[000Z] DEFAD: Representación y tabulación de datos

Gráficos en R

Elvira Ferre Jaén
elvira@um.es

Universidad de Murcia

Marzo 2018

Introducción

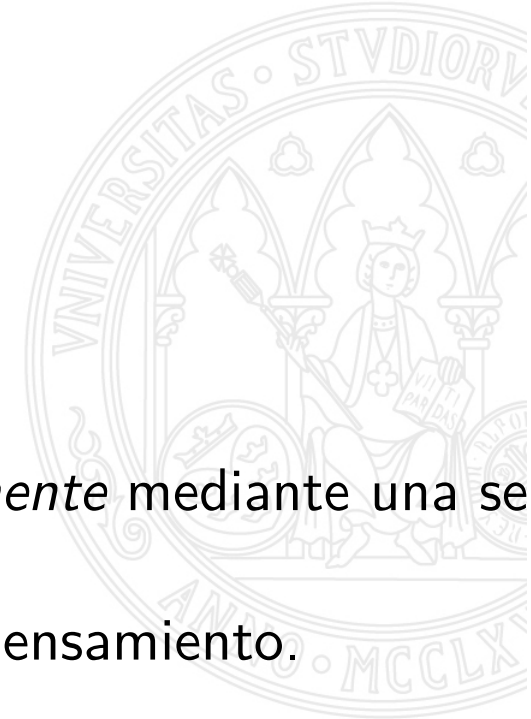
Introducción

- Existen diferentes sistemas gráficos:
  - *base*
  - *ggplot2*
  - *lattice*

Dos tipos de funciones gráficas

- Alto nivel: lanzan un nuevo *device gráfico*
- Bajo nivel: añaden elementos a un gráfico ya existente.

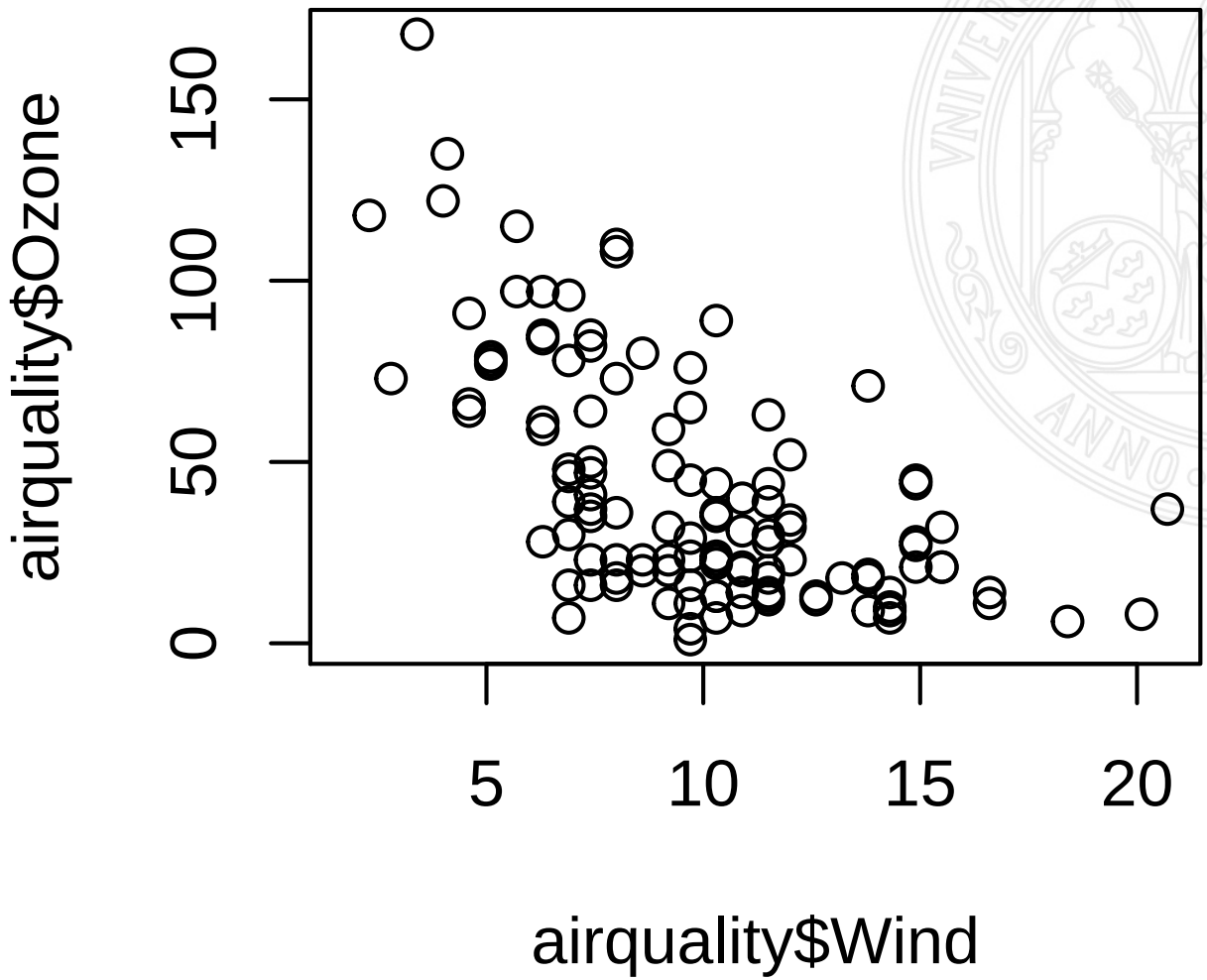
Sistema gráfico Base



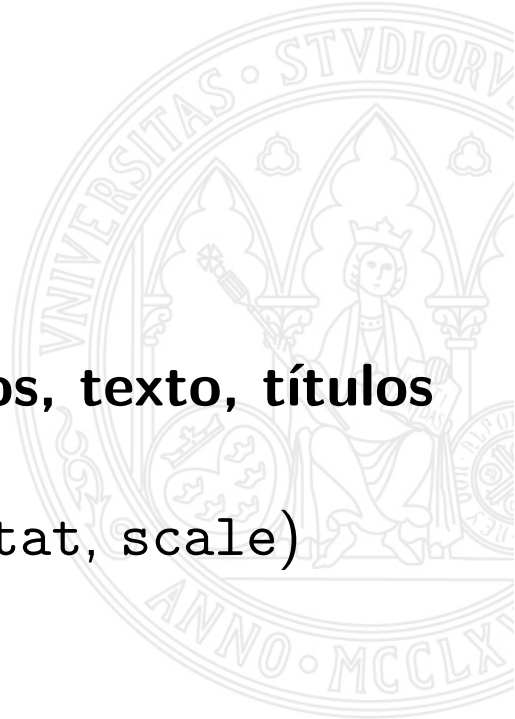
- Los gráficos se construyen *poco a poco*
- Cada aspecto del gráfico se trata *separadamente* mediante una serie de funciones
- Permite construir *reflejando* el proceso de pensamiento.

Base

```
plot( airquality$Wind, airquality$Ozone )
```



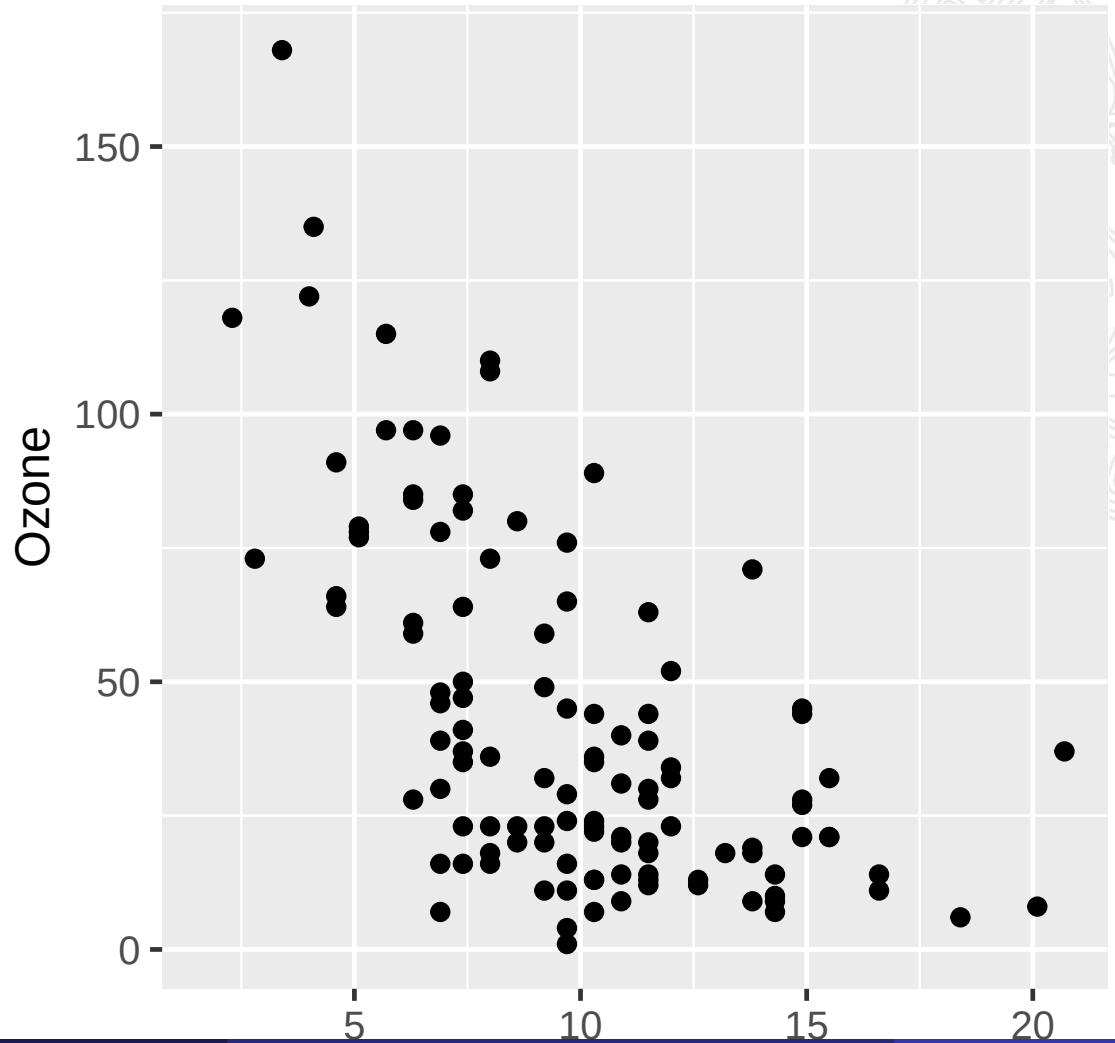
Sistema gráfico *ggplot2*



- Se ocupa **automáticamente** con **espaciados, texto, títulos**
- **Permite añadir** elementos al gráfico
- Trabaja con componentes gráficos (`geom`, `stat`, `scale`)
- Gráficos elegantes pero menos *intuitivos*
- Más sobre *ggplot2* aquí

ggplot2

```
library(ggplot2); qplot( Wind, Ozone, data = airquality )
```



Gráficos en R con el sistema Base

Creación de gráficos en R

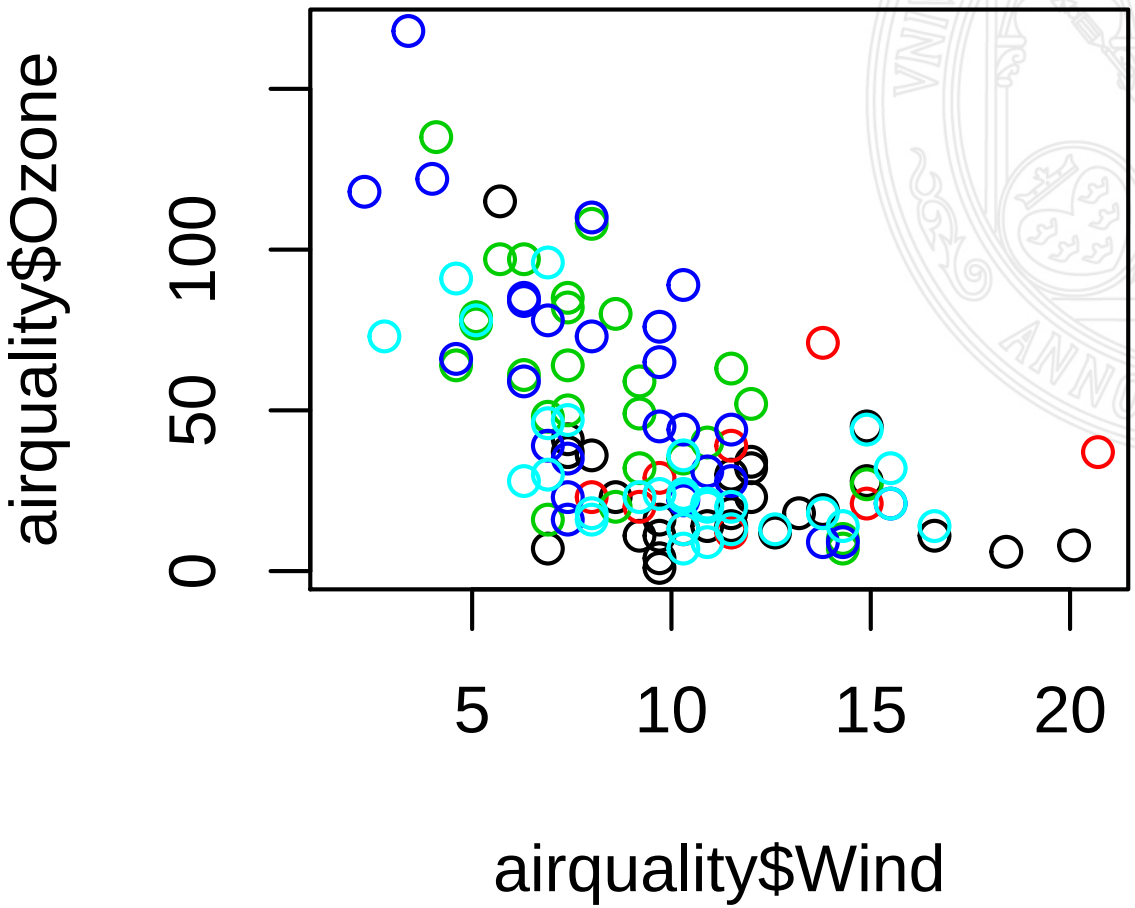
Hay dos tipos de funciones en este sistema

