

# Introducción al software estadístico

## Módulo VII

Nicolás Schmidt

`nschmidt@cienciassociales.edu.uy`

Departamento de Ciencia Política  
Facultad de Ciencias Sociales  
Universidad de la República

# Estructura de la presentación

## 1 Gráficos

- Gráficos del paquete `graphics`
- Funciones de Alto Nivel
- Funciones de Bajo Nivel
- Colores en R
- Parámetros
- Tipos de gráficos
  - `hist()`
  - `boxplot()`
  - `barplot()`
  - Otros

## 2 Ventanas gráficas

## 3 Division de ventanas gráficas

“Un gráfico puede valer más que mil palabras, pero  
puede tomar muchas palabras hacerlo”

John Tukey

# Estructura de la presentación

## 1 Gráficos

- Gráficos del paquete `graphics`
- Funciones de Alto Nivel
- Funciones de Bajo Nivel
- Colores en R
- Parámetros
- Tipos de gráficos

- `hist()`
- `boxplot()`
- `barplot()`
- Otros

## 2 Ventanas gráficas

## 3 Division de ventanas gráficas

# Estructura de la presentación

## 1 Gráficos

### ■ Gráficos del paquete `graphics`

- Funciones de Alto Nivel

- Funciones de Bajo Nivel

- Colores en R

- Parámetros

- Tipos de gráficos

  - `hist()`

  - `boxplot()`

  - `barplot()`

  - Otros

## 2 Ventanas gráficas

## 3 Division de ventanas gráficas

# Gráficos del paquete `graphics`

Una de las fortalezas de R es su capacidad gráfica. Si bien el paquete `graphics` que viene por defecto con R es muy completo, por fuera de este paquete hay grandes desarrollos en términos de visualización de datos. Los paquetes `ggplot2` y `plotly` son un buen ejemplo de ello.

Hay dos tipos de ordenes gráficas:

- **Alto Nivel:** Son funciones que crean un gráfico.
  - ▶ `plot()`, `hist()`, `boxplot()`, `pairs()`,...
- **Bajo Nivel:** Son funciones que añaden información a un gráfico existente.
  - ▶ `points`, `lines`, `text`, `axis`, `legend`,...

# Estructura de la presentación

## 1 Gráficos

- Gráficos del paquete `graphics`

- **Funciones de Alto Nivel**

- Funciones de Bajo Nivel

- Colores en R

- Parámetros

- Tipos de gráficos

  - `hist()`

  - `boxplot()`

  - `barplot()`

  - Otros

## 2 Ventanas gráficas

## 3 Division de ventanas gráficas

# Funciones de Alto Nivel

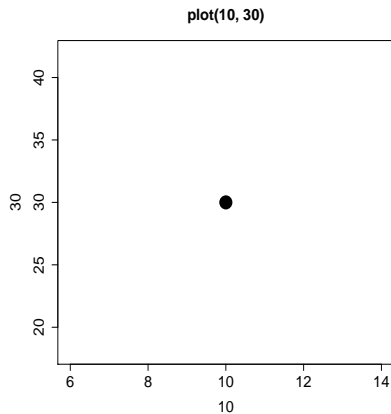
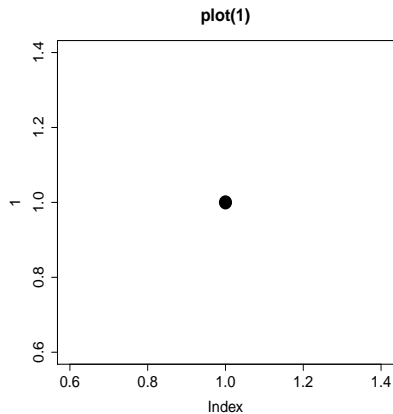
La función genérica para realizar un gráfico en R es `plot()`. Esta función genera un gráfico en función del tipo o tipos de datos de entrada.

Si bien la función `plot` ayuda a visualizar la información lo que debe determinar el tipo de gráfico a utilizar es la estructura de los datos que se desea visualizar y el objetivo que se persigue con lo que se desea comunicar con los datos.

