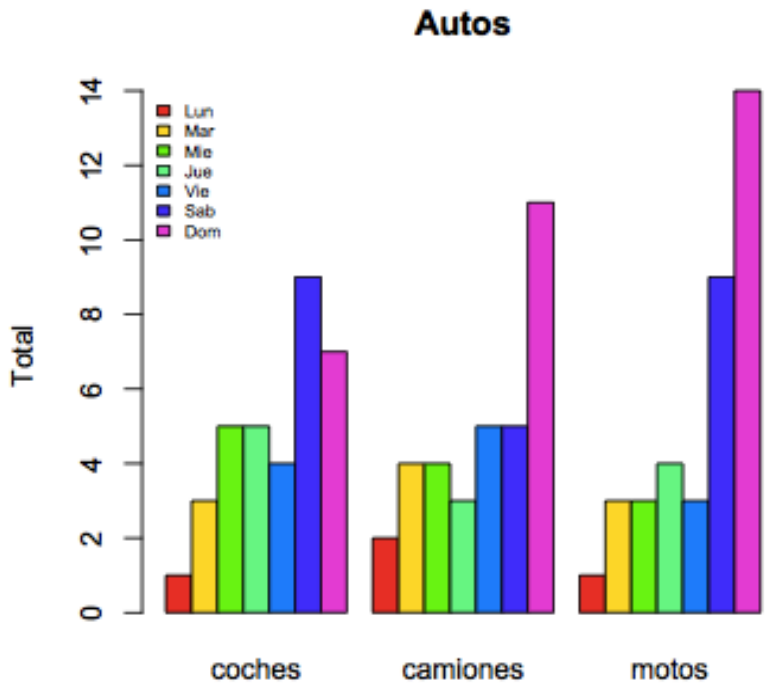
A continuación vamos a realizarle las siguientes modificaciones: añadiremos etiquetas, pondremos los bordes azules a las barras y líneas de densidad:

```
> barplot(df$coches, main="Coches", xlab="Días", ylab="Total", names.arg=c("Lun", "Mar", "Mie", "Jue", "Vie", "Sab", "Dom"), border="blue", density=c(10,20,30,40,50,60,70))
```



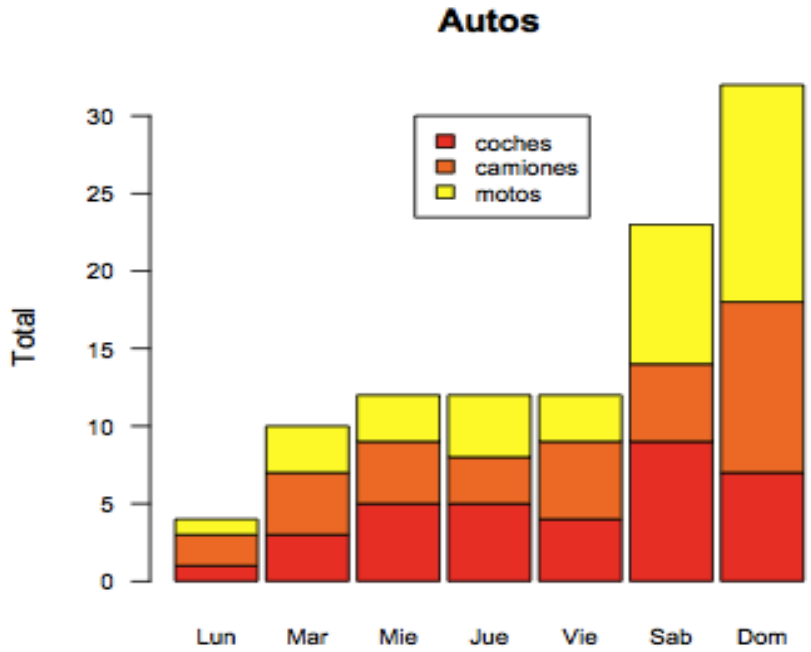
Y realizaremos un gráfico con el total de vehículos por día, usando diferentes colores y le añadiremos una leyenda.

```
> barplot(as.matrix(df), main="Autos", ylab="Total", beside=TRUE, col=rainbow(7))
>  legend("topleft",  c("Lun",  "Mar",  "Mie",  "Jue",  "Vie",  "Sab",  "Dom"),  cex=0.6,
bty="n",                                                          fill=rainbow(7))
```



Y para terminar con los gráficos de barras, los vamos a utilizar para describir el número de vehículos por día usando barras agrupadas y colocaremos la leyenda para este gráfico mediante coordenadas.

```
> barplot(t(df), main="Autos", ylab="Total",col=heat.colors(3),
space=0.1,cex.axis=0.8,las=1,names.arg=c("Lun","Mar","Mie","Jue","Vie", "Sab", "Dom"),
cex=0.8)
> legend(3, 30, names(df), cex=0.8, fill=heat.colors(3))
```

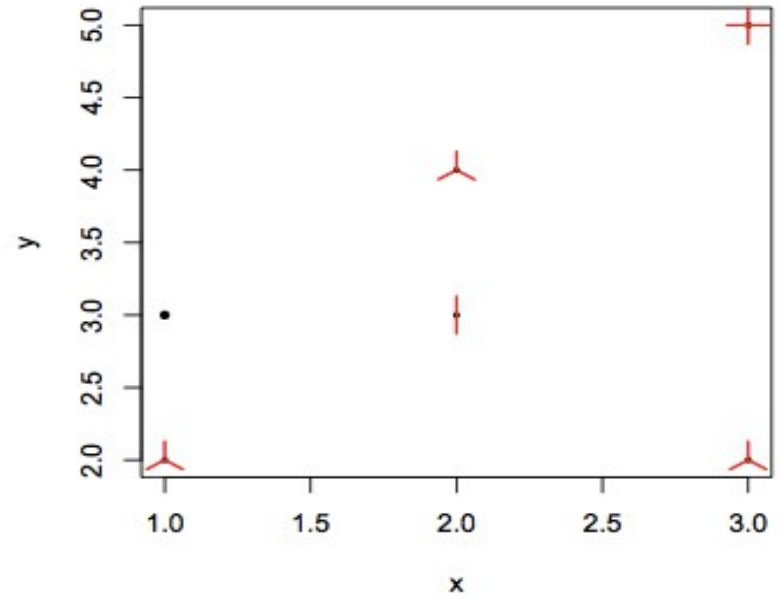


## FUNCIONES GRÁFICAS

Veamos a continuación algunas de las funciones gráficas mas utilizadas:

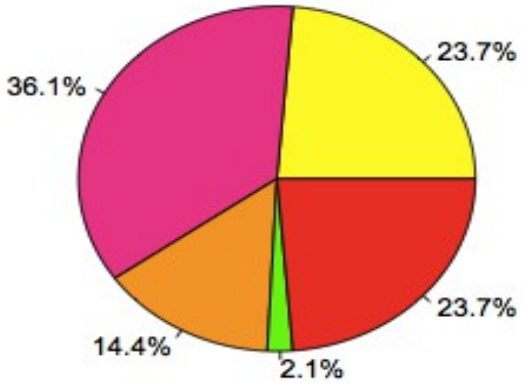
- `sunflowerplot(x, y)`: igual que plot(), pero los puntos de coordenadas similares se dibujan como flores con el número de pétalos igual al número de puntos.

```
> x <- c(1,1,1,1,2,2,2,2,2,3,3,3,3,3,3,3)
> y <- c(2,2,2,3,3,3,4,4,4,5,5,5,5,2,2,2)
> sunflowerplot(x,y)
```



- `pie()`: gráfico de sectores.

```
> pie.fruta <- c(0.23, 0.35, 0.14, 0.02, 0.23)
> names(pie.fruta) <- c("Platanos", "Cerezas", "Naranjas", "Manzanas", "Sandías")
> fruta_labels <- round(pie.fruta/sum(pie.fruta)*100, 1)
> fruta_labels <- paste(fruta_labels, "%", sep="")
>   pie(pie.fruta,   col=c("yellow",   "violetred1",   "orange",   "green",   "red"),
labels=fruta_labels)
```

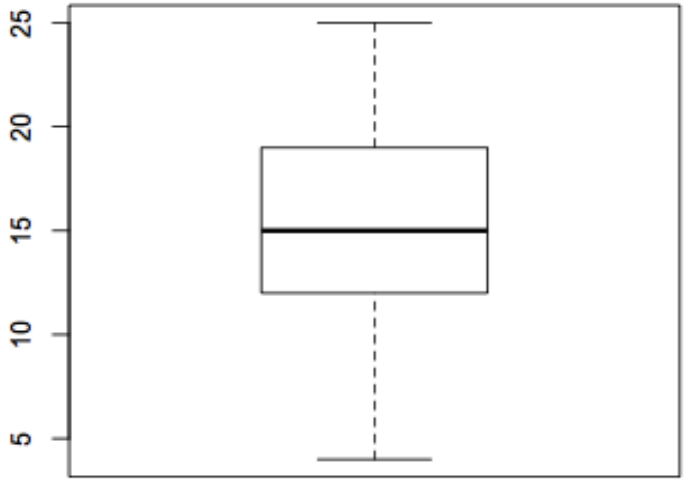



¿Cómo colocarías una leyenda a este gráfico?

```
> legend("topleft", c("Platanos", "Cerezas", "Naranjas", "Manzanas", "Sandías"),
cex=0.6, bty="n", fill=c("yellow", "violetred1", "orange", "green", "red"))
```