- Funciones de **alto nivel**: producen gráficos completos
  `plot( )`, `hist( )`, `boxplot()`, `curve()`, `barplot()`,...
- Funciones de **bajo nivel**: añaden elementos a un gráfico existente
  `points( )`, `text( )`, `legend()`

Funciones gráficas de alto nivel

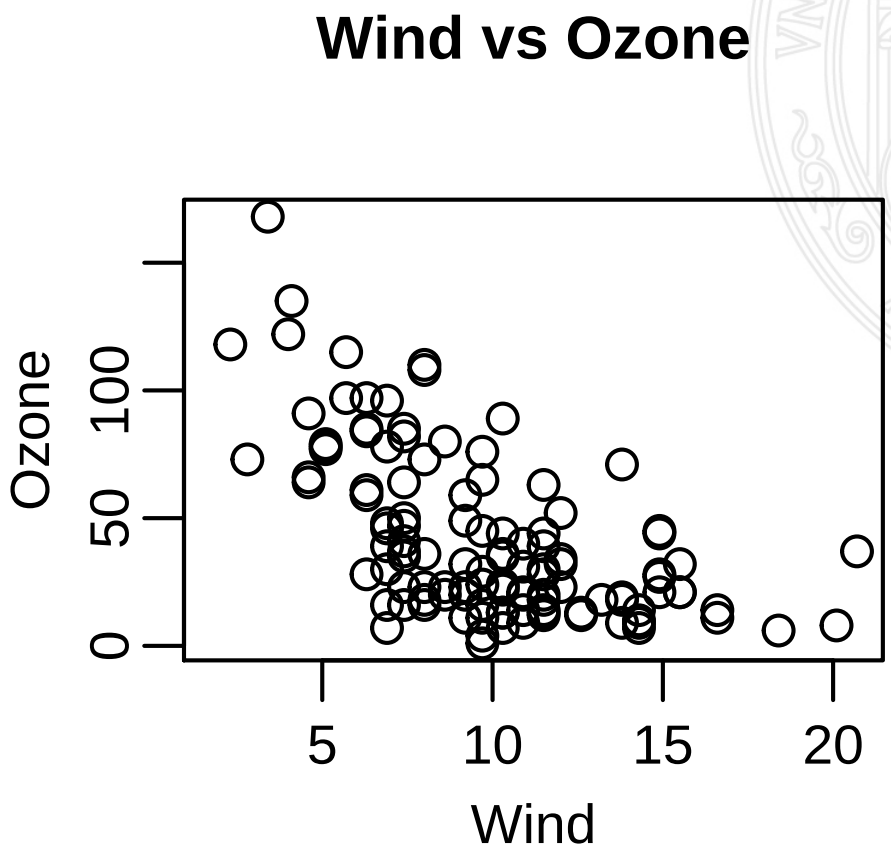
Gráficos de dispersión (por colores)

```
plot( airquality$Wind, airquality$Ozone,
      col = airquality$Month + 4 )
```

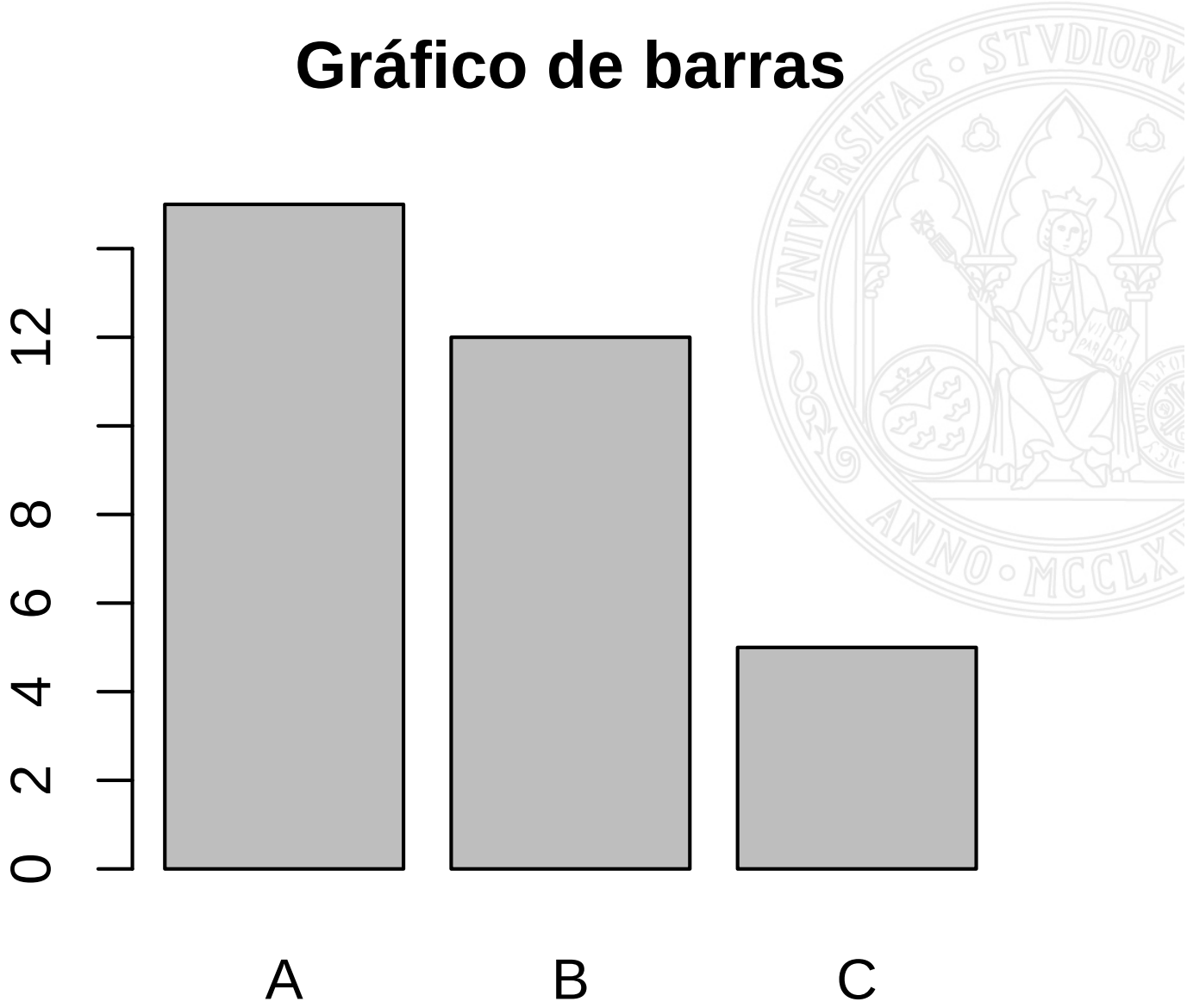


Gráficos de dispersión (etiquetas)

```
plot( airquality$Wind, airquality$Ozone, xlab = "Wind",
      ylab= "Ozone", main = "Wind vs Ozone" )
```



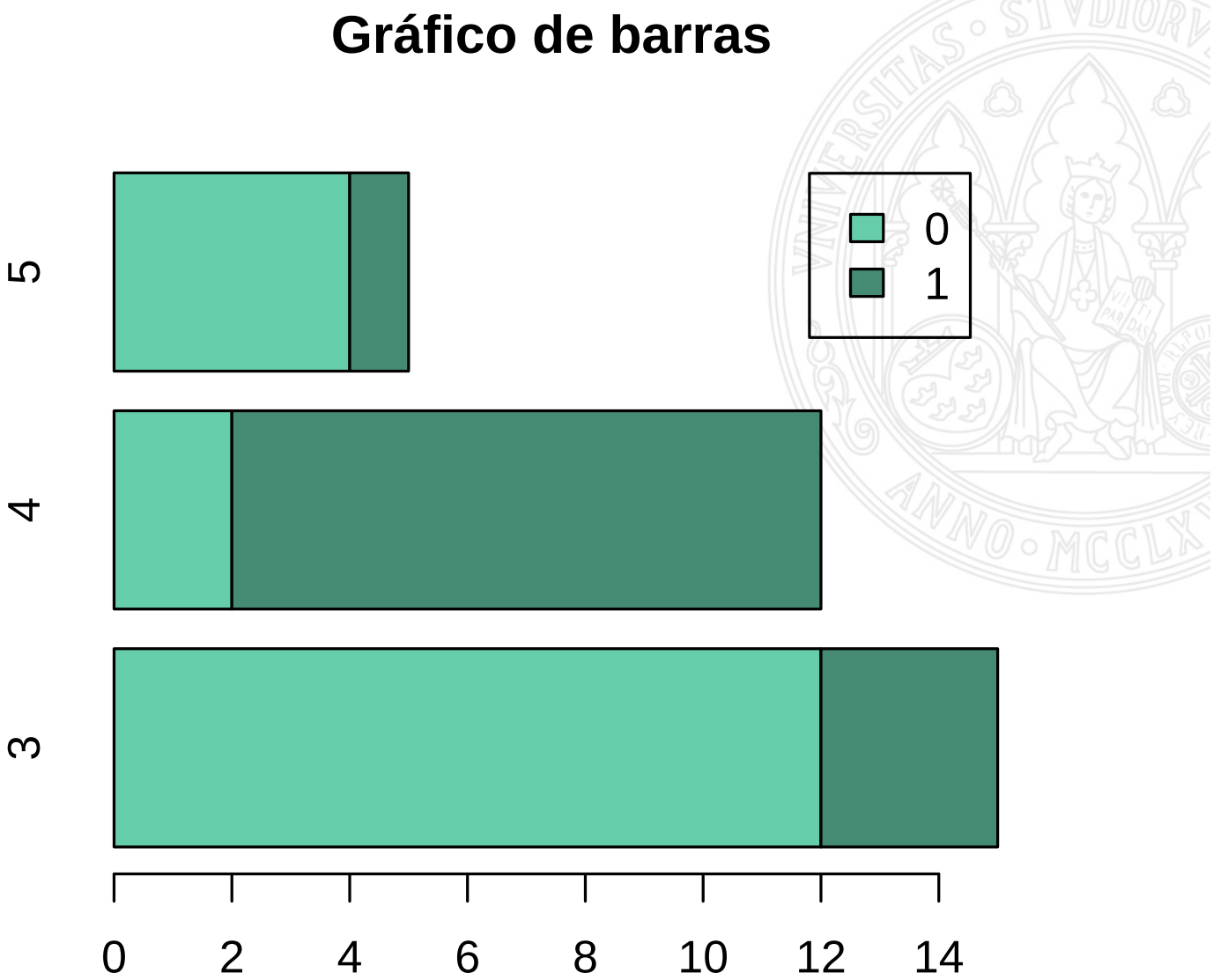
Gráficos de barras (simple)



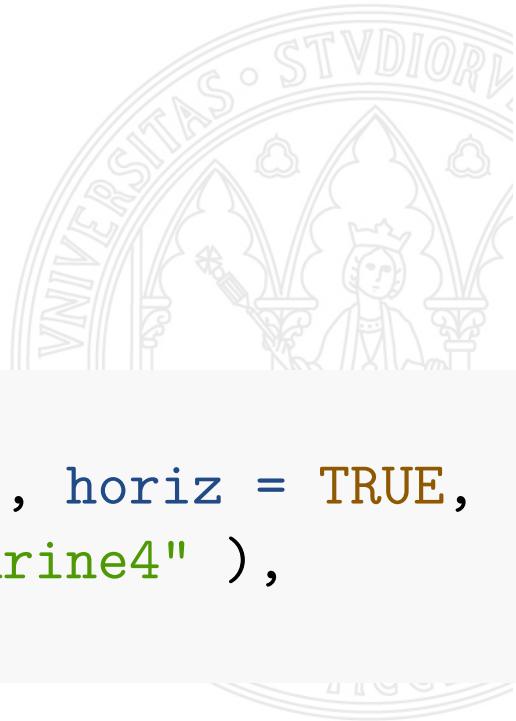
Gráficos de barras (simple)

```r
tt <- table( mtcars$gear ) # matriz
barplot( tt, main = "Gráfico de barras",
         names.arg = c( "A", "B", "C" ) )
```

Gráficos de barras (apilado)