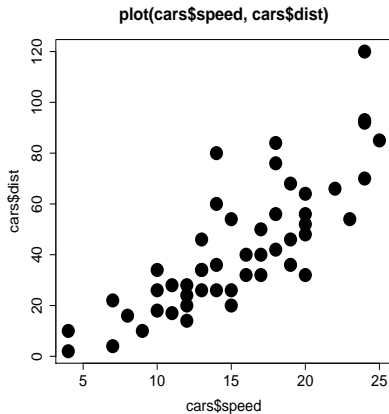
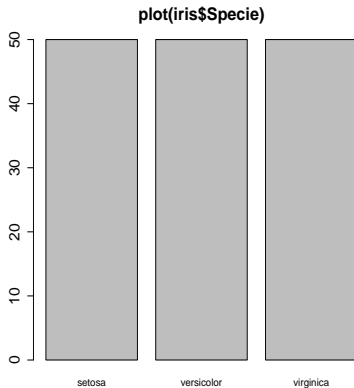
Los gráficos deben estar guiados **estadísticamente**, y luego, si es posible **estéticamente**. Nunca a la inversa!.



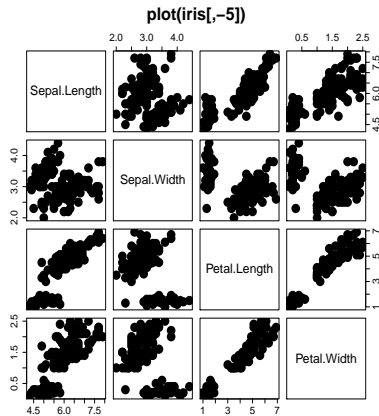
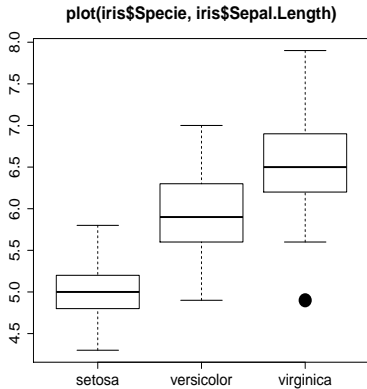
# Funciones de Alto Nivel



# Funciones de Alto Nivel



# Funciones de Alto Nivel

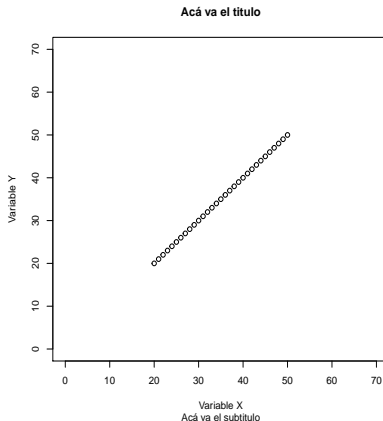
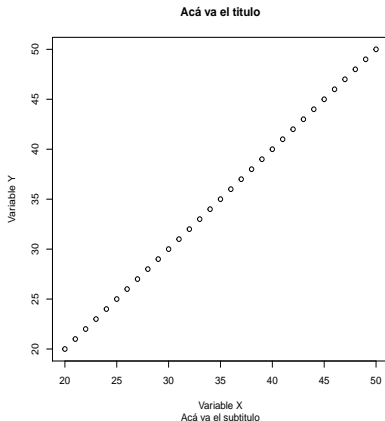


## Argumentos de funciones de alto nivel

Argumento	Descripción	Ejemplo
<code>add = TRUE</code>	Si es TRUE añade un gráfico al existente	
<code>axes = TRUE</code>	Si es FALSE no incorpora ejes ni caja del gráfico	
<code>log = 'x'</code>	Logaritmo del eje x	
<code>log = 'y'</code>	Logaritmo del eje y	
<code>log = 'xy'</code>	Logaritmo de ambos ejes	
<code>xlab</code>	Títulos de los eje x	<code>xlab = 'Eje x'</code>
<code>ylab</code>	Títulos de los eje y	<code>ylab = 'Eje y'</code>
<code>xlim</code>	Limite inferior y superior del eje y	<code>xlim = c(1, 10)</code>
<code>ylim</code>	Limite inferior y superior del eje x	<code>ylim = c(1, 10)</code>
<code>main</code>	Titulo principal	<code>main = 'Titulo'</code>
<code>sub</code>	Subtitulo	<code>sub = 'subtitulo'</code>