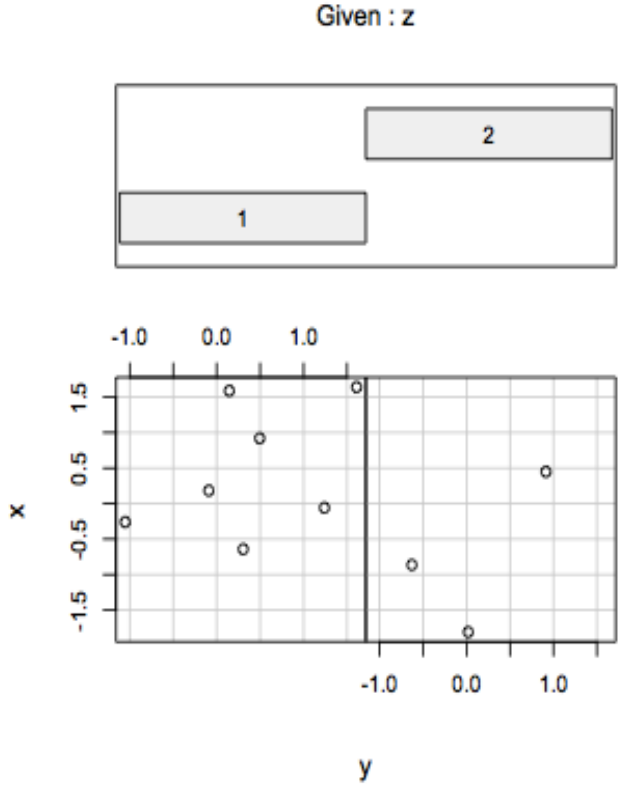
- `boxplot()`: gráfico de caja-bigote (después veremos más opciones)

```
> data(cars)
> attach(cars)
> boxplot(speed)
```



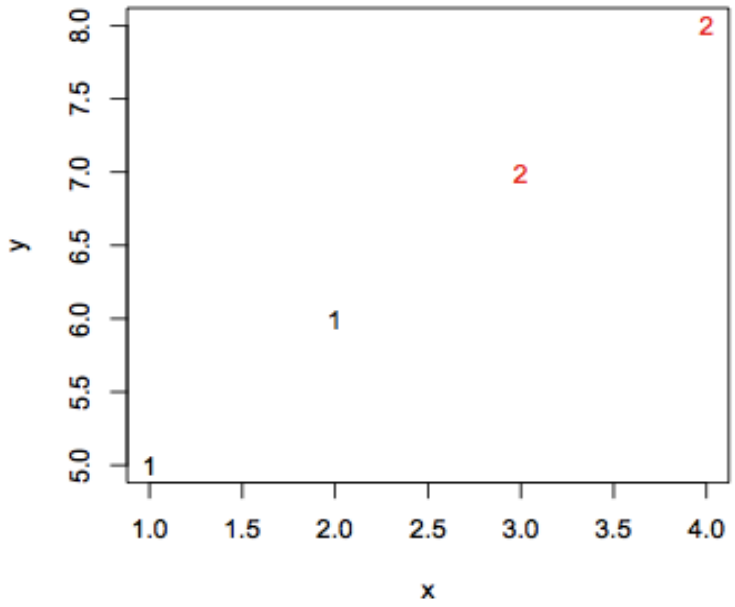
- `coplot(x~y|z)`: Gráfico bivariado de x e y para cada valor o intervalo de valores de z.

```
> x <- rnorm(10)
> y <- rnorm(10)
> z <- as.factor(c(1,1,2,1,1,1,2,1,2,1))
> z
 [1] 1 1 2 1 1 1 2 1 2 1
Levels: 1 2
> coplot(x~y|z)
```



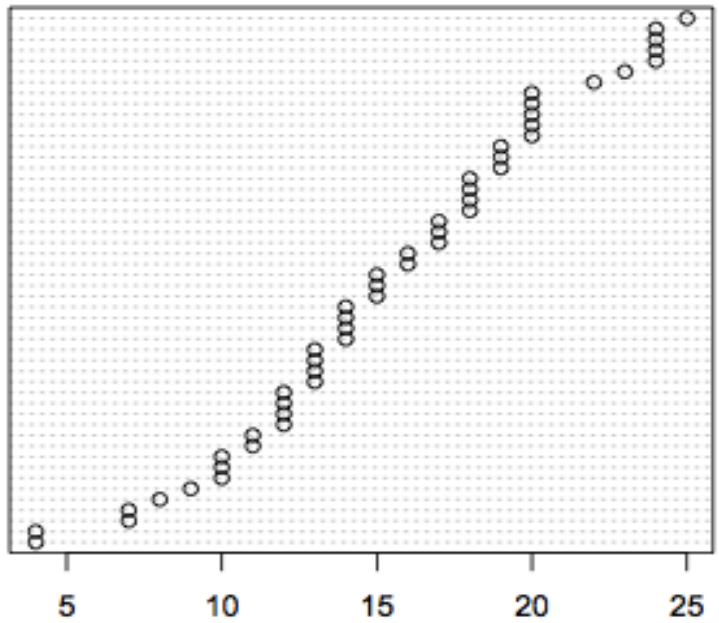
- `matplot(x,y)`: gráfica bivariada de la primera columna de x vs la primera columna de y, la segunda columna de x vs la segunda columna de y.

```
> x <-matrix(1:4,2,2)
> y <- matrix(5:8,2,2)
> matplot(x,y)
```



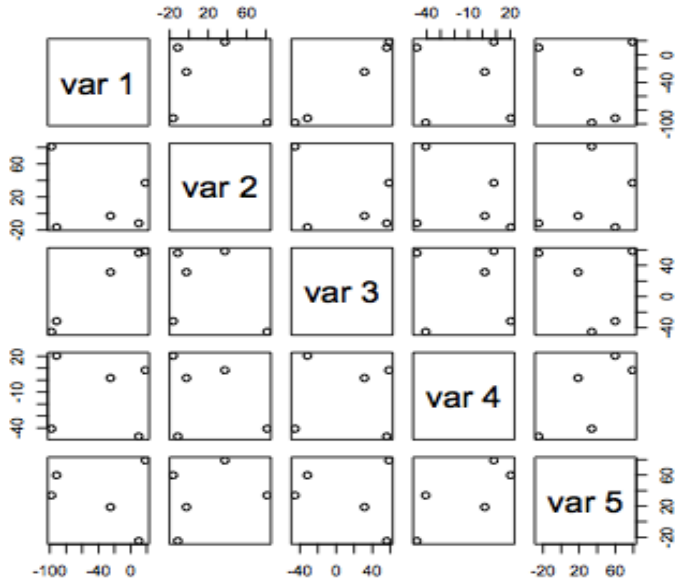
- `dotchart(x)`: Si x es un marco de datos, realiza gráficos apilados fila por fila y columna por columna.

```
> data(cars)
> attach(cars)
> dotchart(speed, data=cars)
```



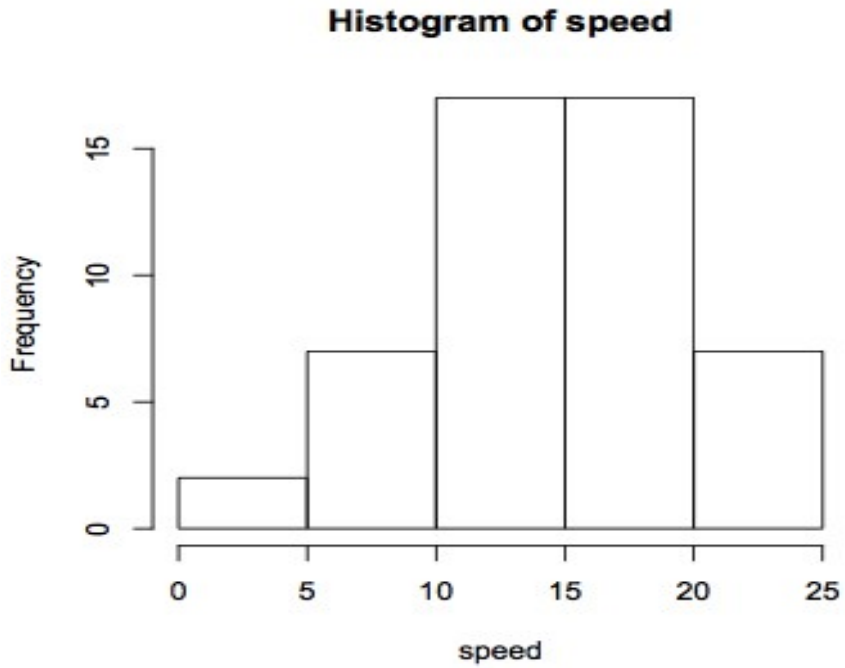
- `pairs(x)`: Si x es una matriz o un marco de datos, dibuja todas las posibles gráficas bivariadas entre las columnas de x.

```
> m=matrix(runif(25, -100, 100), 5, 5)
> m
           [,1]       [,2]      [,3]       [,4]      [,5]
[1,] -98.05804  81.069679 -45.19335 -40.600529  33.89534
[2,]  10.06733 -11.701413  55.75135 -46.998732 -24.95002
[3,] -91.70267 -16.430918 -31.53982  20.412409  59.85117
[4,]  18.23087  37.179871  58.22109   8.315046  79.01977
[5,] -24.96424  -2.767666  31.30015   1.800871  18.73845
> pairs(m)
```