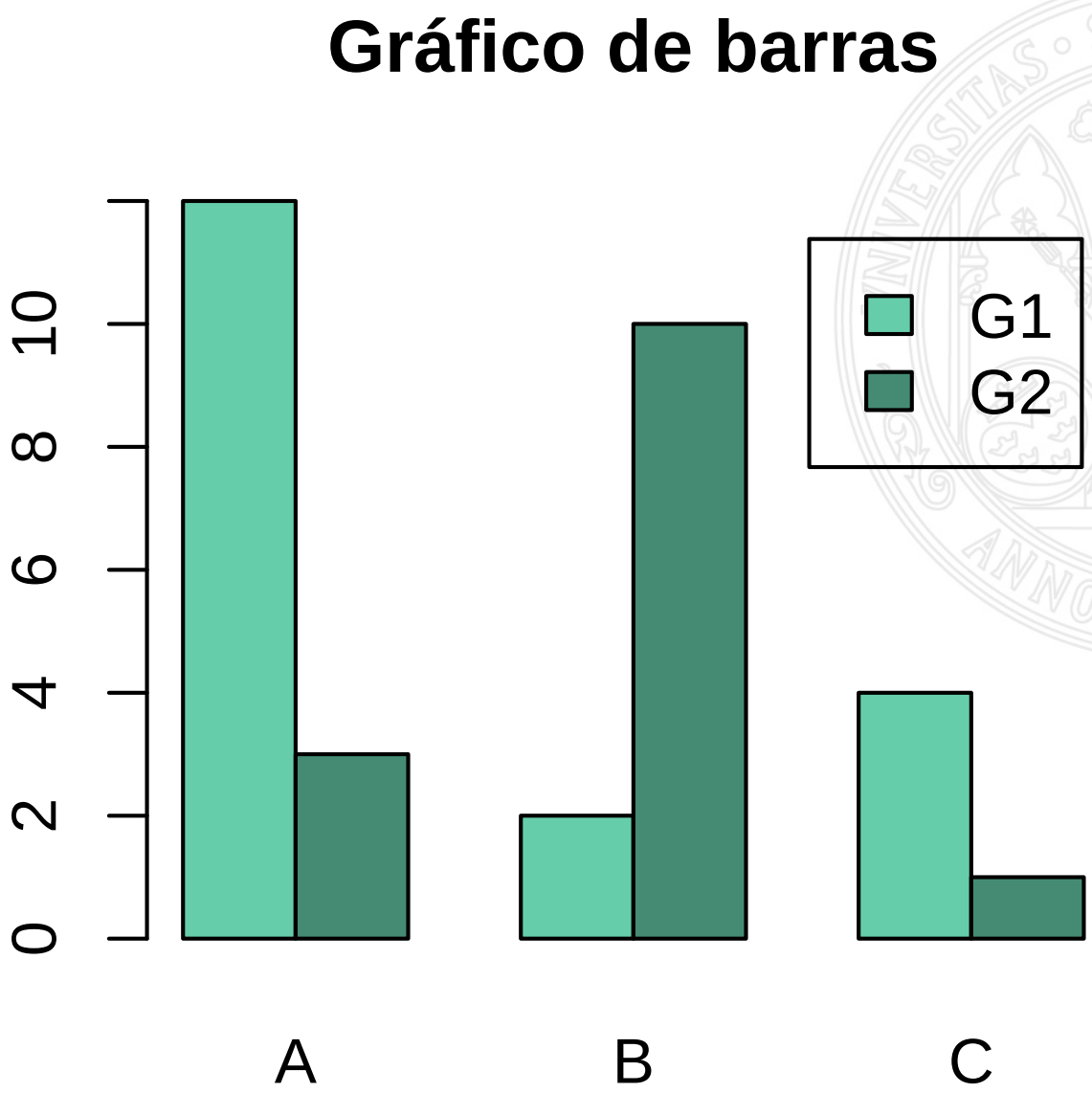


Gráficos de barras (apilado)



```
tt <- table( mtcars$vs, mtcars$gear )
barplot( tt, main = "Gráfico de barras", horiz = TRUE,
         col = c( "aquamarine3","aquamarine4" ),
         legend = rownames( tt ))
```

Gráficos de barras (agrupado)

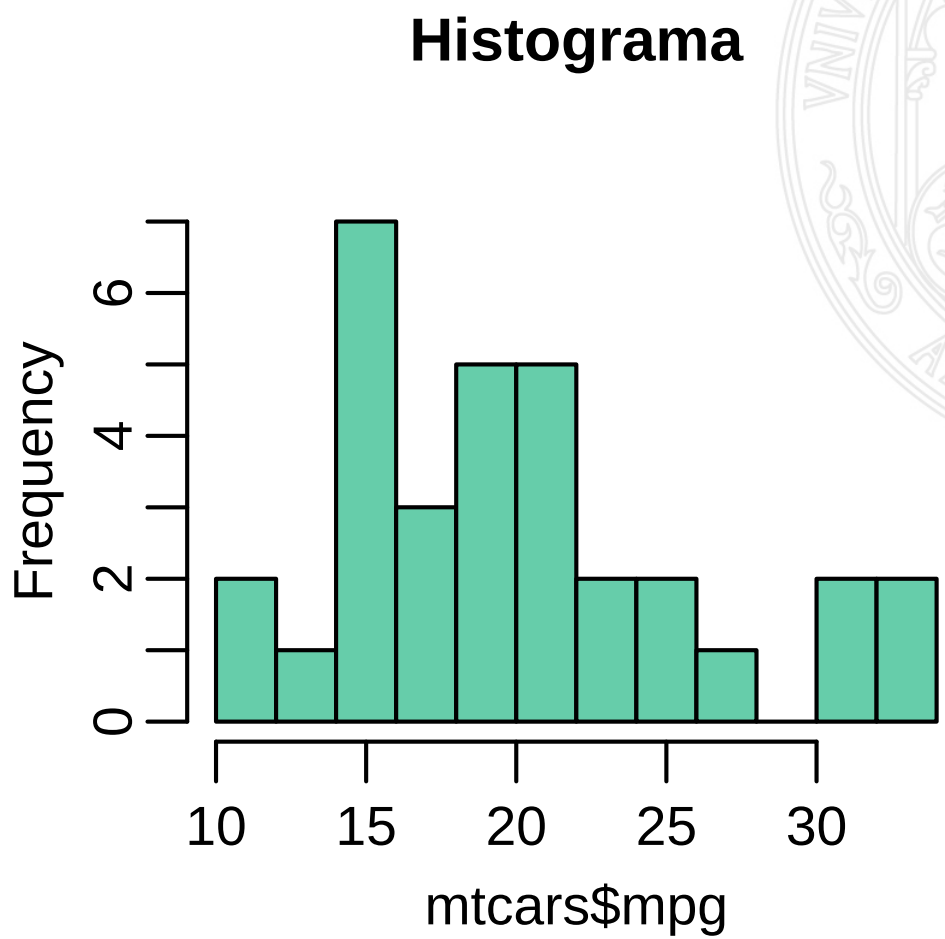


Gráficos de barras (agrupado)


```
tt <- table( mtcars$vs, mtcars$gear )
barplot( tt, main = "Gráfico de barras", beside = TRUE,
         col = c( "aquamarine3","aquamarine4" ),
         names.arg = c( "A", "B", "C" ),
         legend = c("G1", "G2") )
```

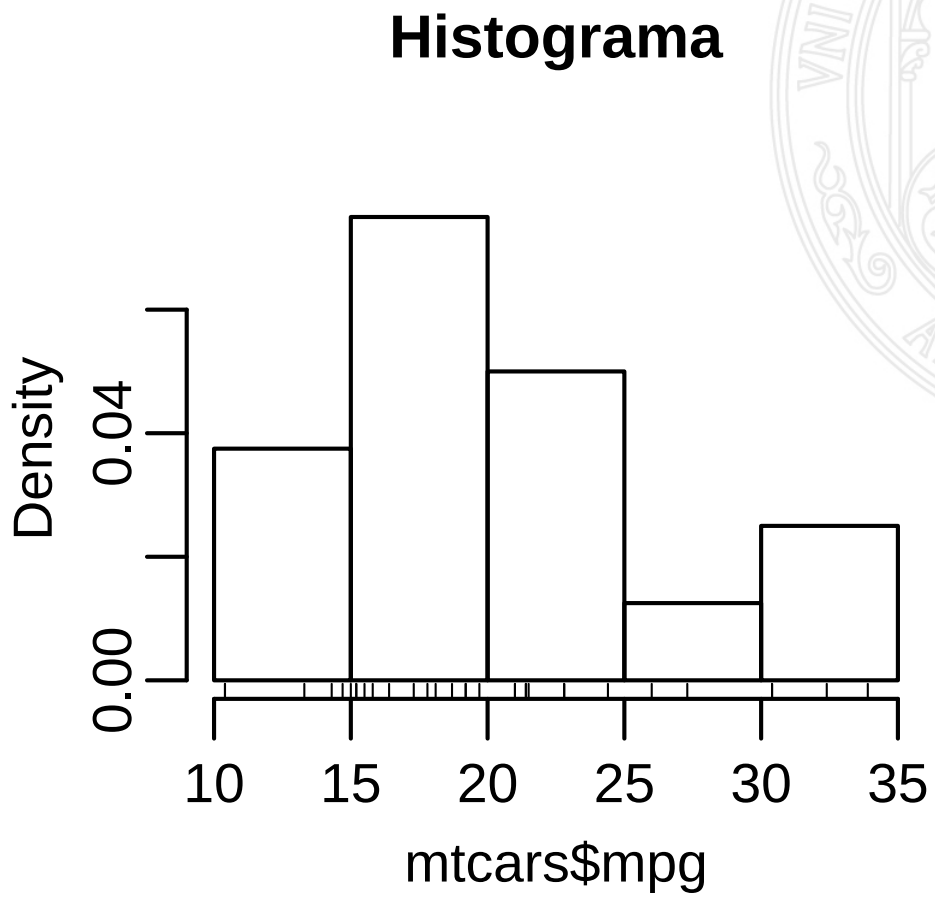
Histogramas

```
hist( mtcars$mpg, breaks = 9, col= "aquamarine3",
      main = "Histograma")
```



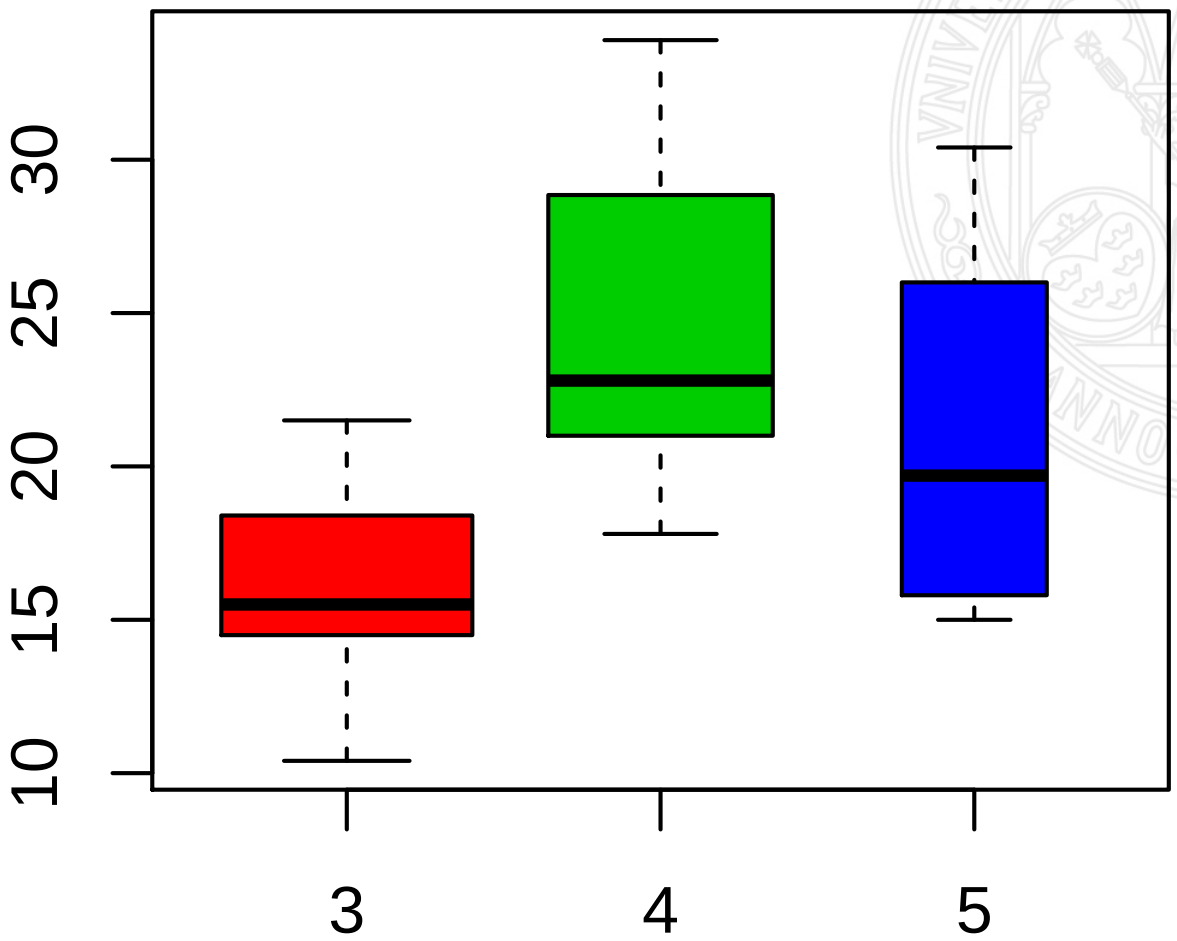
Histogramas

```
hist( mtcars$mpg, main = "Histograma", freq = FALSE )
rug( mtcars$mpg )
```