# Argumentos de funciones de alto nivel

```
plot(20:50, 20:50, main = "Acá va el titulo", sub = "Acá va el subtítulo",  
     xlab = "Variable X", ylab = "Variable Y")  
plot(20:50, 20:50, main = "Acá va el titulo", sub = "Acá va el subtítulo",  
     xlab = "Variable X", ylab = "Variable Y", xlim = c(0, 70), ylim = c(0, 70))
```

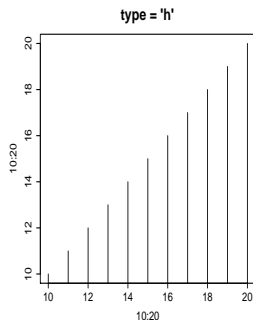
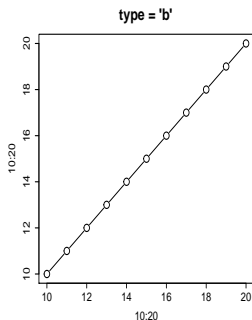
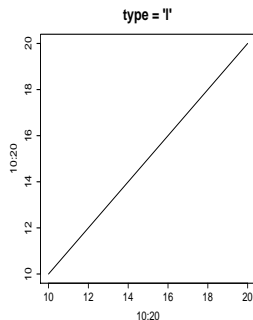


Argumento type = "p"

Tipo	Descripción
type = "p"	Puntos (valor por defecto)
type = "l"	Líneas
type = "b"	Puntos y líneas con espacio
type = ".o"	Puntos y líneas sin espacio
type = "h"	Líneas verticales
type = "s"	Escalera. Inicia hacia la derecha.
type = "S"	Escalera Inicia hacia arriba.
type = "n"	No se dibuja ningún gráfico, solo los ejes.

# Argumentos de funciones de alto nivel

```
plot(10:20, 10:20, type = "l", main = "type = 'l'")  
plot(10:20, 10:20, type = "b", main = "type = 'b'")  
plot(10:20, 10:20, type = "h", main = "type = 'h'")
```



# Estructura de la presentación

## 1 Gráficos

- Gráficos del paquete `graphics`
- Funciones de Alto Nivel
- **Funciones de Bajo Nivel**
- Colores en R
- Parámetros
- Tipos de gráficos
  - `hist()`
  - `boxplot()`
  - `barplot()`
  - Otros

## 2 Ventanas gráficas

## 3 Division de ventanas gráficas



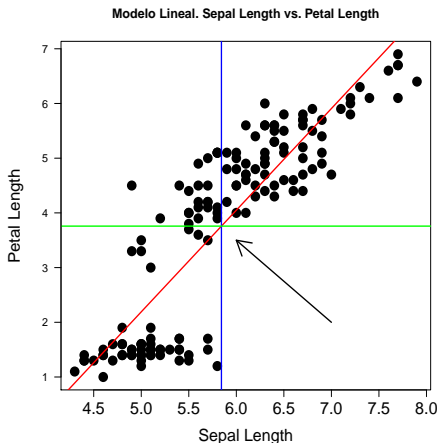
## Argumentos de funciones de alto nivel

Función	Descripción
<code>points(x, y)</code>	Añade puntos
<code>lines(x, y)</code>	Añade líneas
<code>text(x, y, etiquetas, ...)</code>	Añade la etiqueta en la coordenada $xy$
<code>abline(a,b)</code>	Recta de pendiente $b$ y ordenada en el origen $a$
<code>abline(h = y)</code>	Línea horizontal de altura $y$
<code>abline(v = x)</code>	Línea vertical
<code>abline(objeto lm)</code>	Ajusta una recta de regresión
<code>legend(x, y, leyenda...)</code>	Añade leyenda al gráfico
<code>title(main, sub)</code>	Añade título y subtítulo
<code>axis(side, ...)</code>	Añade un eje, <code>side</code> puede ser 1, 2, 3, o 4

```

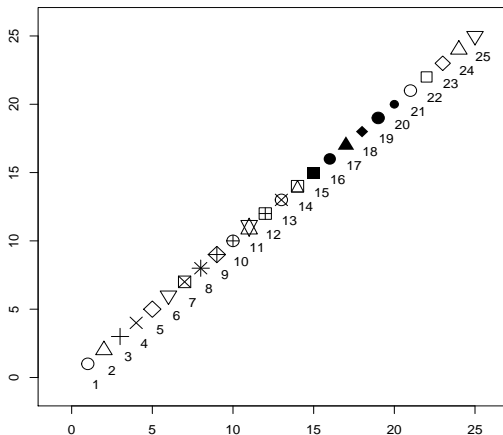
plot(iris[,1], iris[,3], pch = 16, cex = 2, yaxt="n", xlab="Sepal Length",
     ylab="Petal Length", cex.axis = 1.5, cex.lab = 1.5)
abline(lm(iris[,3] ~ iris[,1]), col = "red", lwd = 2)
abline(v=mean(iris[,1]), col="blue",lwd = 2 )
abline(h=mean(iris[,3]), col="green", lwd = 2)
arrows(7, 2, 6, 3.5, lwd = 2)
axis(side = 2, las = 2)
title(main="Modelo Lineal. Sepal Length vs. Petal Length",cex=1.5)