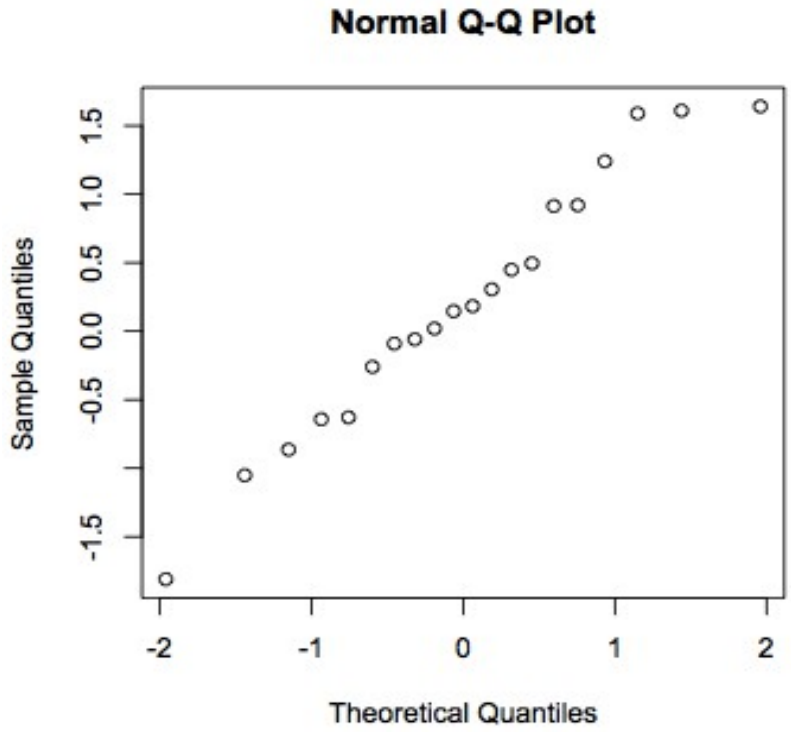
- `hist(x)`: histograma de las frecuencias de x

```
> hist(speed)
```



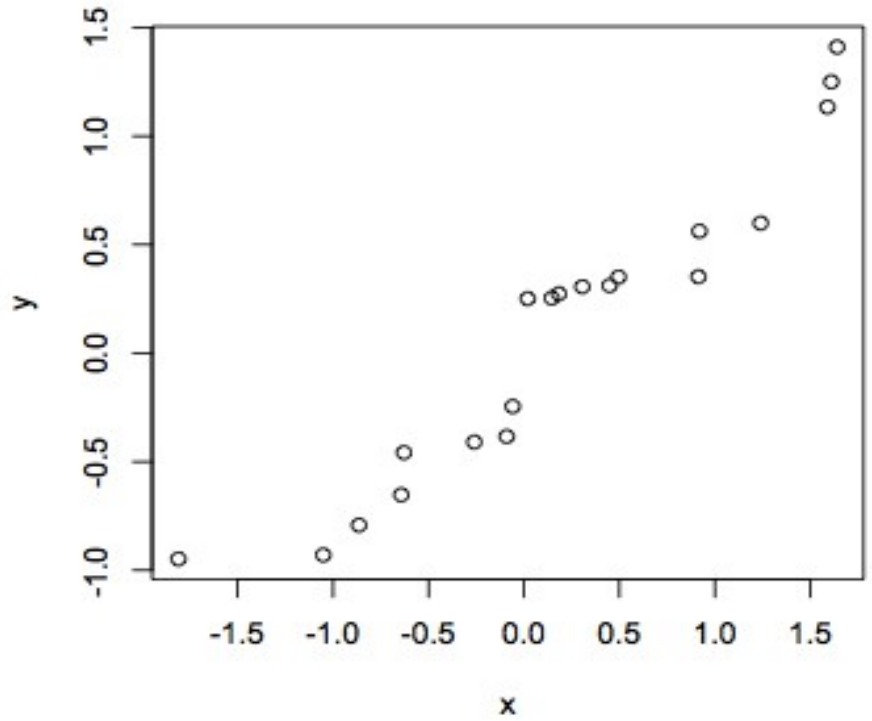
- `qqnorm(x)`: cuartiles de x con respecto a lo esperado bajo una distribución normal.

```
> x <- rnorm(20)
> y <- rnorm(20)
> qqnorm(x)
```



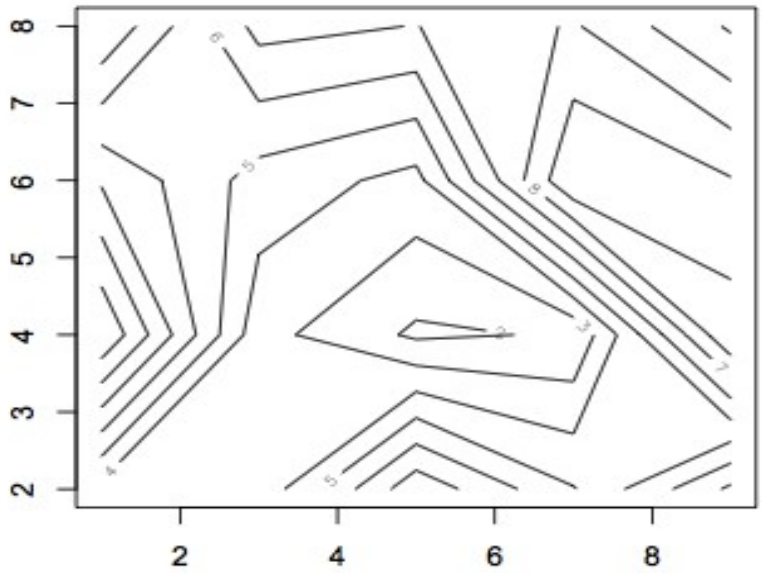
- `qqplot(x, y)`: cuartiles de y con respecto a los de x

```
> x <- rnorm(20)
> y <- rnorm(20)
> qqplot(x,y)
```



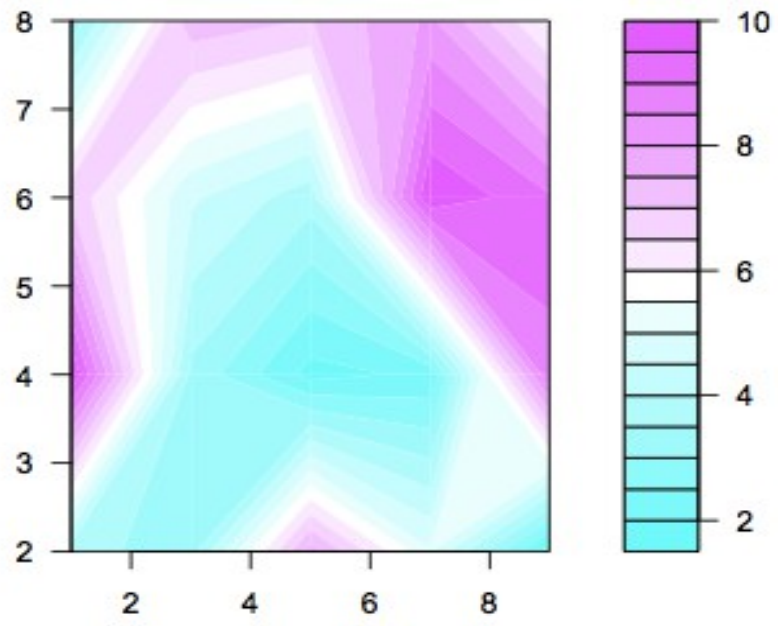
- `contour(x, y, z)`: gráfico de contornos (los datos son interpolados para dibujar las curvas), x e y deben ser vectores, z debe ser una matriz tal que dim(z)=c(length(x), length(y)).

```
> x <- c(1,3,5,7,9)
> y <- c(2, 4, 6, 8)
> z <- matrix(runif(20, 1, 10), 5, 4)
> contour(x, y, z)
```



- `filled.contour(x, y, z)`: igual que el anterior, pero las áreas entre contornos están coloreadas, y se dibuja una leyenda de colores.

```
> filled.contour(x, y, z)
```

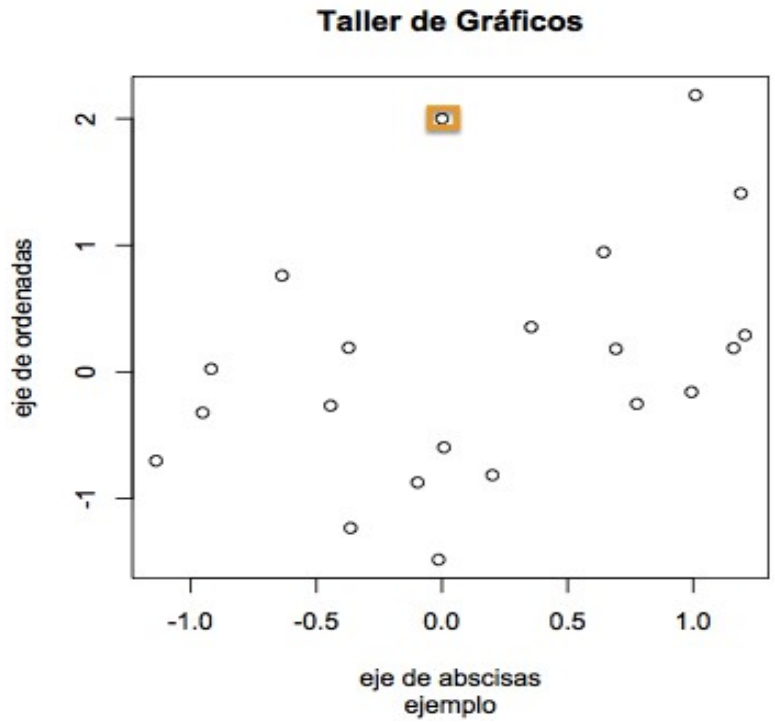


## COMANDOS PARA GRÁFICACIÓN DE BAJO NIVEL

Vamos a ver los más importantes:

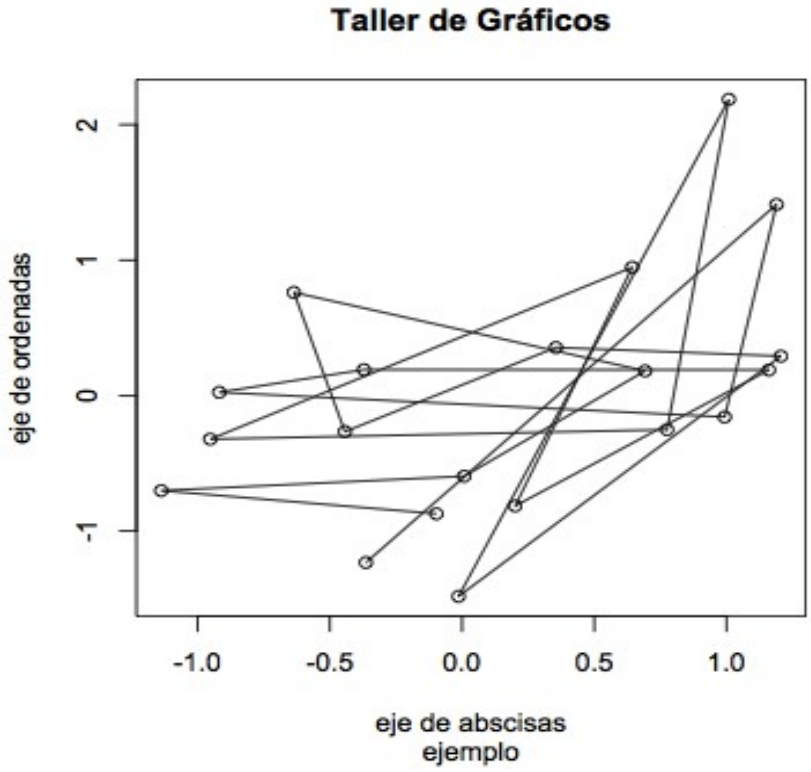
- `points(x, y)`: agrega puntos con coordenadas (x,y)

```
> x <- rnorm(20)
> y <- rnorm(20)
> plot(x, y, xlab="eje de abscisas", ylab="eje de ordenadas", main="Taller de
Gráficos", sub="ejemplo")
> points(0,2)
```



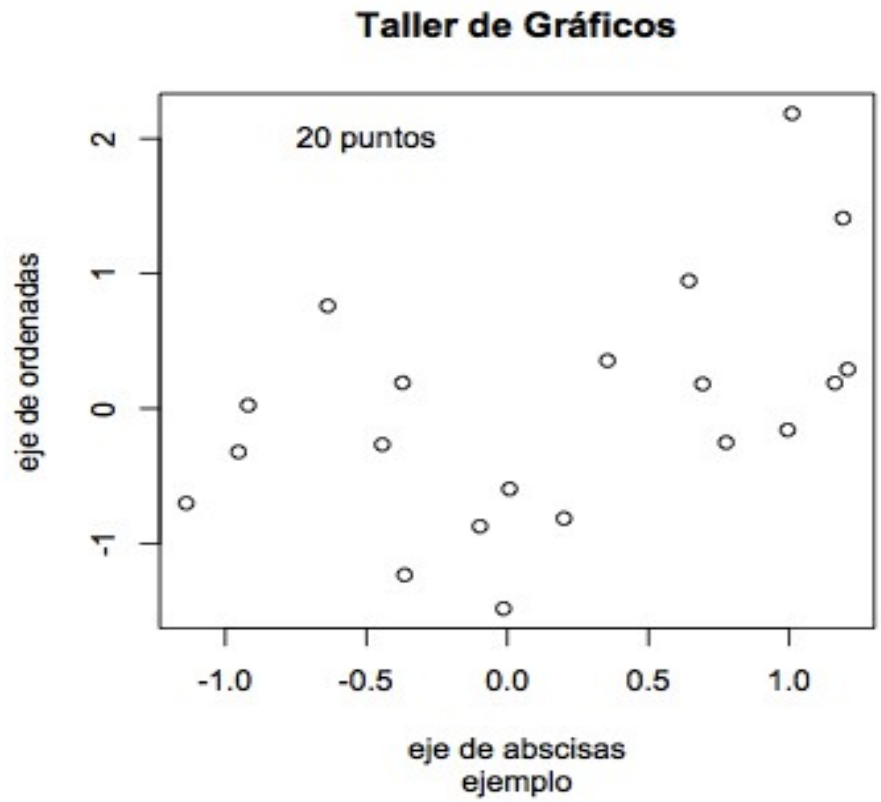
- `lines(x, y)`: igual a la anterior pero con lineas

```
> plot(x, y, xlab="eje de abscisas", ylab="eje de ordenadas", main="Taller de Gráficos", sub="ejemplo")
> lines(x,y)
```



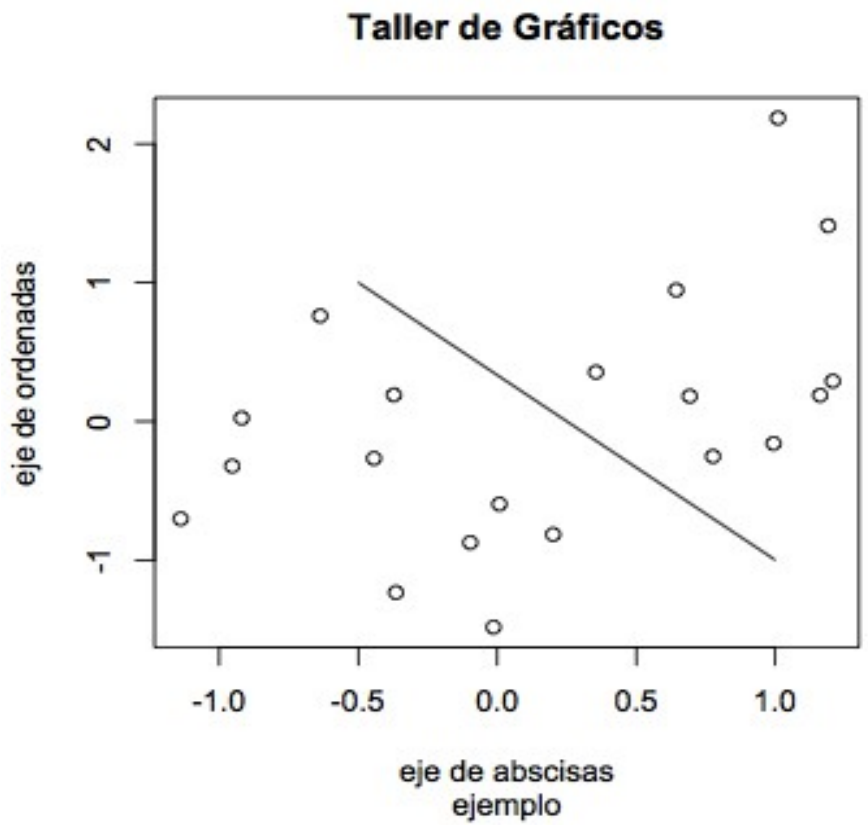
- `text(x, y, labels, …)`: agrega texto dado por labels en las coordenadas (x, y)

```
> plot(x, y, xlab="eje de abscisas", ylab="eje de ordenadas", main="Taller de Gráficos", sub="ejemplo")
> text(-0.5, 2, "20 puntos")
```