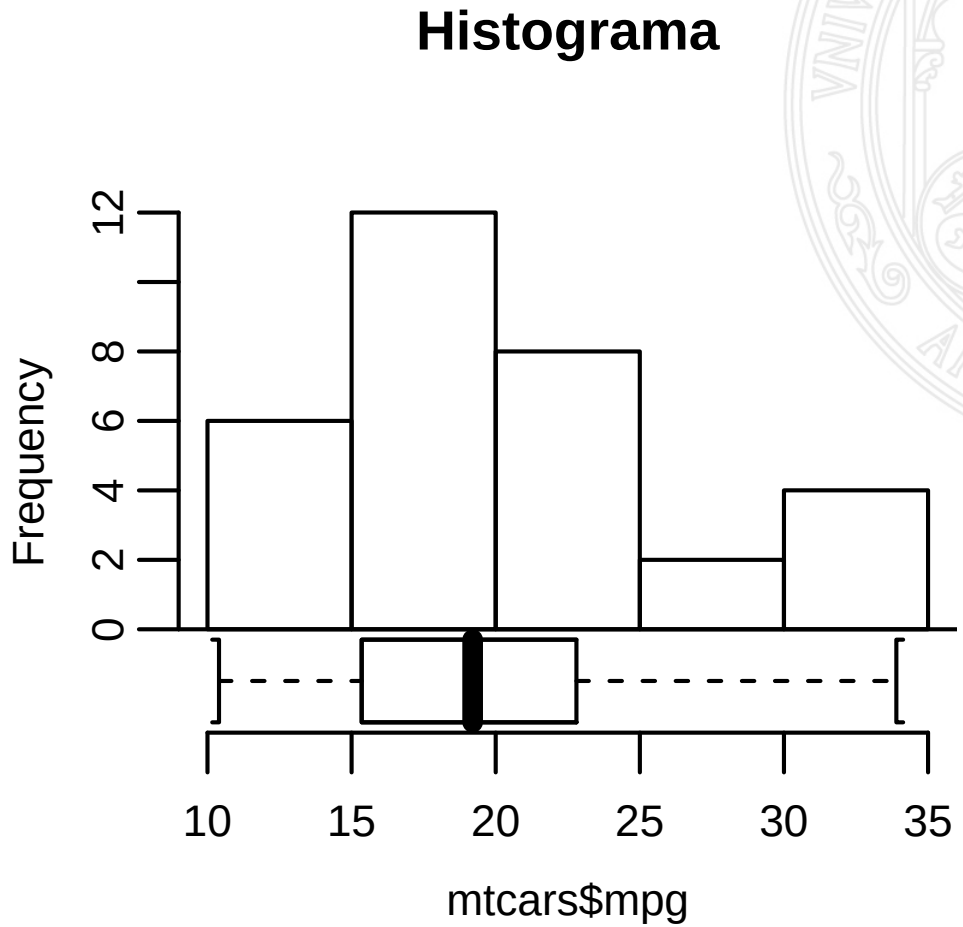
Boxplot

```
boxplot( mtcars$mpg ~ mtcars$gear, col = 2:4, varwidth = T )
```



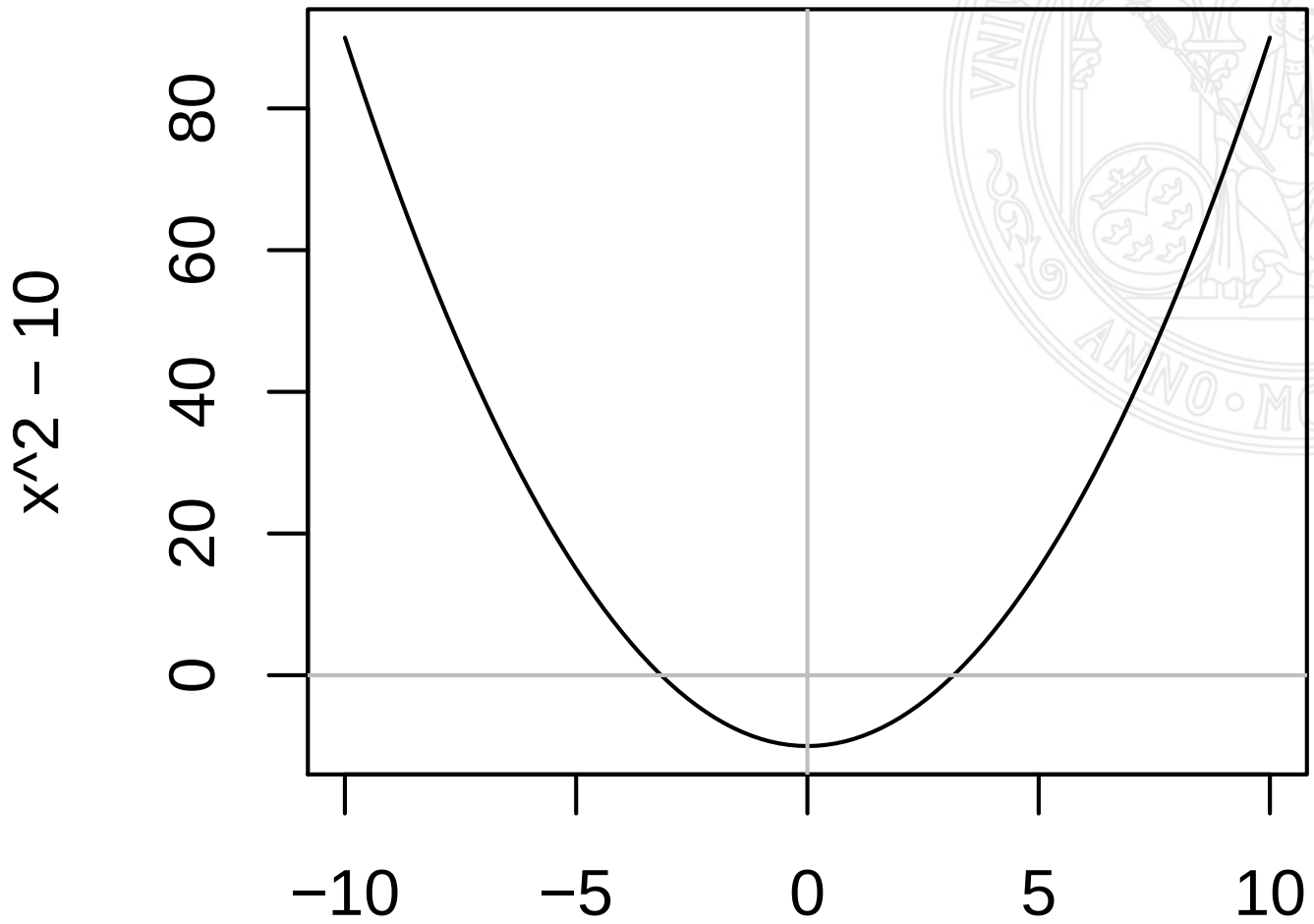
Histograma y boxplot

```
library( sfsmisc )
histBxp( mtcars$mpg, boxcol=0, medcol=1, main="Histograma" )
```

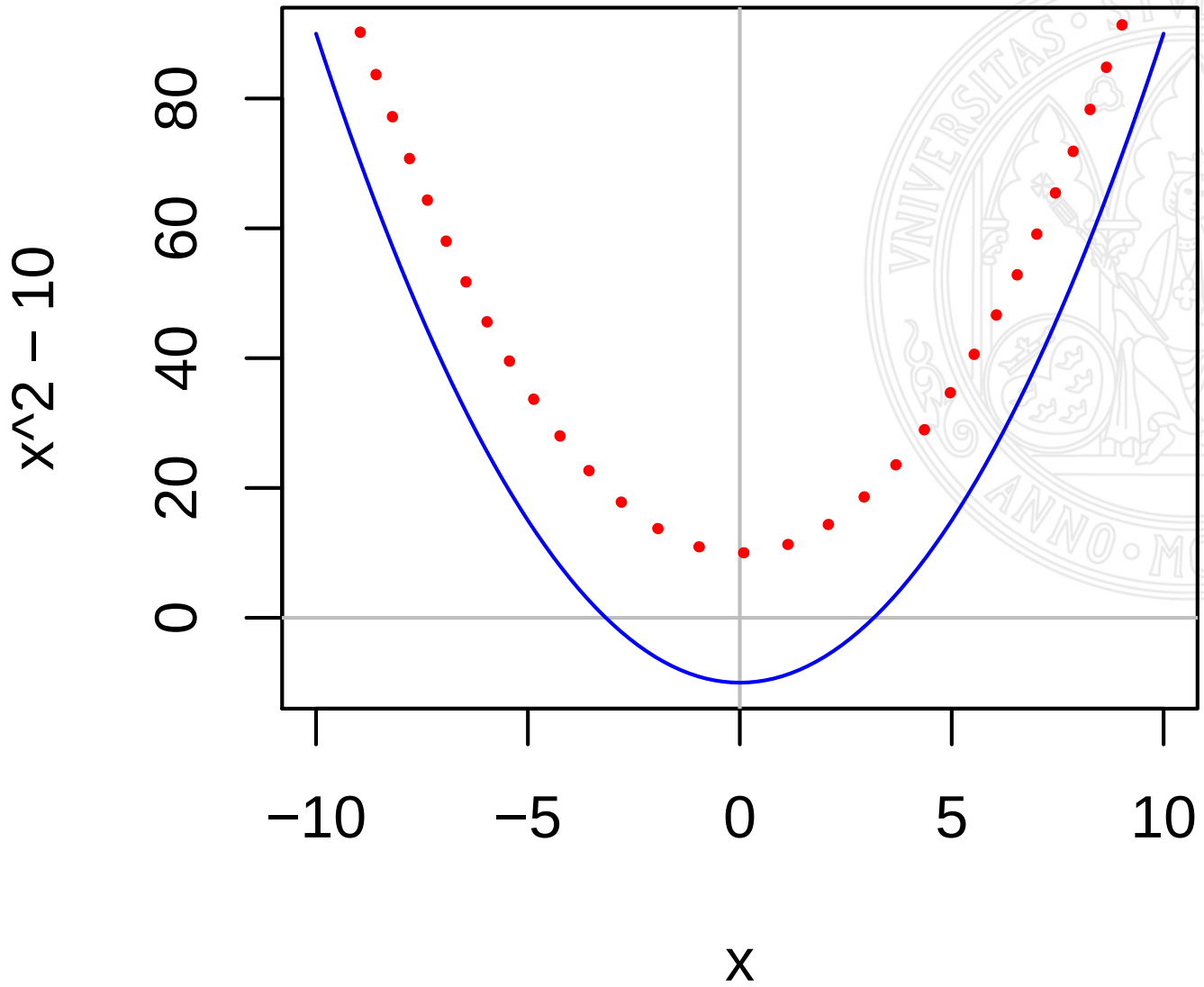


Curvas

```
curve( x^2 - 10, -10, 10 )
abline( v = 0, h = 0, col = "grey" ) # Ejes de coordenadas
```



Curvas

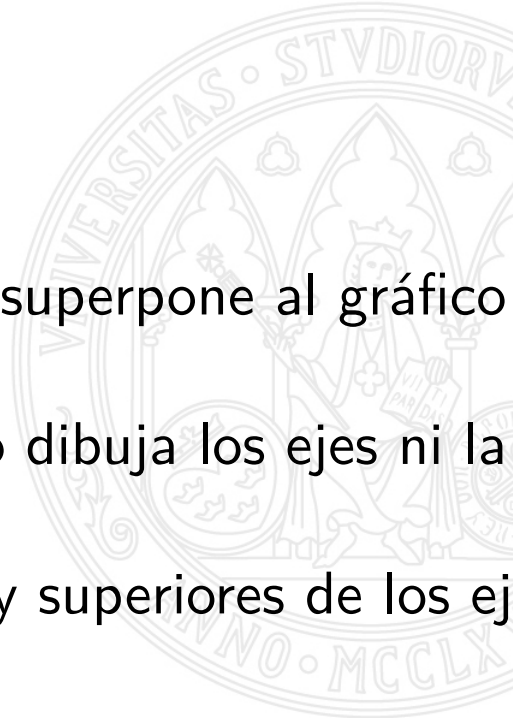


Curvas


```
curve( x^2 - 10, -10, 10, type = "n" )
abline( v = 0, h = 0, col = "grey" )
curve( x^2 - 10, -10, 10, col = "blue", add = TRUE )

# Y añadimos una curva más en el gráfico
curve( x^2 + 10, -10, 10, col = "red",
       lwd = 3, lty = 3, add = TRUE )
```

Argumentos generales (resumen)



- `add = FALSE` (por defecto): si es TRUE se superpone al gráfico existente
- `axes = TRUE` (por defecto): si es FALSE no dibuja los ejes ni la caja del gráfico
- `xlim`, `ylim`: especifica los límites inferiores y superiores de los ejes
- `xlab`, `ylab` : añade etiquetas a los ejes
- `main` : añade el título principal
- `sub` : añade un subtitulo (letra más pequeña).