```



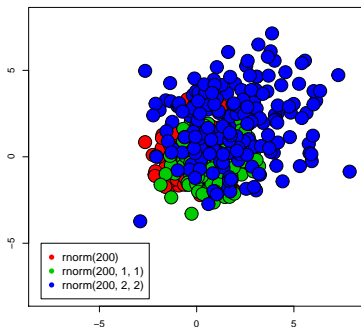
# Argumento pch()

```
plot(1:25, pch = 1:25, cex = 2, xlim = c(-1,26), ylim = c(-1,26), xlab = "",  
     ylab = "")  
text(1:25 + 0.5, 1:25 - 1.5, 1:25 )
```



## Argumento pch()

```
plot(-8:8, -8:8, type = "n", xlab = "", ylab = "", main = "")
points(rnorm(200), rnorm(200), pch = 21, bg = 2, cex = 3)
points(rnorm(200, 1, 1), rnorm(200), pch = 21, bg = 3, cex = 3)
points(rnorm(200, 2, 2), rnorm(200, 2, 2), pch = 21, bg = 4, cex = 3)
legend(-8,-5, c("rnorm(200)", "rnorm(200, 1, 1)", "rnorm(200, 2, 2)"),
       cex = 1.2, col = c(2,3,4), pch = 16)
```



# Estructura de la presentación

## 1 Gráficos

- Gráficos del paquete `graphics`
- Funciones de Alto Nivel
- Funciones de Bajo Nivel
- Colores en R
- Parámetros
- Tipos de gráficos
  - `hist()`
  - `boxplot()`
  - `barplot()`
  - Otros

## 2 Ventanas gráficas

## 3 Division de ventanas gráficas

# Colores en R

R cuenta con una paleta amplia de colores o se pueden establecer colores con `rgb()` o con `hsv()`. A su vez hay una gran cantidad de paquetes que contienen paletas de colores.

Ejemplo:

```
palette()

## [1] "black"    "red"      "green3"   "blue"     "cyan"     "magenta"  "yellow"
## [8] "gray"
```

```
colors()[sample(1:length(colors()), 20)]

## [1] "chartreuse1"    "tan"        "grey13"
## [4] "magenta2"      "gray43"     "plum4"
## [7] "blueviolet"    "grey21"     "aquamarine1"
## [10] "indianred2"    "grey36"     "salmon1"
## [13] "steelblue2"    "palegreen4" "lightgoldenrod1"
## [16] "hotpink4"      "gray31"     "limegreen"
## [19] "paleturquoise1" "pink3"
```

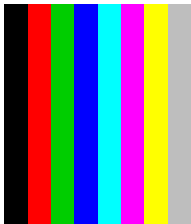
```
length(colors())

## [1] 657
```

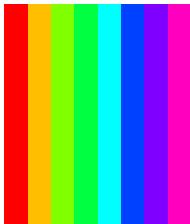
# Colores en R: paquete grDevices

```
barplot(rep(8, 8), col = palette(), axes = FALSE, main = "palette()", cex.main = 2,  
        space = 0, border = NA )  
barplot(rep(8, 8), col = rainbow(8), axes = FALSE, main = "rainbow()", cex.main = 2,  
        space = 0, border = NA )  
barplot(rep(8, 8), col = colors()[sample(1:length(colors()), 8)],  
        axes = FALSE, main = "colors()", cex.main = 2, space = 0, border = NA )
```