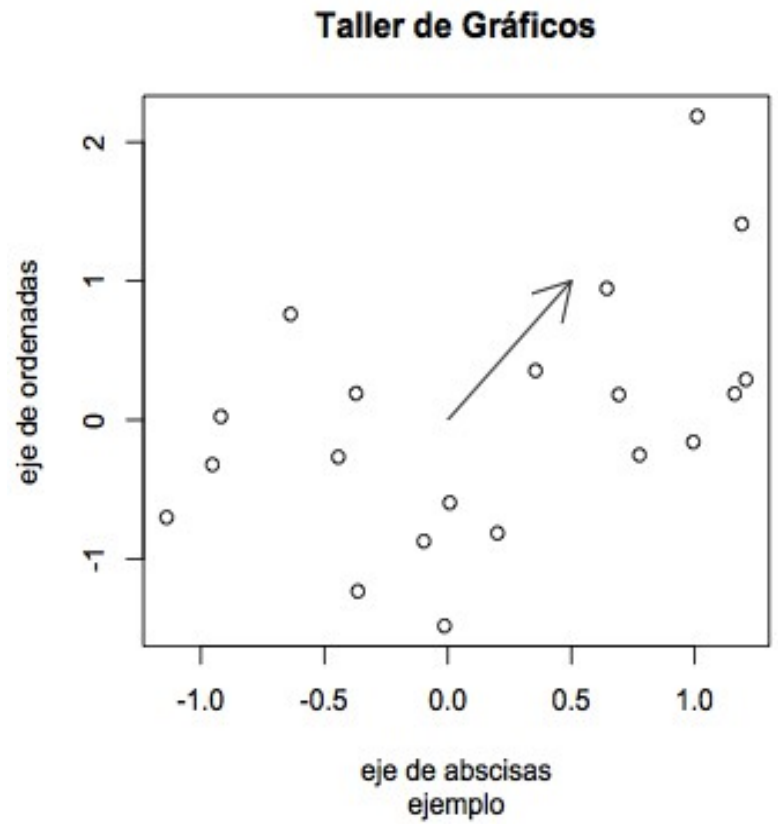
- `segments(x0, y0, x1, y1)`: Dibuja una linea desde el punto (x0,y0) a (x1, y1)

```
> plot(x, y, xlab="eje de abscisas", ylab="eje de ordenadas", main="Taller de Gráficos", sub="ejemplo")
> segments(-0.5,1,1,-1)
```



- `arrows(x0, y0, x1, y1)`: igual que el anterior pero con flechas.

```
> plot(x, y, xlab="eje de abscisas", ylab="eje de ordenadas", main="Taller de
Gráficos", sub="ejemplo")
> arrows(0,0,0.5,1)
```



- `abline(a, b)`: dibuja una linea con pendiente b e intercepto a

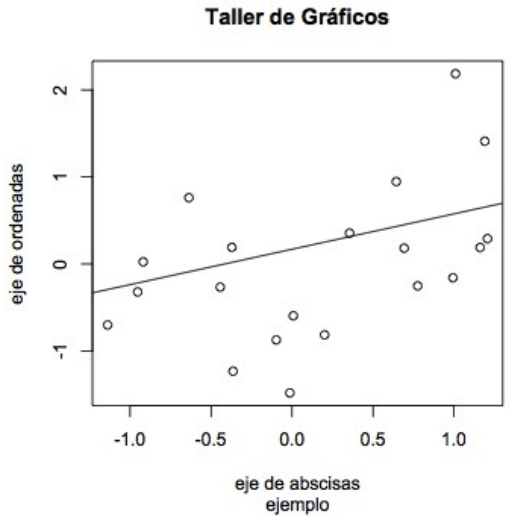
```
> plot(x, y, xlab="eje de abscisas", ylab="eje de ordenadas", main="Taller de Gráficos", sub="ejemplo")
> abline(1,0)
```

- `abline( h=y)`: dibuja una linea horizontal en la ordenada y
- `abline(v=x)`: dibuja una linea vertical en la abcisa x

```
> plot(x, y, xlab="eje de abscisas", ylab="eje de ordenadas", main="Taller de
Gráficos", sub="ejemplo")
> abline(h=2)
> abline(v=-0.5)
```

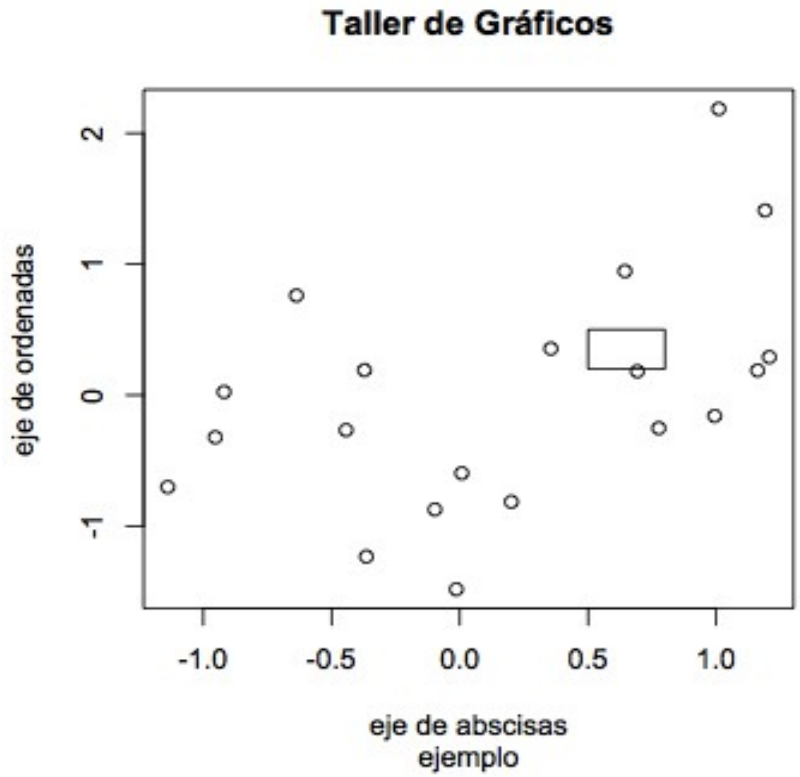
- `abline(lm.obj)`: dibuja la linea de regresión dada por lm.obj

```
> lm <- lm(x~y)
> lm

Call:
lm(formula = x ~ y)

Coefficients:
(Intercept)            y
     0.1684       0.4065

> plot(x, y, xlab="eje de abscisas", ylab="eje de ordenadas", main="Taller de
Gráficos", sub="ejemplo")
> abline(lm)
```



- `rect(x1, y1, x2, y2)`: Dibuja un rectángulo donde las esquinas izquierda, derecha, superior e inferior están dadas por x1, x2, y1, y2 respectivamente.

```
> plot(x, y, xlab="eje de abscisas", ylab="eje de ordenadas", main="Taller de Gráficos", sub="ejemplo")
> rect(0.5,0.2,0.8,0.5)
```

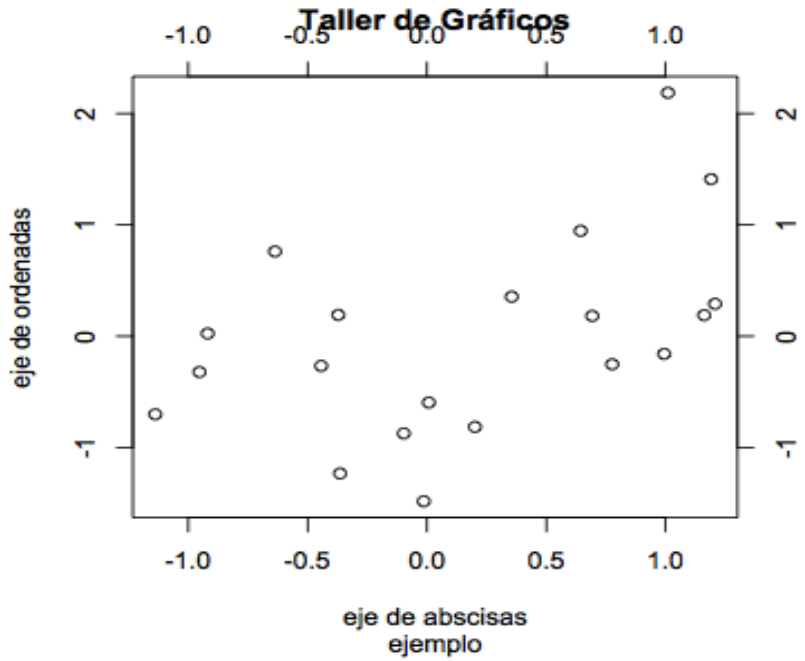


- `title()`: agrega un título y opcionalmente un subtítulo.

```
> plot(x,y)
> title("Taller de Gráficos", "Ejemplo")
```

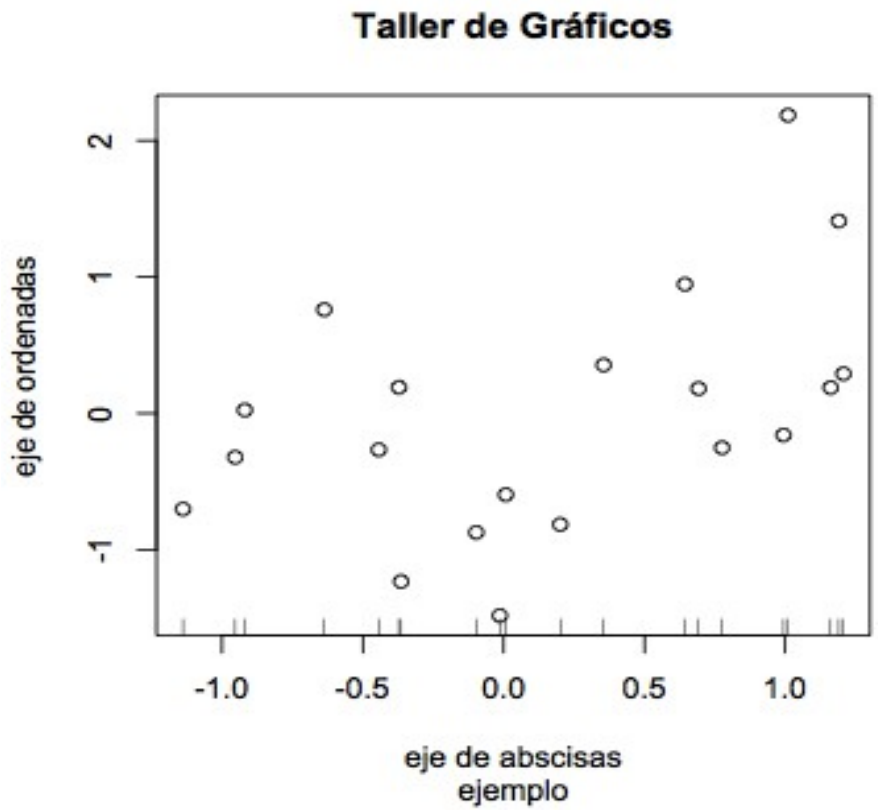
- `axis(side=)`: agrega un eje en la parte inferior (side=1), izquierda(2), superior (3) o derecha (4).

```
> plot(x, y, xlab="eje de abscisas", ylab="eje de ordenadas", main="Taller de
Gráficos", sub="ejemplo")
> axis(side=3)
> axis(side=4)
```