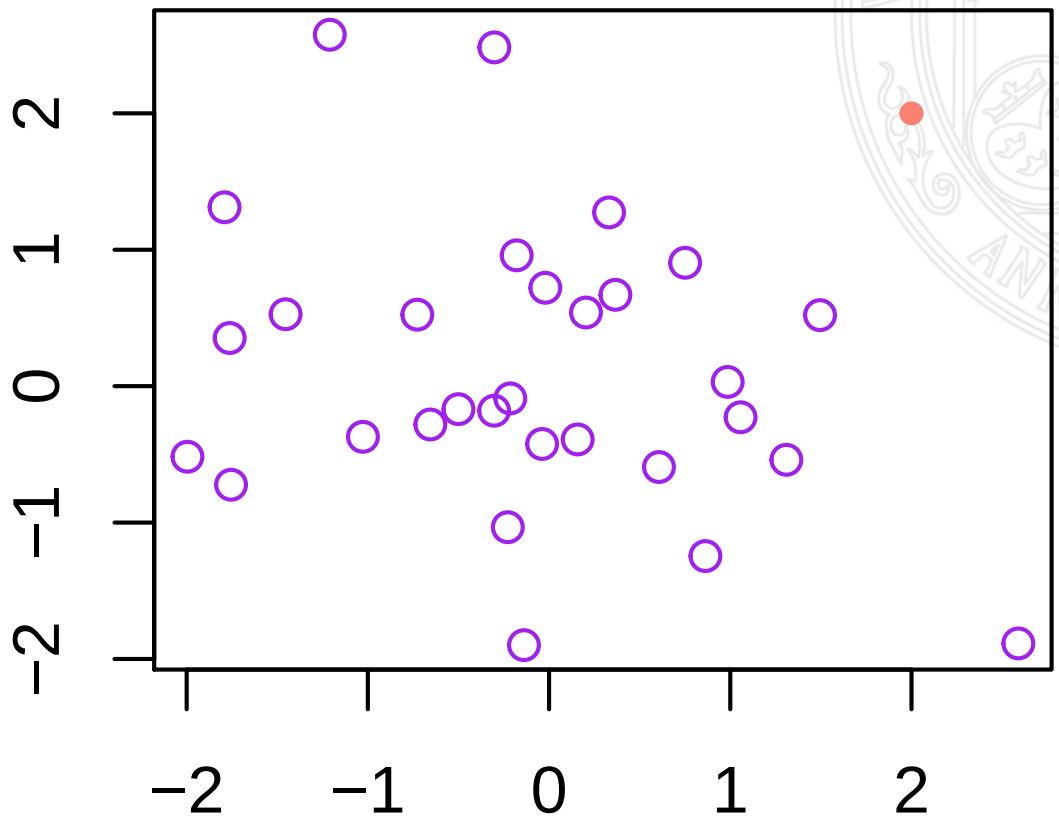
Funciones de bajo nivel

Funciones de bajo nivel

- Una vez tenemos ya un *device gráfico* lanzado
  - podemos añadirle una serie de "objetos"
  - mediante un conjunto de expresiones
    - puntos, rectas
    - texto, etiquetas, etc

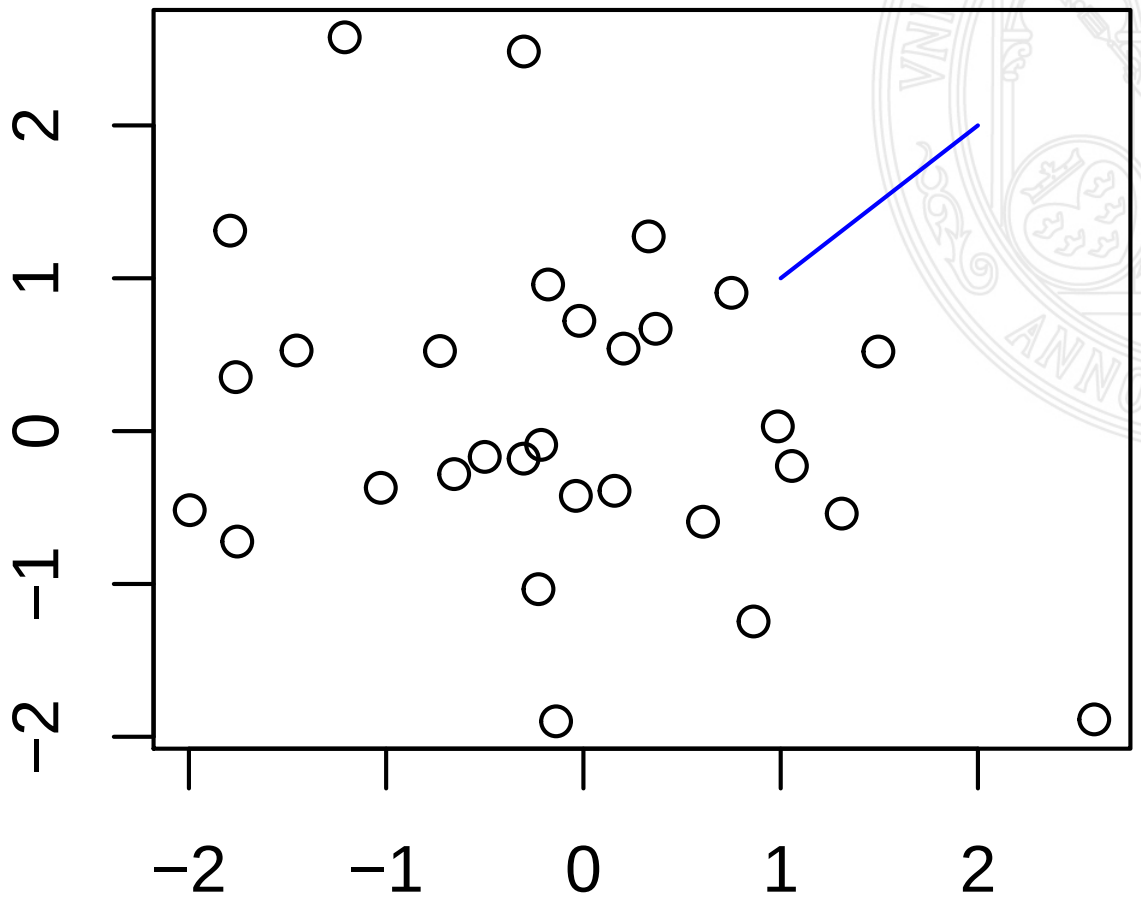
Puntos

```
plot( x, y, type = "n", ann= FALSE )
points( x, y, col= "purple" )
points( 2, 2 , col = "salmon", pch = 20 )
```

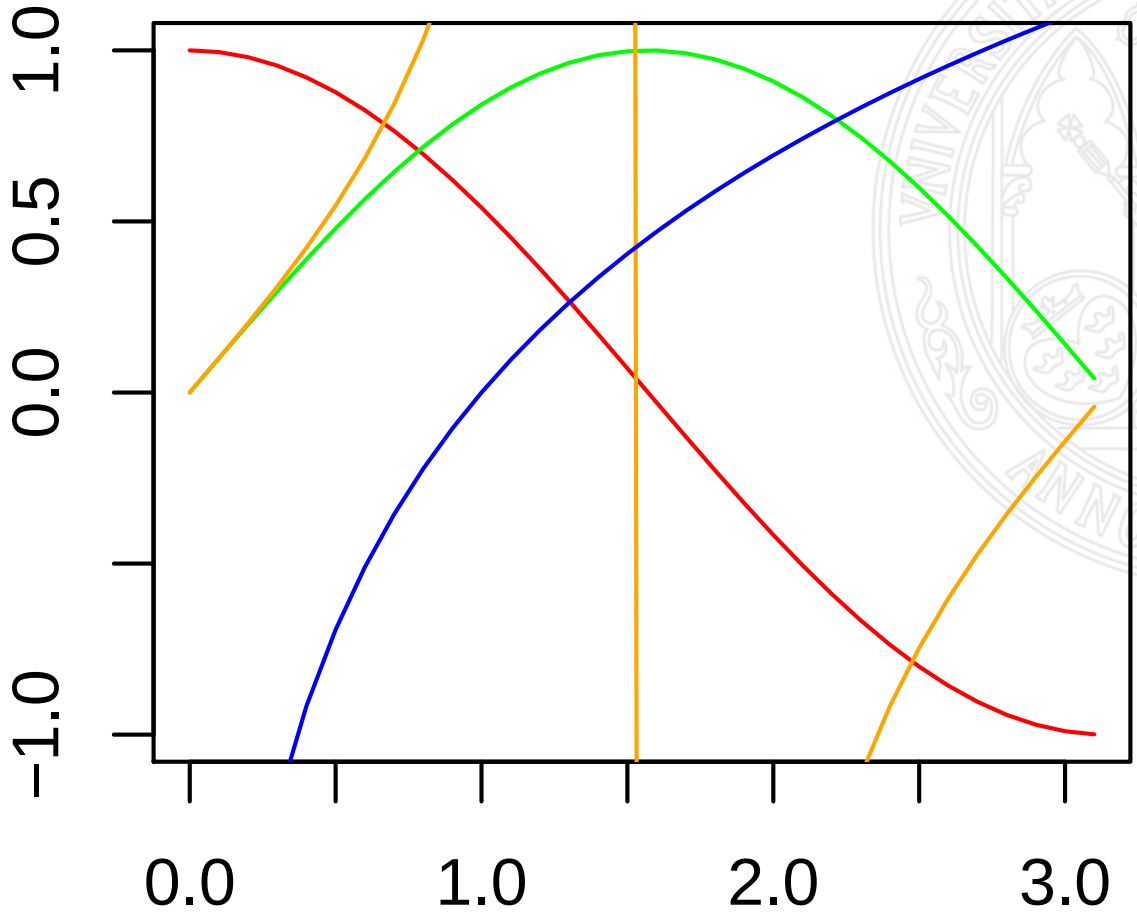


Lineas

```
plot(x, y, xlab= "abscisas", ylab= "ordenadas", ann= FALSE)
lines( c( 1, 2 ) , col = "blue" )
```



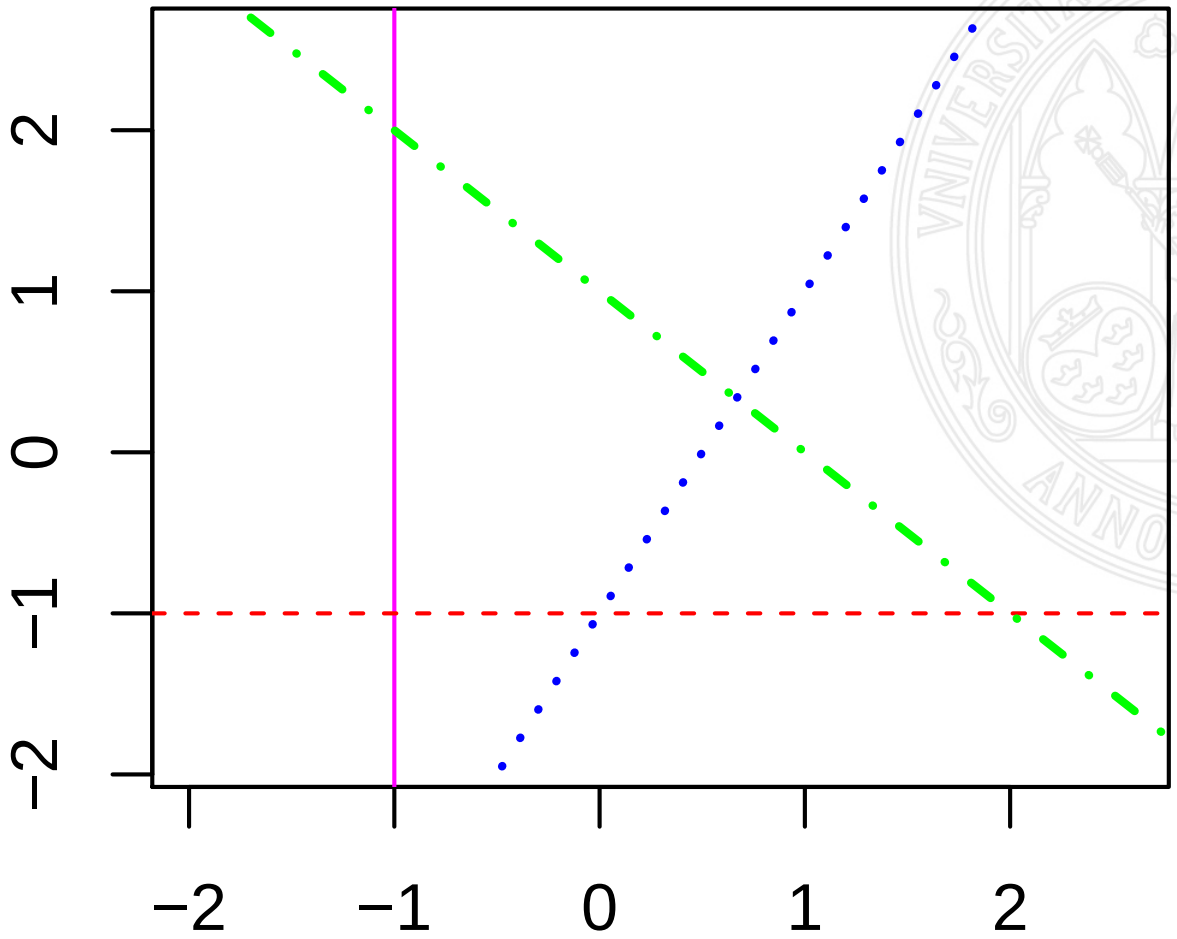
Más líneas



Más líneas


```
s <- seq( 0, pi, 0.1 )
plot( s, cos( s ), type = "l", col = "red" )
lines( s, sin( s ), col = "green" )
lines( s, tan( s ), col = "orange" )
lines( s, log( s ), col = "blue" )
```