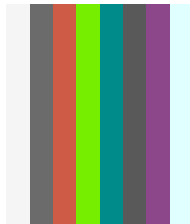
palette()



rainbow()



colors()



# Colores en R: paquete grDevices

```
par(mfrow=c(3,1), mar=c(0.7, 3, 0, 0))

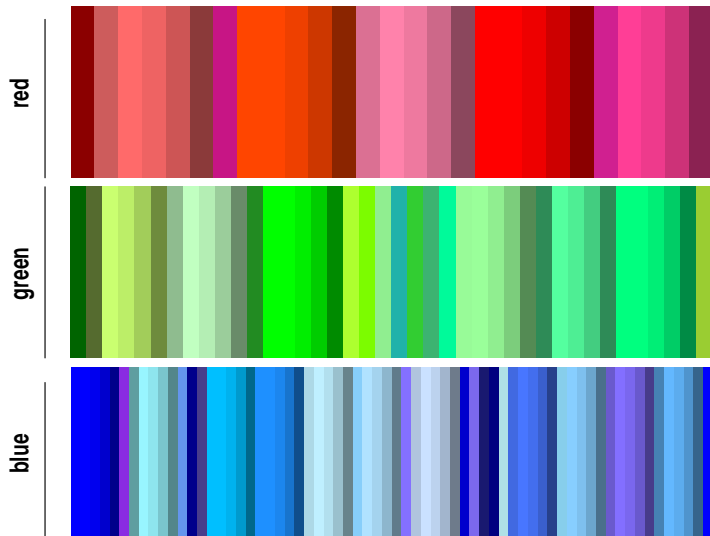
sec <- sum(grepl("red", colors()))
barplot(rep(sec, sec), col = colors()[which((grepl("red", colors())) == TRUE)],
        border = NA, space = 0, axes = FALSE)
axis(2, labels = FALSE, lwd.ticks = 0)
mtext("red", side = 2, line = 1, cex = 1.5, font = 2, las = 0)

sec <- sum(grepl("green", colors()))
barplot(rep(sec, sec), col = colors()[which((grepl("green", colors())) == TRUE)],
        border = NA, space = 0, axes = FALSE)
axis(2, labels = FALSE, lwd.ticks = 0)
mtext("green", side = 2, line = 1, cex = 1.5, font = 2, las = 0)

sec <- sum(grepl("blue", colors()))
barplot(rep(sec, sec), col = colors()[which((grepl("blue", colors())) == TRUE)],
        border = NA, space = 0, axes = FALSE)
axis(2, labels = FALSE, lwd.ticks = 0)
mtext("blue", side = 2, line = 1, cex = 1.5, font = 2, las = 0)
```

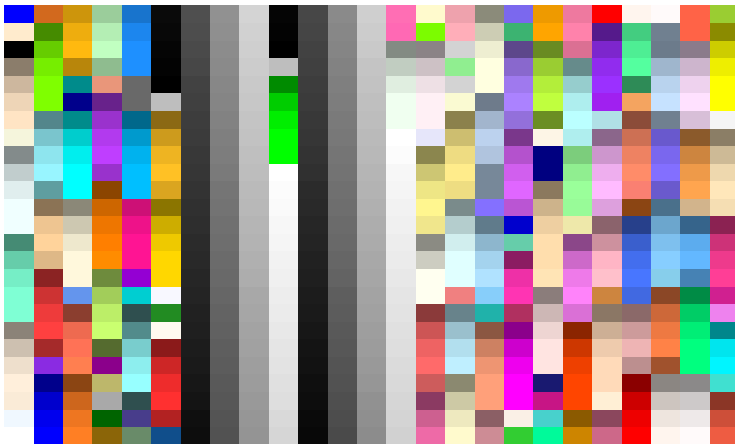


## Colores en R: paquete grDevices



## colors()

```
par(mar = rep(0,4))  
image(matrix(1:625, 25, 25, byrow = TRUE), main = "", col = colors())
```



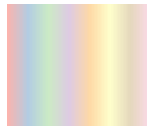
# rgb()

```
animation::saveGIF({  
  par(mfrow=c(1, 1),mar=rep(0, 4))  
  
  sec_rgb <- character()  
  for(i in 1:100){  
    for(j in 1:225){  
      s <- seq(0, 1, 0.02)  
      sec_rgb[j] <- rgb(sample(s, 1), sample(s, 1),  
                        sample(s, 1), sample(s, 1))  
    }  
    image(matrix(1:225, 15, 15, byrow = TRUE), main = "", col = sec_rgb)  
  }  
}, interval = 0.8)
```