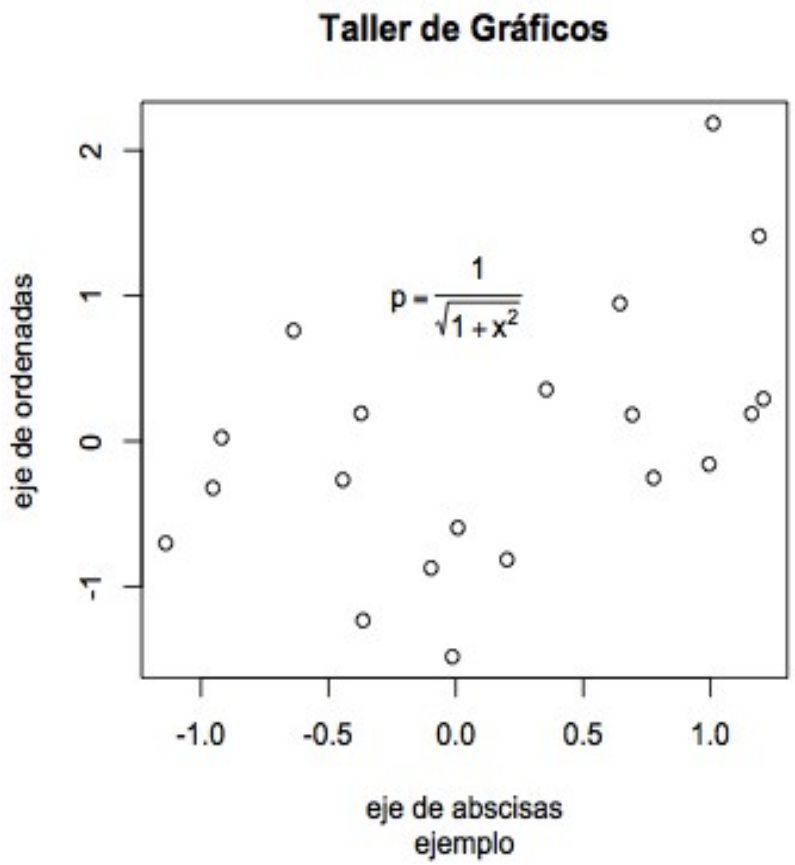
- `rug(x)`: dibuja los datos x en el eje x como pequeñas líneas verticales.

```
> plot(x, y, xlab="eje de abscisas", ylab="eje de ordenadas", main="Taller de Gráficos", sub="ejemplo")
> rug(x)
```



También podemos añadir expresiones matemáticas a una gráfica con el comando text(x, y, expression(...)), donde la función expression transforma su argumento en una ecuación matemática. Veamos un sencillo ejemplo:

```
> plot(x, y, xlab="eje de abcisas", ylab="eje de ordenadas", main="Taller de
Gráficos", sub="ejemplo")
> text(0,1, expression(p==over(1,sqrt(1+x^2))))
```



## PARÁMETROS GRÁFICOS (FUNCIÓN PAR)

Además de la utilización de los gráficos de bajo nivel, la presentación de estos puede verse mejorada con parámetros gráficos adicionales. Estos se pueden utilizar como opciones aunque no funcionan para todas. Este procedimiento se puede realizar también utilizando la función `par()` que nos permite cambiar de forma permanente los parámetros gráficos. Veamos las principales posibilidades que nos ofrece esta función.

- adj: El valor que tome determina la forma en la cual las “cadenas” de texto son justificadas. 0 para justificar el texto a la izquierda, 0.5 para centrar el texto y 1 para justificarlo a la derecha.
- ann: Si lo ajustamos como FALSE entonces en las funciones de graficación de alto nivel no se realiza anotación sobre los ejes, sólo producen los ejes con sus títulos y el respectivo gráfico.
- ask: Si es TRUE, el usuario es preguntado por el imput antes que una nueva figura sea dibujada.
- bg: Nos especifica que color va a ser utilizado para los fondos de los gráficos.
- bty: Determina el tipo de caja que será dibujada alrededor de los gráficos “o”, “l”, “7”, “c”, “u”, “]”, “n”(suprime la caja)
- cex: Es un valor que controla el tamaño del texto y símbolos con respecto al valor por defecto. Podemos utilizar, cex.axis (números de los ejes), cex.lab (títulos de los ejes), cex.main (título principal) y cex.sub (subtítulo)
- cin: Tamaño del carácter “(ancho, alto)” en pulgadas
- col: controla el color de los símbolos al igual que podemos elegir el color de cada elemento del gráfico con los comandos: col.axis (ejes), col.lab (títulos de los ejes), col.main(título principal) y col.sub (subtítulo)
- fg: El color a ser usado para el primer plano de los gráficos. Es el color que por defecto es usado en objetos como los ejes y las cajas alrededor de los gráficos.
- font: Un entero que especifica la fuente a usar para el texto, de modo que 1 corresponde a texto plano, 2 corresponde a texto en negrita, 3 texto en ítalica, 4 texto itálica-negrita. Podemos utilizar, font.axis, font.lab, font.main, font.sub.

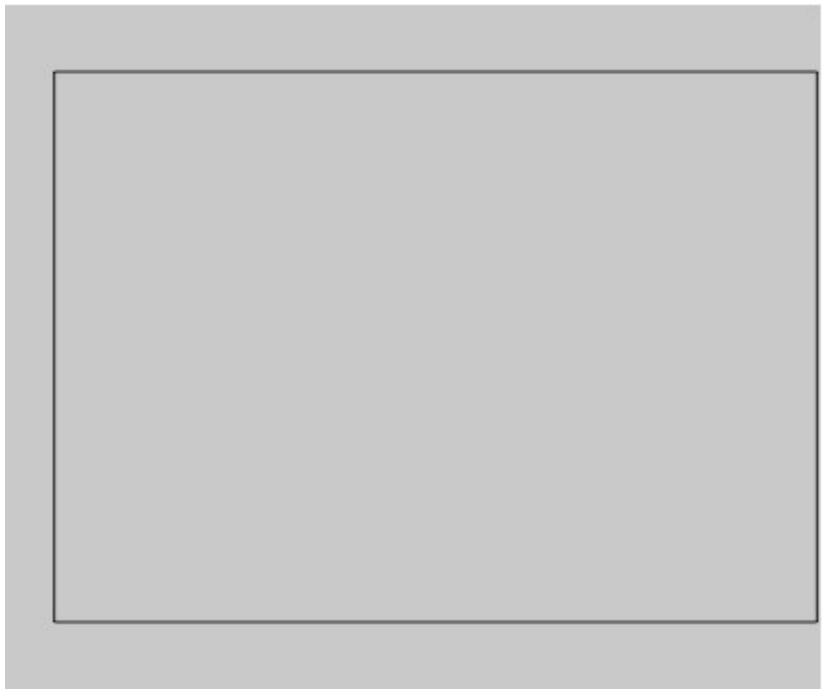
- las: Un número (0, 1, 2, 3), con el cual se especifica el estilo de las etiquetas de las anotaciones de los ejes: 0 (defecto) paralelo al eje, 1 horizontal, 2 perpendicular al eje, 3 vertical
- lty: Un entero que controla el tipo de las lineas (1: solida, 2: quebrada, 3: punteada, 4: punto-linea, 5: linea larga-corta, 6: dos lineas cortas)
- lwd: Un número positivo para determinar el ancho de linea (defecto=1)
- mar: Un vector con 4 valores numéricos que controla el espacio entre los ejes y el borde de la gráfica en la forma c(inferior, izquierda, superior, derecha) los valores por defecto son c(5,4,4,2) + 0.1
- mfcol: Es un vector del tipo c(nf,nc) que divide la ventana gráfica como una matriz cn nf filas y nc columnas. Las gráficas se van dibujando sucesivamente por columnas.
- mfrow: igual que el anterior pero las gráficas se dibujan por filas.
- pch: controla el tipo de símbolo con un entero entre 1 y 25.
- ps: Un entero que controla el tamaño (en puntos) de texto y símbolos.
- pty: Caracter entre comillas para especificar el tipo de región gráfica a ser usada: “s” región cuadrada, “m” región máxima
- xaxt: si xaxt = “n” el eje x se coloca pero no se muestra
- yaxt: si yaxt= “n” el eje y se coloca pero no se muestra