Rectas

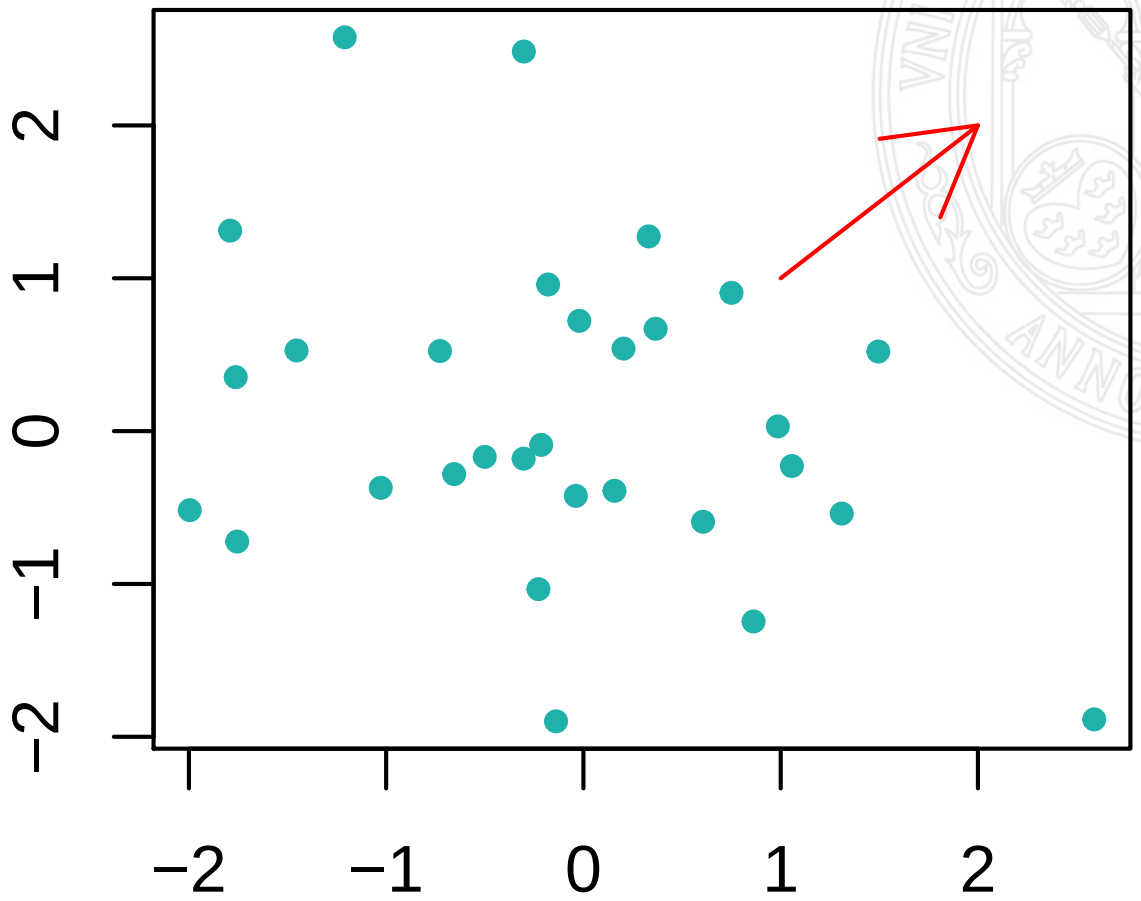


Rectas


```
plot( x, y, main = "Rectas", type = "n" )
abline( v = -1, lty = 1, col = "magenta" )
abline( h = -1, lty = 2, col = "red" )
abline( -1, 2, lty = 3, col = "blue", lwd = 2 )
abline( 1, -1, lty = 4, col = "green", lwd = 2 )
```

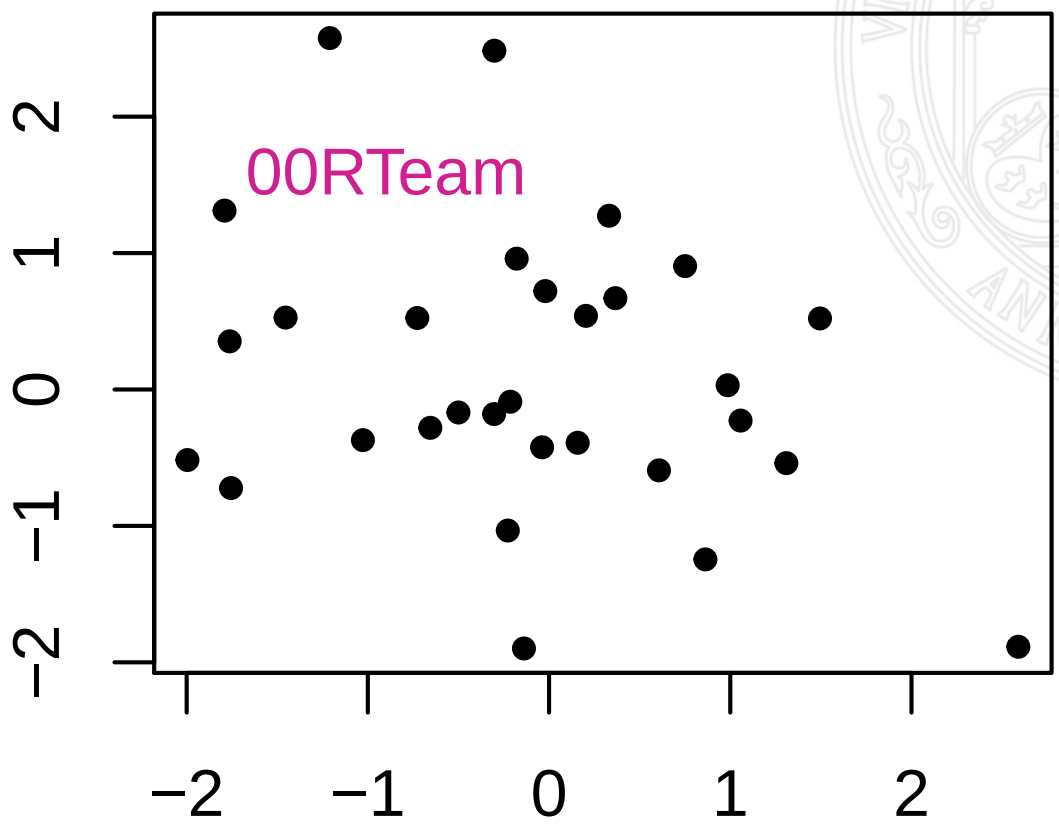
Flechas

```
plot( x, y, col = "lightseagreen", pch = 20, ann= FALSE )
arrows( 1, 1, 2, 2, col = "red" )
```



Texto

```
plot( x, y, xlab = "x", ylab = "y", pch = 20, ann = FALSE )
text( -0.9, 1.6, "00RTeam", col = "violetred" )
```



Leyendas

- `legend( posición, título, leyenda, ... )`.

Otras opciones

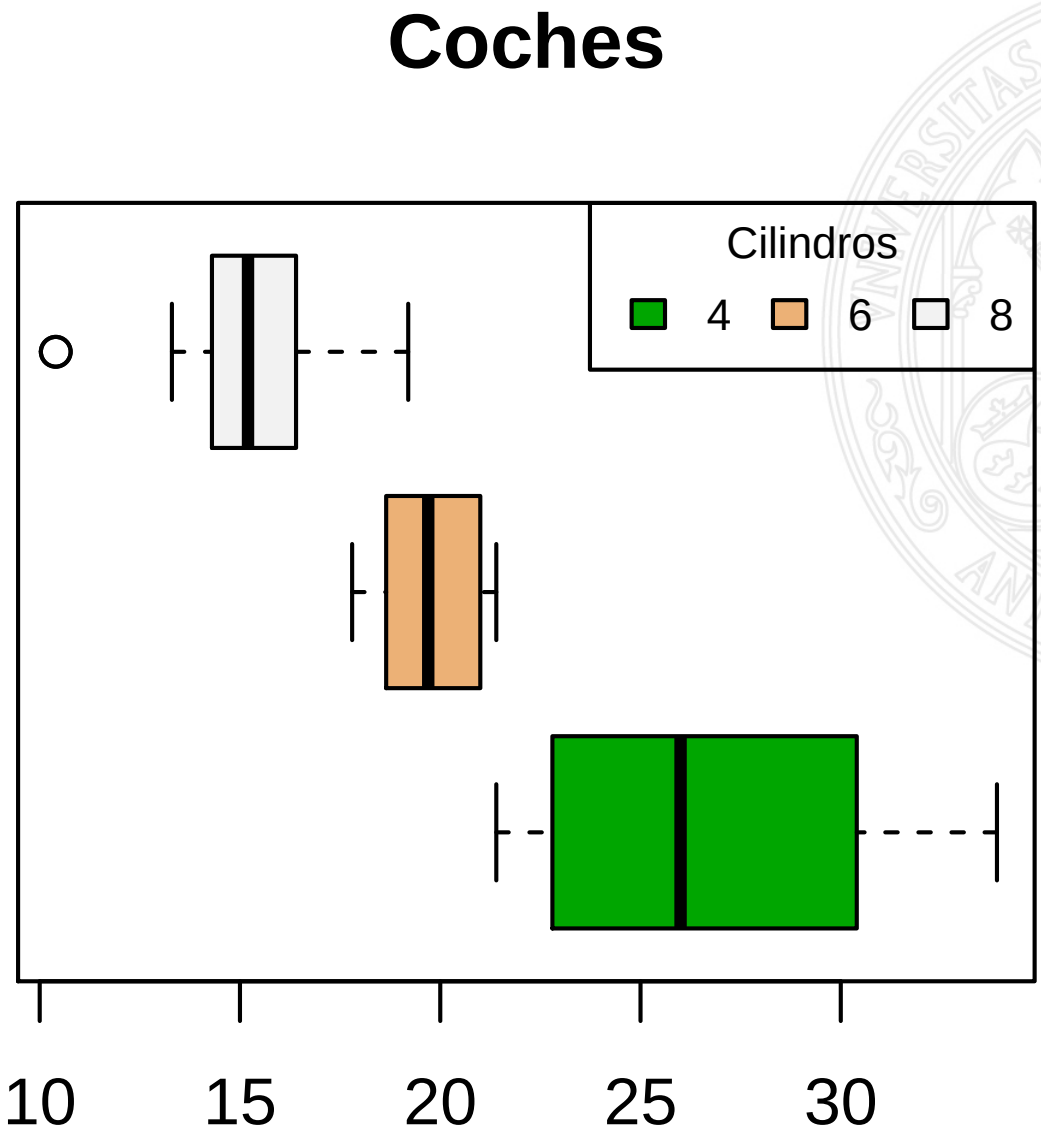
- `bty`: fondo caja
- `bg`: para color de fondo
- `cex`: para el tamaño
- `col`: para el color del texto.
- `horiz = TRUE`: se dibuja la leyenda horizontal

Leyendas

```r
boxplot( mtcars$mpg ~ mtcars$cyl,
         main = "Coches",
         yaxt = "n",
         xlab = "Millas",
         horizontal = TRUE,
         col = terrain.colors( 3 )
         )

legend( "topright",                          # disposición
        cex = 0.7,                           # tamaño letra
        c( "4", "6", "8" ),                  # etiquetas
        fill = terrain.colors( 3 ),          # colores
        horiz = TRUE,                        # posición
        title = "Cilindros"  )
```

Leyendas



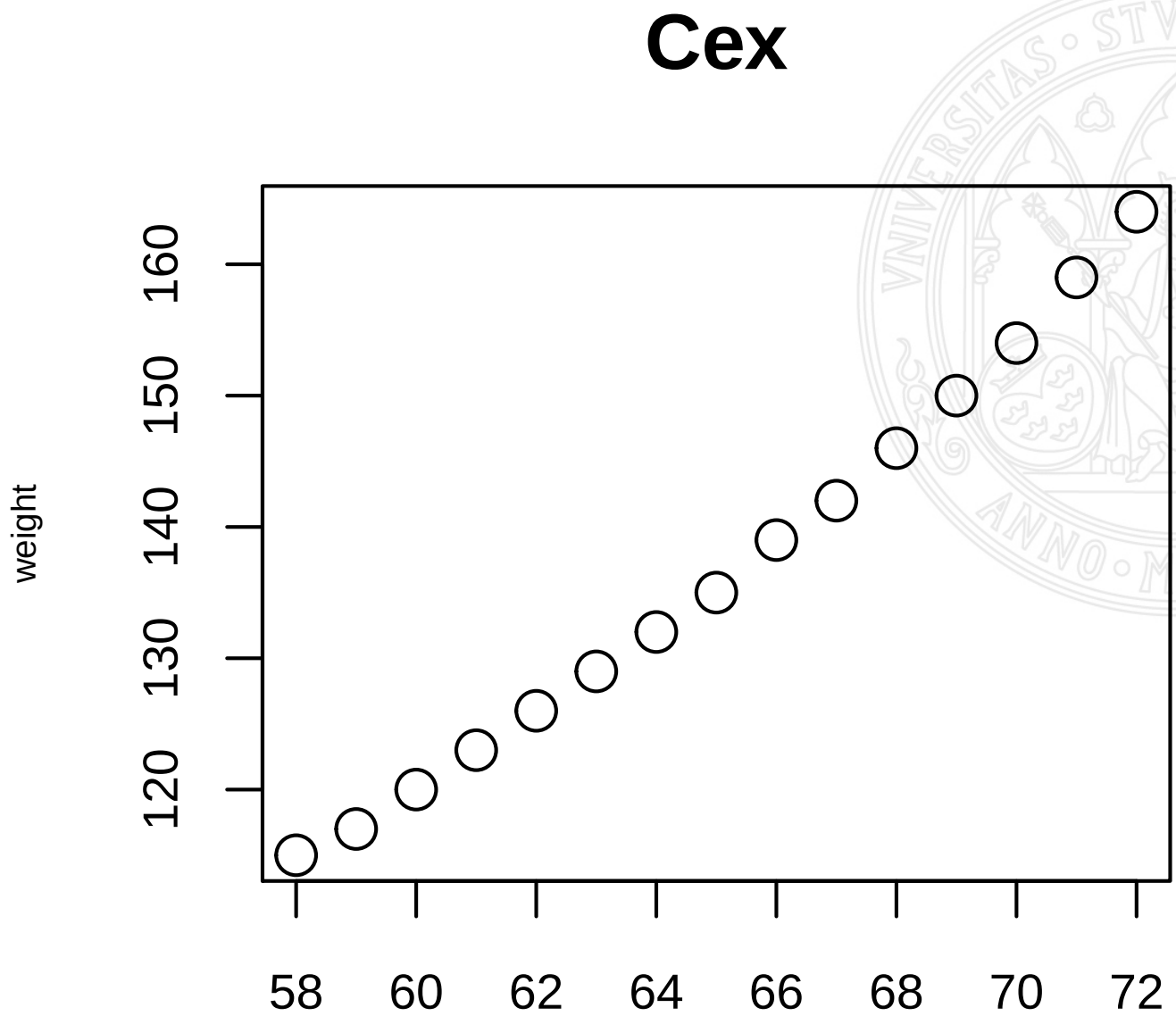
Parámetros gráficos

Tamaño del texto

Podemos utilizar las siguientes opciones para controlar el tamaño del texto y de los símbolos en los gráficos.