# paletas

```
library(RColorBrewer)
paletas <- c("Blues", "Oranges", "Greys", "OrRd", "RdYlBu", "PuOr", "PuBuGn",
             "Pastel1")
par(mfrow = c(2, 4), mar = c(1, 1, 1, 1))
for(i in 1:length(paletas)){
  barplot(rep(10, 100),
          col = colorRampPalette(brewer.pal(n = 8, name = paletas[i]))(100),
          space = 0, border = NA, axes = F)
}
```



# Paquete: paletteer



```
colores <- paletteer::paletteer_c(package = "scico", palette = "oslo", n = 100)
barplot(rep(10, length(colores)), col = colores,
        space = 0, border = NA, axes = F)
```



# Paquete: wesanderson

**BottleRocket1**



**BottleRocket2**



**Rushmore1**



**Royal1**



**Royal2**



**Zissou1**



**Darjeeling1**



**Darjeeling2**



**Chevalier1**



**FantasticFox1**



**Moonrise1**



**Moonrise2**



**Moonrise3**



**Cavalcanti1**



**GrandBudapest1**



**GrandBudapest2**



**IsleofDogs1**



**IsleofDogs2**



# Estructura de la presentación

## 1 Gráficos

- Gráficos del paquete `graphics`
- Funciones de Alto Nivel
- Funciones de Bajo Nivel
- Colores en R
- **Parámetros**
- Tipos de gráficos
  - `hist()`
  - `boxplot()`
  - `barplot()`
  - Otros

## 2 Ventanas gráficas

## 3 Division de ventanas gráficas



## par()

La función `par()` permite controlar todos los aspectos de bajo nivel de un gráfico. Si se ejecuta la función `par()` aparece la lista de todos los argumentos con los valores por defecto. Cada vez que se modifica algún parámetro no hay retorno hasta que vuelva a ser cambiado o al reiniciar sesión. Por eso es recomendable guardar los valores por defecto:

Ejemplo:

```
op <- par(no.readonly = TRUE)
par(op)
```

# par()

Valores por defecto

```
length(par())

## [1] 72

str(par()[1:15])

## List of 15
## $ xlog      : logi FALSE
## $ ylog      : logi FALSE
## $ adj       : num 0.5
## $ ann       : logi TRUE
## $ ask       : logi FALSE
## $ bg        : chr "transparent"
## $ bty       : chr "o"
## $ cex       : num 1
## $ cex.axis  : num 1
## $ cex.lab   : num 1
## $ cex.main  : num 1.2
## $ cex.sub   : num 1
## $ cin       : num [1:2] 0.15 0.2
## $ col       : chr "black"
## $ col.axis  : chr "black"
```

## Parámetros gráficos: par()

Parámetro	Descripción
adj	Justificación del texto adj = 0 → izquierda adj = 0,5 → centrado adj = 1 → derecha
bty	Tipo de caja del gráfico Tipos posible: "o", "l", "7", "c", "u", "]"
cex	Controla el tamaño de símbolos y textos
cex.axis	Tamaño de los valores de los ejes
cex.lab	Tamaño de las etiquetas de los ejes
cex.main	Tamaño del titulo
cex.sub	Tamaño del subtítulo
col.axis	Color de los valores de los ejes
col.lab	Color de las etiquetas de los ejes
col.main	Color del titulo
col.sub	Color del subtítulo
col	Color de símbolos