## Veamos un ejemplo con la función `par()`

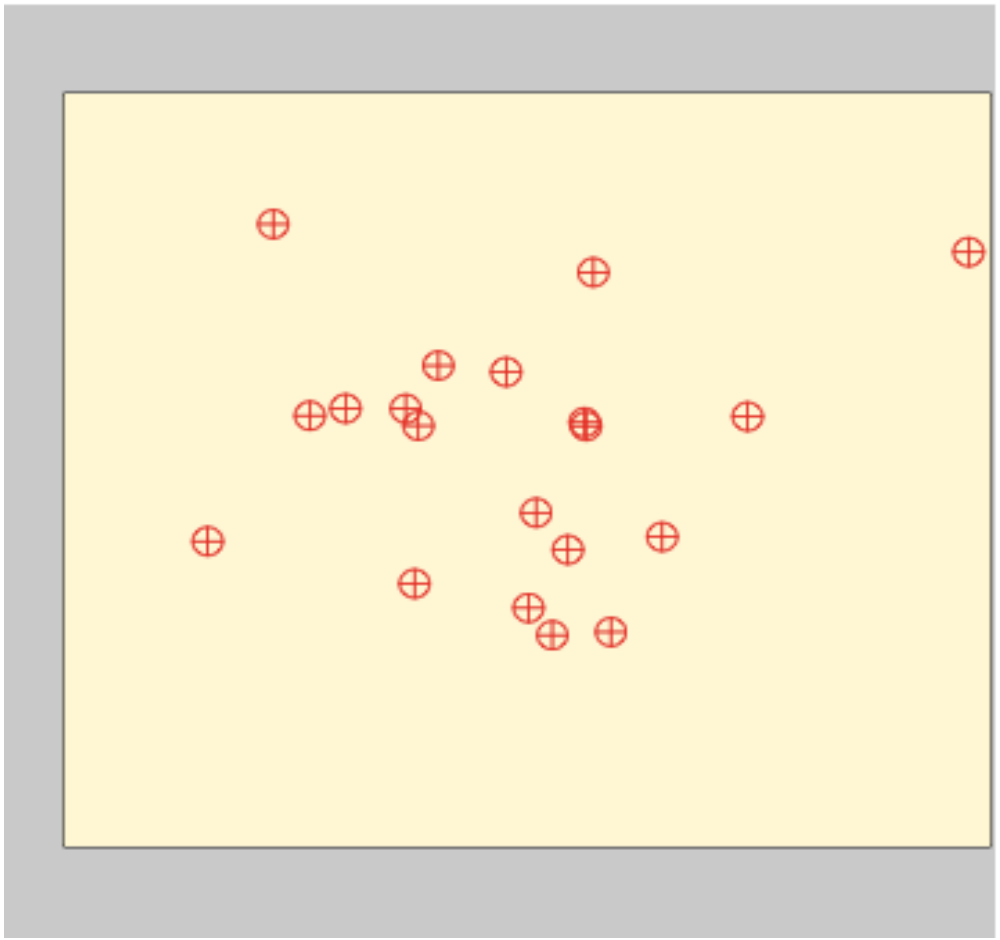
```
> x <- rnorm(20)
> y <- rnorm(20)
> opar <- par()
> par(bg ="lightyellow", col.axis="blue", mar=c(4, 4, 2.5, 2.5))
> plot(x, y, xlab="eje de abscisas", ylab="eje de ordenadas", xlim= c(-2,2), ylim=c(-
2,2), pch=22, col="red", bg="yellow", bty="l", las=1, cex=1.5)
> title("Realizando un gráfico en R", font.main=3, adj=1)
> par(opar)
```

## Ejemplo 2 (Vamos a realizar una gráfica desde cero)

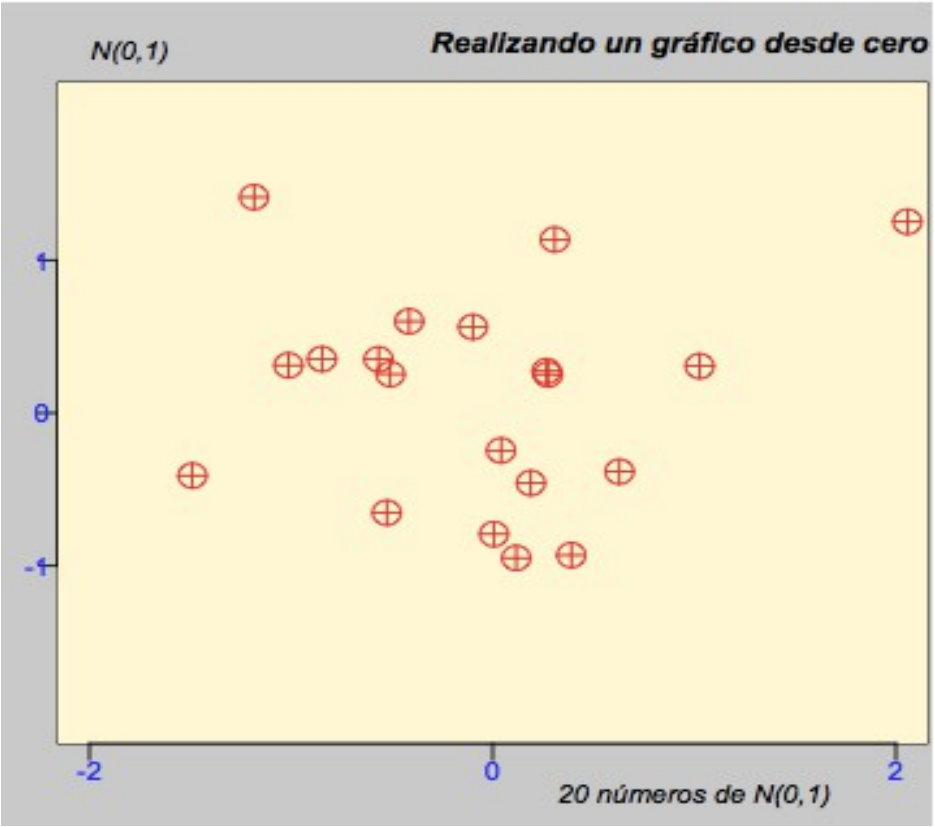
```
> x <- rnorm(20)
> y <- rnorm(20)
> opar <- par()
> par(bg= "lightgray", mar=c(2.5,1.5,2.5,0.25))
>  plot(x,  y,  type="n",  xlab="",  ylab="",  xlim=c(-2,2),  ylim=c(-2,2),  xaxt="n",
yaxt="n")
```



```
> rect(-3, -3, 3, 3, col="cornsilk")
> points(x, y, pch=10, col="red", cex=2)
```



```
> axis(side=1, c(-2,0,2), labels=FALSE)
> axis(side=2, -1:1, labels=FALSE)
> title("Realizando un gráfico desde cero", font.main=4, adj=1, cex.main=1)
> mtext("20 números de N(0,1)", side=1, line=1, at=1, cex=0.9, font=3 )
> mtext("N(0,1)", line=0.5, at=-1.8, cex=0.9, font=3)
> mtext(c(-2,0,2), side=1, las=1, at=c(-2,0,2), line=0.3, col="blue", cex=0.9)
> mtext(-1:1, side=2, las=1,at=-1:1, line=0.2, col="blue", cex=0.9)
> par(opar)
```



# MANEJO DE GRÁFICOS

Trabajando con múltiples dispositivos gráficos:

Las ventanas gráficas se llaman x11 y se puede abrir una nueva ventana gráfica con el comando x11(). Dependiendo del tipo de gráfico que se quiere crear podemos utilizar los comandos: postscript(), pdf(),...

La lista de dispositivos gráficos disponible se obtiene ejecutando el comando ?device

El último dispositivo abierto se convierte en el dispositivo activo, sobre el cual se van a dibujar las gráficas generadas.

La función dev.list() muestra una lista con los dispositivos abiertos: veamos un ejemplo para el manejo de dispositivos gráficos:

creamos los siguientes dispositivos gráficos:

```
> x11(); x11(); pdf()
> dev.list()
X11 X11 pdf
  2   3   4
```

Para saber cual es el dispositivo activo, cambiarlo o bien cerrarlo, utilizamos los siguientes comandos:

```
> dev.cur()
pdf
  4
> dev.set(3)
X11
```

```
  3
> dev.off(2)
X11
  3
> dev.off()
pdf
  4
```

Disposición de una Gráfica:

Vamos a ver las dos más importantes: split.screen() y layout()

La función split.screen() divide el dispositivo en dos partes que se pueden seleccionar con screen(1) y screen(2); el comando erase.screen() borra la última gráfica realizada. Además cada parte del dispositivo se puede dividir en partes más pequeñas utilizando el mismo comando. Vamos a trabajar un poco con esto, para ellos vamos a creamos las siguientes variables:

```
> x <- rnorm(20)
> y <- rnorm(20)
```

Vamos a dividir el dispositivo gráfico en 2 y cada uno de ellos en otros 2:

```
> split.screen(c(1,2))
[1] 1 2
> screen(1)
> split.screen(c(1,2))
[1] 3 4
> screen(2)
> split.screen(c(1,2))
[1] 5 6
```