- `cex`: valor que escala el tamaño del texto y de los símbolos
  - Por defecto es 1
  - Un 1.5 significa un 50 % más grande
  - Un 0.5 es un 50 % más pequeño
- `cex.axis`: incremento de los ejes con respecto al `cex`
- `cex.lab`: aumenta las etiquetas de x e y en relación a `cex`.
- `cex.main`: magnificación de los títulos relativos al `cex`.
- `cex.sub`: ampliación del subtitulo con respecto al `cex`.

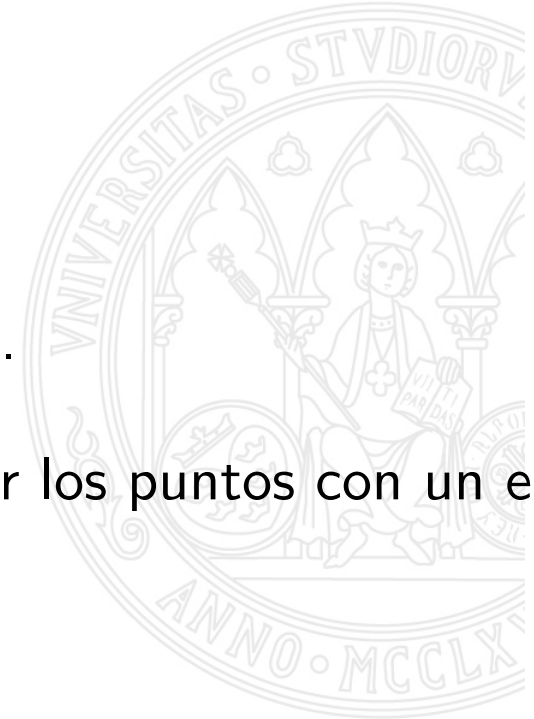
Tamaño del texto



tamaños


```
plot( women
      , cex = 1.5
      , main= "Cex"
      , cex.main = 1.3
      , cex.lab = 0.6
      , cex.axis = 0.8 )
```

Símbolos

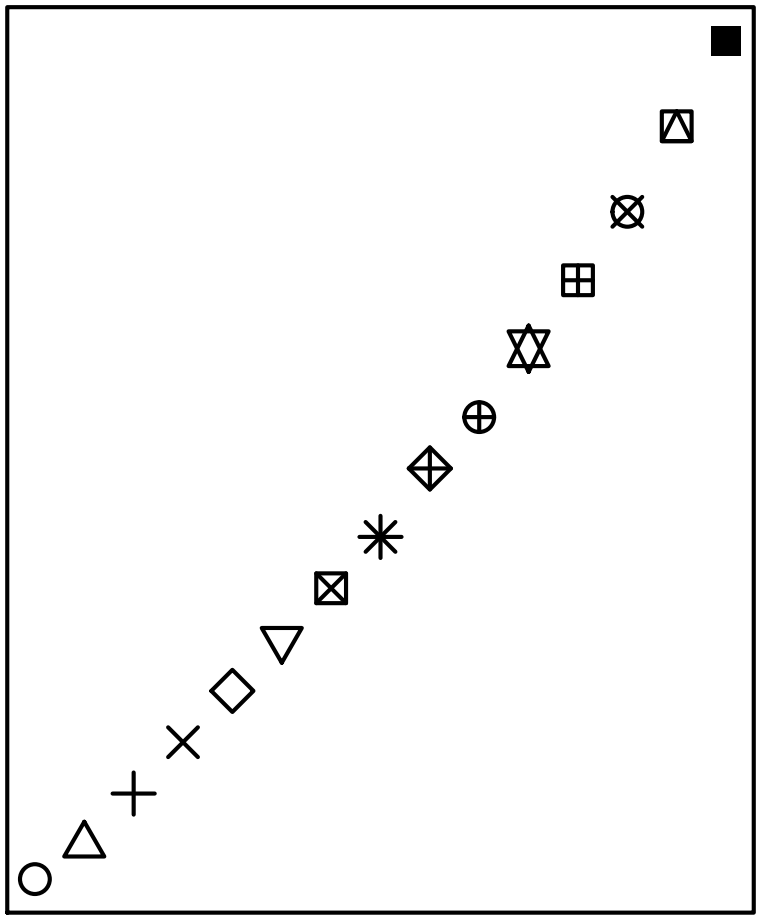


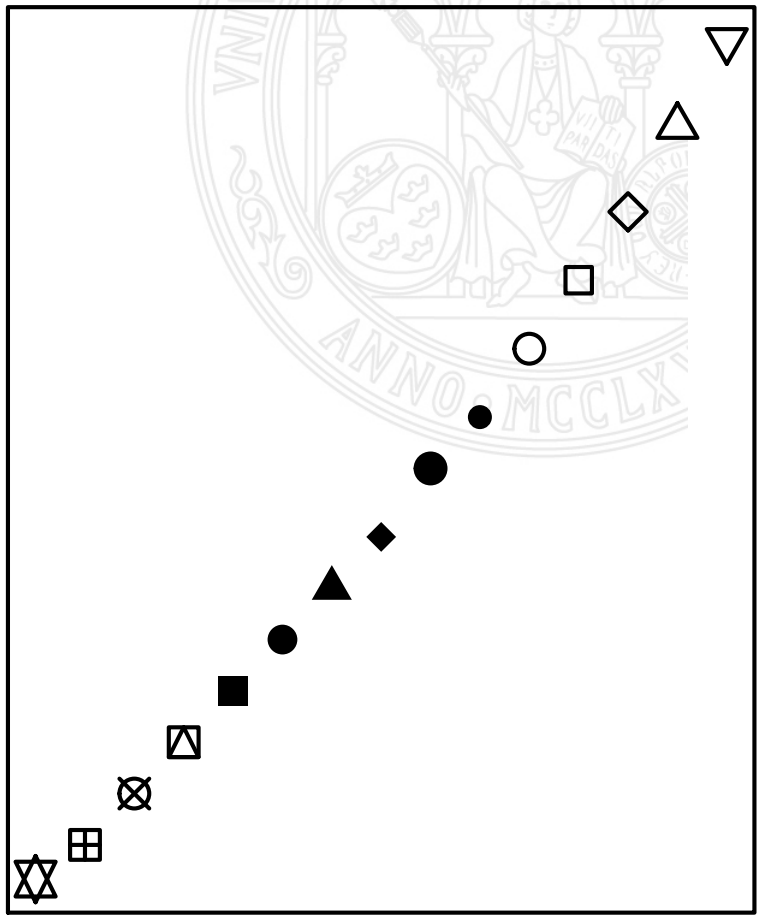
Podemos personalizar las opciones de los puntos.

- `pch`: controlar el tipo de símbolo para trazar los puntos con un entero entre 1 y 25.
- `col`: determinar el color del borde
- `bg`: modificar el color de relleno

Símbolos

```
plot( women, pch =  1:20 ) # Un símbolo para cada punto
plot( women, pch = 11:30 )
```





Símbolos

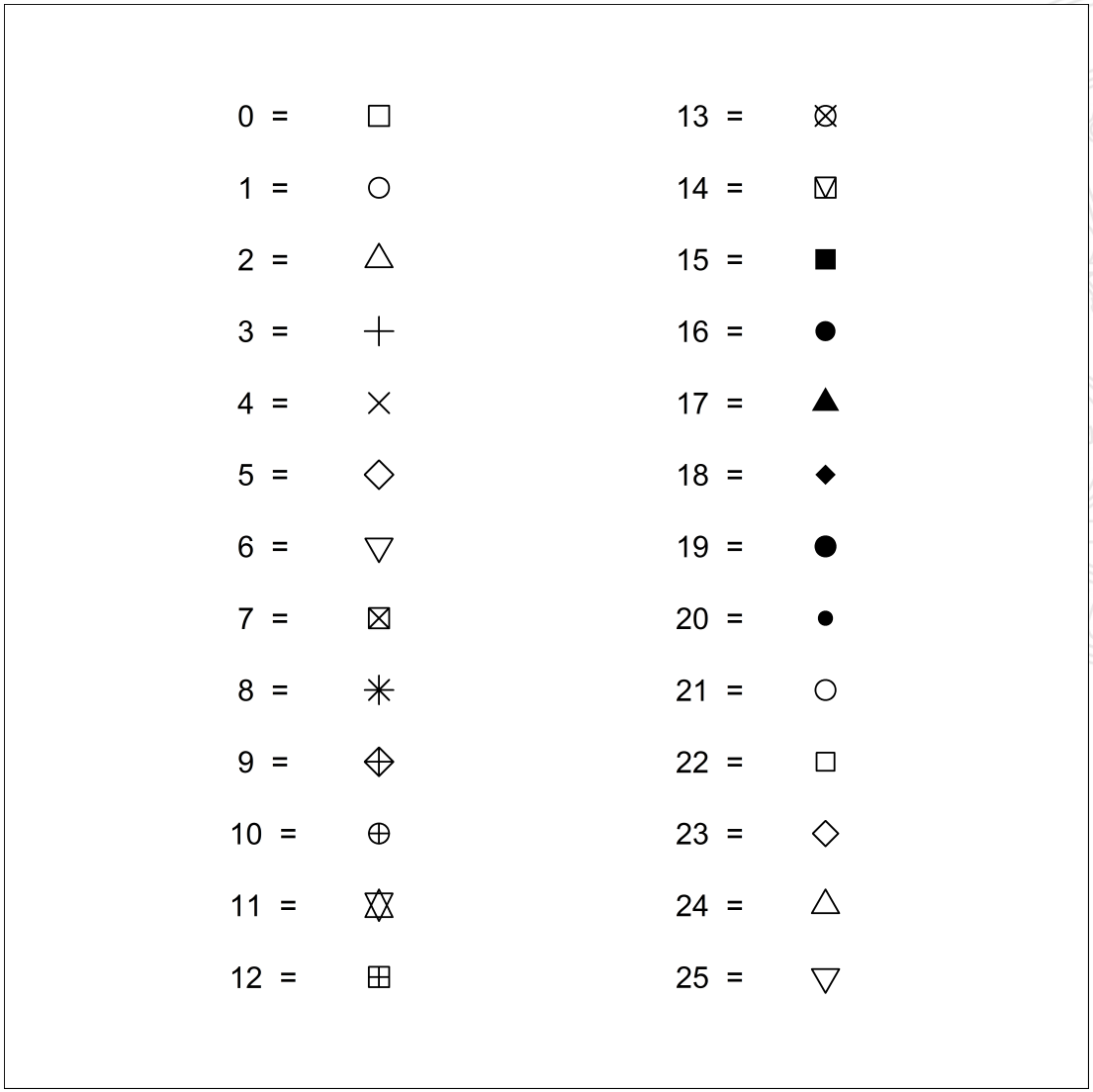
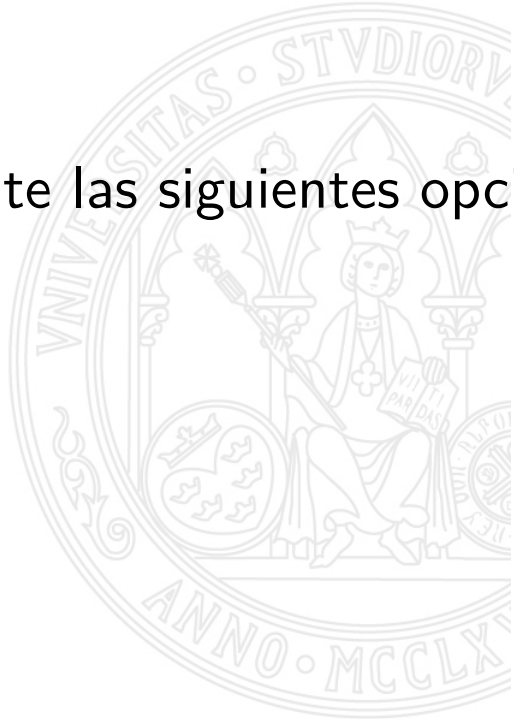


Figura 1:

Líneas



Podemos cambiar el trazado de las lineas mediante las siguientes opciones.