También podemos crear directamente el número de particiones que queramos para nuestro dispositivo gráfico directamente con split.screen(c(filas, columnas)

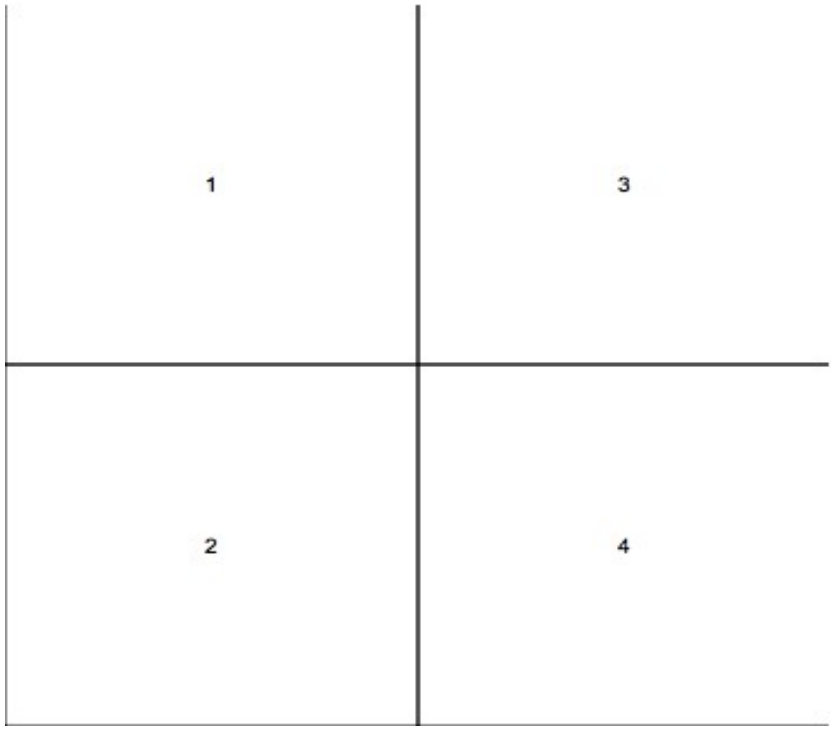
```
> split.screen(c(3,2))
[1] 1 2 3 4 5 6
> screen(4)
> plot(5)
> screen(6)
> plot(7)
```

La función layout() divide el dispositivo activo en varias partes donde se colocarán las gráficas de manera sucesiva. Esta función tiene como argumento principal una matriz con números enteros indicando el número de subventanas. Por ejemplo si queremos dividir el dispositivo en 4 partes iguales:

```
> layout(matrix(1:4,2,2))
> plot(1)
> plot(2)
> plot(5)
> plot(7)
```

Otra forma para ver previamente como se van a introducir las gráficas sería de la siguiente manera:

```
> mat <- matrix(1:4,2,2)
> mat
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> layout(mat)
> layout.show(4)
```

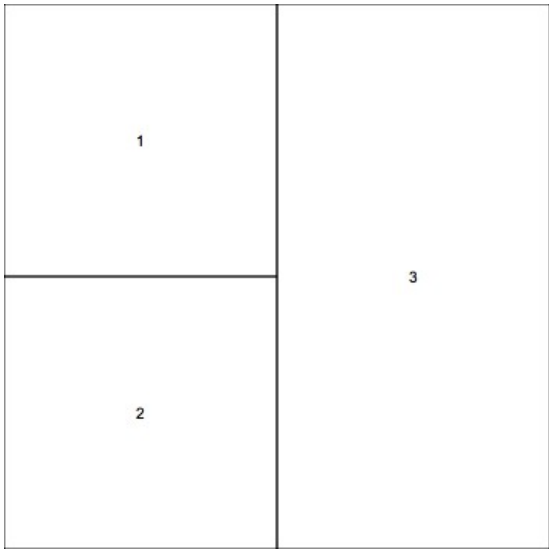


Veamos algunos ejemplos que nos muestran las posibilidades de la función layout():

```
> layout(matrix(1:6,3,2))
> layout.show(6)
```

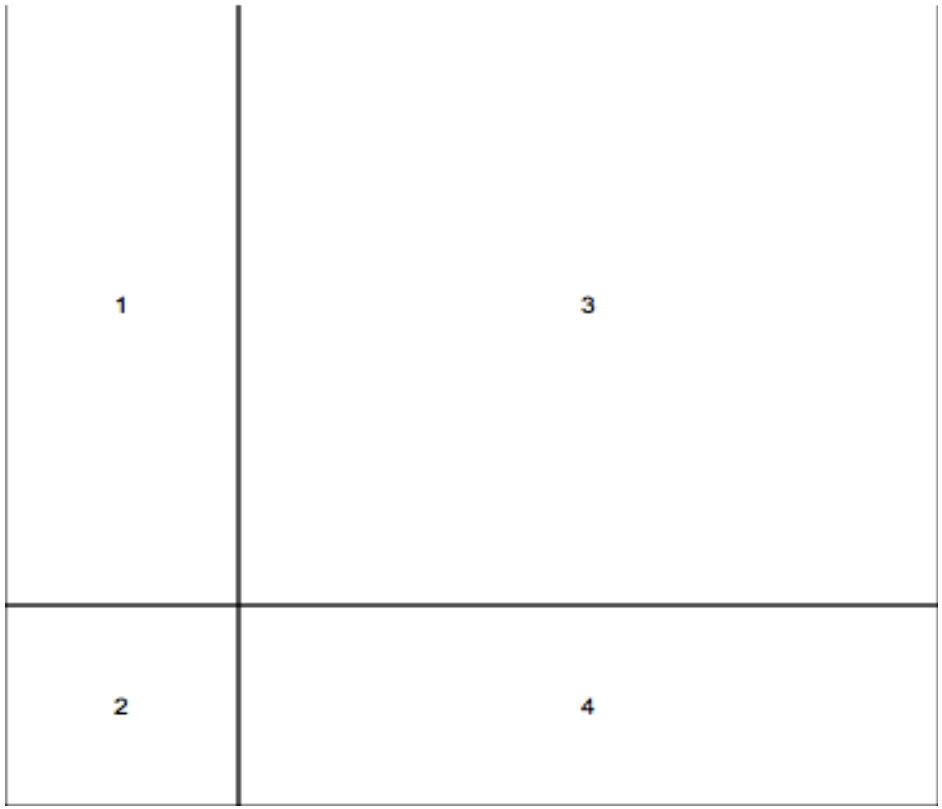
```
> layout(matrix(1:6,2,3))
> layout.show(6)
```

```
> m <- matrix(c(1:3,3),2,2)
> m
     [,1] [,2]
[1,]    1    3
[2,]    2    3
> layout(m)
> layout.show(3)
```



Por defecto layout() divide el dispositivo en dimensiones regulares, pero esto puede ser modificado con las opciones widths y heights:

```
> m <- matrix(1:4,2,2)
> layout(m, widths=c(1,3), heights=c(3,1))
> layout.show(4)
```



```
> m <- matrix(0:3,2,2)
> layout(m,c(1,3), c(1,3))
> layout.show(3)
```

Ejemplo:

Tenemos un conjunto de datos que nos muestran las personas que asistieron al cine, teatro y circo durante 5 semanas en las que se contabilizaron. Vamos a realizar un archivo que nos muestre tres gráficas distintas en el mismo dispositivo.

```
> pcine <- c(221, 323, 254, 198, 433)
> pteatro <- c(132, 221, 210, 97, 325)
> pcirco <- c(75, 110, 97, 86, 45)
> m <- matrix(c(1,1:3), 2, 2)
> layout(m)
> layout.show(3)
> df <- data.frame(pcine, pteatro, pcirco)
> df
  pcine pteatro pcirco
1   221     132     75
2   323     221    110
3   254     210     97
4   198      97     86
5   433     325     45
> boxplot(df, border=c("red", "yellow", "violetred1"), col="blue", main="Ventana
primera", sub="boxplot", ylab="n° de personas")
```

```
> rango <- range(pcirco, pteatro, pcine)
[1]  45 433
plot(df$pcine, type="o", col="red", main="Ventana 2ª",sub=”Dig. Lineas” ylim=rango,
ann=FALSE)
> lines(pteatro, type="o", col="yellow", pch=22, lty=2 )
> lines(pcirco, type="o", col="violetred1", pch=23, lty=3)
> pie.personas <- c(sum(pcine), sum(pteatro), sum(pcirco))
> names(pie.personas) <- c("cine", "teatro", "circo")
>  pie(pie.personas,  col=c("red",  "yellow",  "violetred1"),  main="Ventana  3ª",
sub="gr.sectores")
```

Ejercicio:

Generar dos variables aleatorias de 10 números cada una (normal y uniforme (-2,2)) y realizar un diagrama bivariado (plot) en el que aparezca a parte de los elementos principales (título, subtítulo, etiquetado de los ejes...), una flecha que vaya desde el punto (0,0) al punto (0.7,0.7) y añadir justo en esta coordenada otro punto (en color verde para distinguirlo). Añadir la palabra “punto verde” encerrada en una caja (rectángulo)

Añadirle en la esquina izquierda la fórmula y=1/sen(x).

Bibliografía:


- R para principiantes (Emmanuel Paradis)
- Gráficos Estadísticos con R (Juan Carlos Correa y Nelfi González)
- Producing Simple Graphs with R (Frank McCown)