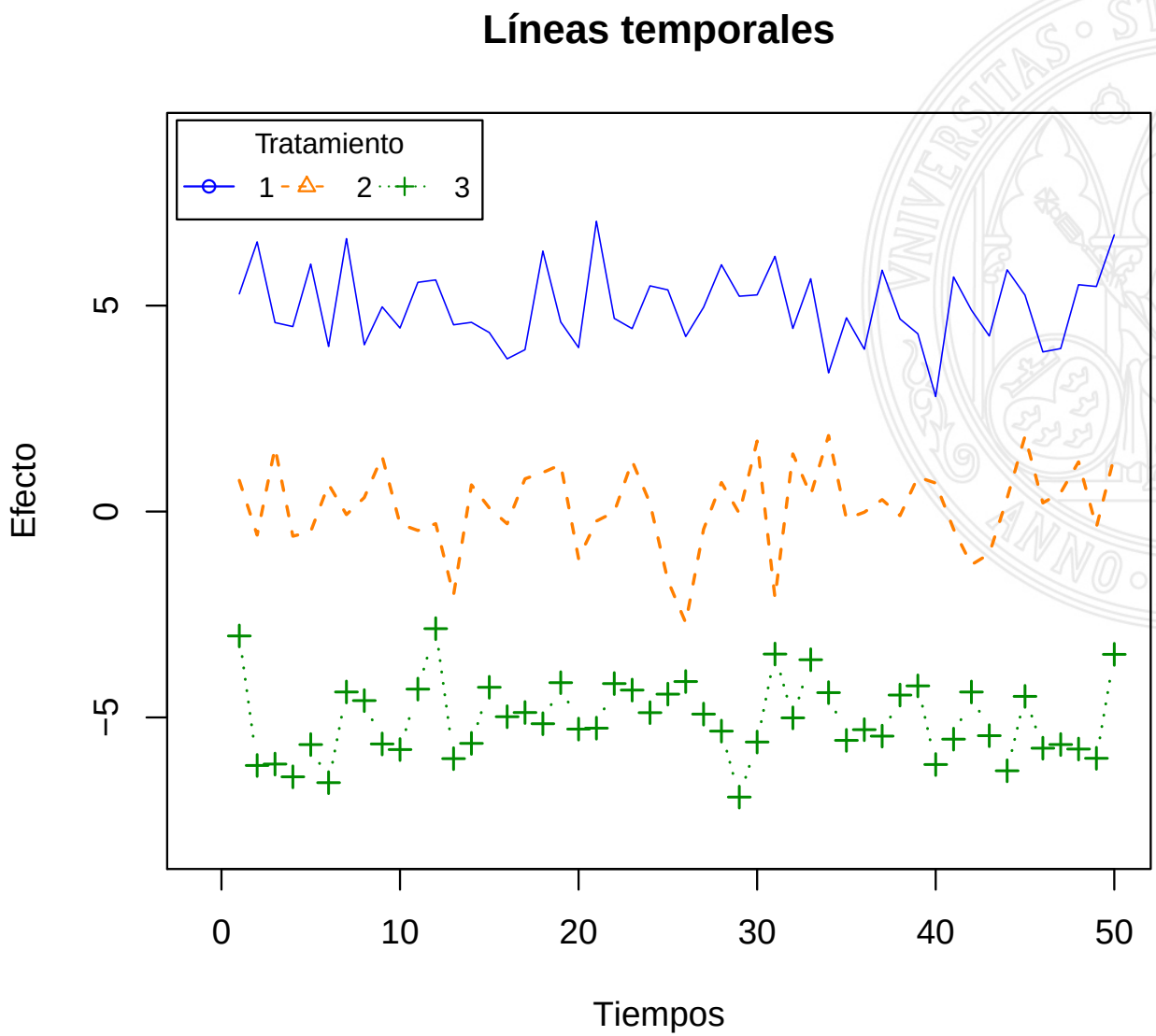
- `lty`: controla el tipo de las lineas
  - 1: solida
  - 2: quebrada
  - 3: punteada
  - 4: punto-linea
  - 5: linea larga-corta
  - 6: dos lineas cortas
- `lwd`: determinar el ancho de linea en relación con el valor por defecto (defecto=1)

un ejemplo de datos

```
x  <- c( 1:50 )
y0 <- rnorm( 50, mean = 5 )
y1 <- rnorm( 50, mean = 0 )
y2 <- rnorm( 50, mean = -5 )
head(data.frame(x,y0,y1,y2))
```

```
##    x       y0         y1        y2
##  1 1 5.285587  0.7564639 -3.021106
##  2 2 6.547195 -0.5716176 -6.166184
##  3 3 4.588391  1.5511209 -6.132424
##  4 4 4.491834 -0.5992649 -6.442365
##  5 5 6.007765 -0.4857675 -5.658748
##  6 6 4.008265  0.6611807 -6.583768
```

Líneas


Líneas temporales
Tratamiento
1
2
3
Efecto
5
0
−5
0
10
20
30
40
50
Tiempos

Líneas

```
plot( c(-1,50), c(-8,9), type="n", xlab="Tiempos",
     ylab="Efecto", main = "Líneas temporales" )

lines( x, y0, lwd = 0.7, lty = 1, col = "blue", pch = 1 )
lines( x, y1, lwd = 1.5,lty = 2, col = "darkorange1" )
lines( x, y2, type = "b", lwd = 1.3, lty = 3,
       col = "green4", pch = 20 )

legend( -2.5, 9.5, 1:3, cex=0.8, horiz = TRUE,
      col = c("blue", "darkorange1", "green4"),
     pch =1:3, lty=1:3, title="Tratamiento" )
```

Función par()

Todos estos parámetros y muchos más se pueden modificar mediante la función par().

```
par( mfrow = c(1, 2), mex = 0.5, mgp = c(2, 0.5, 0),
     xaxt='n',  yaxt='n', ann = FALSE )

plot( women, pch =  1:20 )
plot( women, pch = 11:30 )
```

La forma más adecuada de modificar los parámetros que vienen por defecto en un gráfico es definirlos inicialmente mediante la función par(), que afectará a todas las funciones que vengan detrás.

Colores

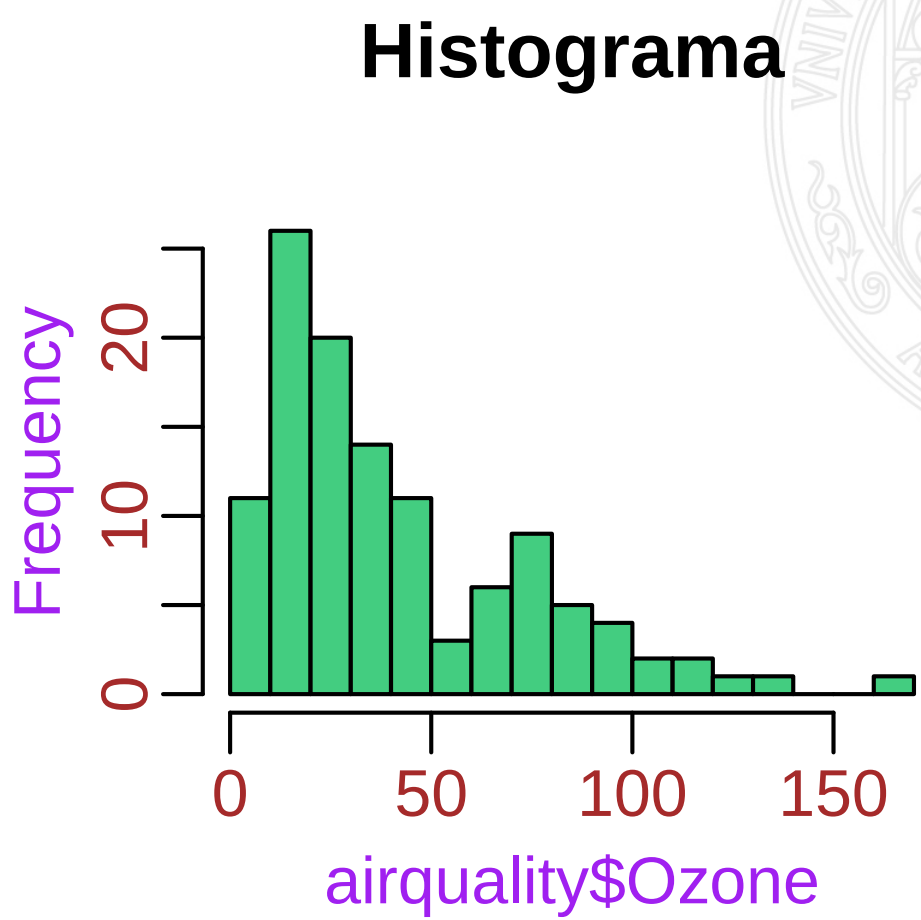
Colores

- `col`: color por defecto del trazado
- `col.axis`: color para los ejes
- `col.lab`: color de las etiqueta de los ejes
- `col.main`: color del título principal
- `col.sub`: color para los subtítulos
- `fg`: color para el primer plano del gráfico (ejes, cajas)
- `bg`: color de fondo

Podemos ver toda la carta de colores con `colors()`

Colores

```
hist( airquality$Ozone, breaks = 13, col= "seagreen3",
 col.lab = "purple", col.axis = "brown",main = "Histograma")
```

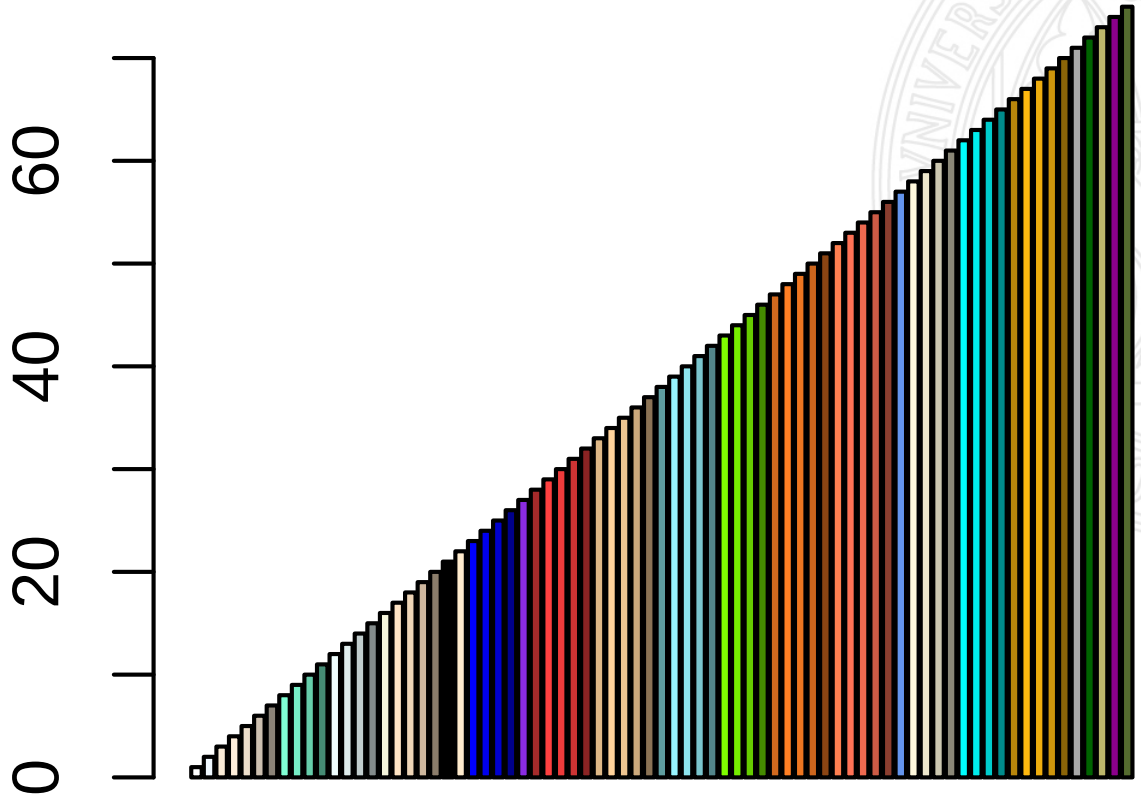


Colores

```
colors()[ 1:25 ]
barplot( 1:75, col = colors(1:75)[1:75] )
```

También podemos crear un vector de `n` colores contiguos utilizando las funciones `rainbow( n )`, `heat.colors( n )`, `terrain.colors( n )`, `topo.colors( n )`, y `cm.colors( n )`.

Colores



Galerías de colores

- *coolors*: `https://coolors.co/`
- *paletton*: `http://paletton.com/`
- *colorBrewer 2.0, color advice for cartography*: `http://colorbrewer2.org/`

Galerías de gráficos con R

Galerías de gráficos con R

- Quick R
- New Energy Research
- R Graphs Gallery (York U)
- R graph gallery (por palabras clave)
- From Data to Graphics (un montón de ejemplos...)
- Un curso/galería, muy elegante.
- YaRrr! The Pirate's Guide to R