## Parámetros gráficos: par()

font	Tipo de letra font = 1 → normal font = 2 → cursiva font = 3 → negrita font = 4 negrita y cursiva
las	Controla la orientación de las etiquetas las = 0 →paralelo a los ejes las = 1 →horizontal las = 2 →perpendicular a los ejes las = 3 →vertical
lty	Tipo de linea. Numero entero del 1 al 5
lwd	Ancho de linea. Numero entero
mfcol	Vector de dos enteros. Divide ventana gráfica
mfrow	Vector de dos enteros. Divide ventana gráfica
pch	Tipo de símbolo. Numero entero del 1 al 25
bg	Color de fondo
mar	Vector numerico: c(abajo, izquierda, arriba, derecha)

# Estructura de la presentación

## 1 Gráficos

- Gráficos del paquete `graphics`
- Funciones de Alto Nivel
- Funciones de Bajo Nivel
- Colores en R
- Parámetros
- Tipos de gráficos
  - `hist()`
  - `boxplot()`
  - `barplot()`
  - Otros

## 2 Ventanas gráficas

## 3 Division de ventanas gráficas

# Tipos de gráficos

El tipo de gráfico a utilizar va a depender al menos de dos elementos fundamentales:

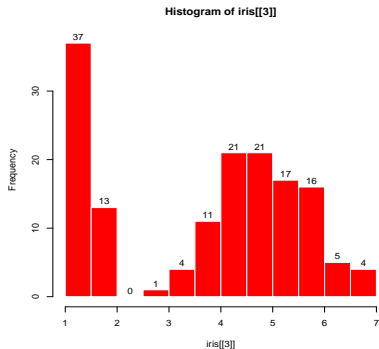
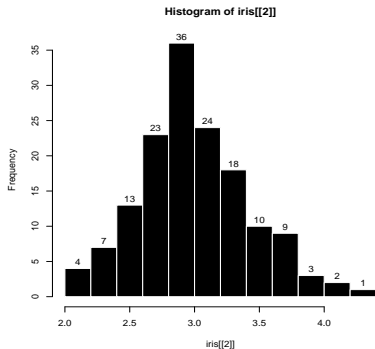
1. El tipo de variable que se va a graficar: cuantitativa o cualitativa.
2. La cantidad de variables que se quiere graficar.

# hist()

Un histograma es una representación gráfica de una variable continua que se visualiza como un diagrama de barras con intervalos constantes.

Ejemplo:

```
hist(iris[[2]], col = 1, border = "white", labels = TRUE)
hist(iris[[3]], col = 2, border = "white", labels = TRUE)
```



Todos los parámetros de construcción de un histograma se pueden modificar. Si desea ver los cálculos hechos por defecto puede guardar como un objeto un histograma y ver el contenido del mismo.

```
h <- hist(iris[[2]], plot = FALSE); h

## $breaks
## [1] 2.0 2.2 2.4 2.6 2.8 3.0 3.2 3.4 3.6 3.8 4.0 4.2 4.4
##
## $counts
## [1] 4 7 13 23 36 24 18 10 9 3 2 1
##
## $density
## [1] 0.13333333 0.23333333 0.43333333 0.76666667 1.20000000 0.80000000
## [7] 0.60000000 0.33333333 0.30000000 0.10000000 0.06666667 0.03333333
##
## $mids
## [1] 2.1 2.3 2.5 2.7 2.9 3.1 3.3 3.5 3.7 3.9 4.1 4.3
##
## $xname
## [1] "iris[[2]]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```



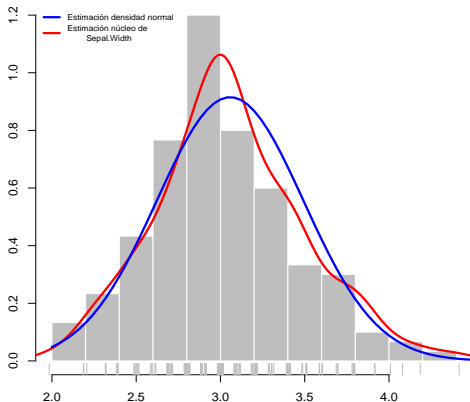
## hist()

Parámetro	Descripción
breaks	Indica los valores de corte de los intervalos de la variable por defecto son todos iguales.
counts	Indica la frecuencia de los valores de la variable en cada intervalo.
density	Cuando el argumento freq es FALSE se calcula la densidad y el eje y cambia. Se calcula como: la frecuencia en el intervalo dividido el numero total datos por el ancho del intervalo. a la unidad.
mids	Punto medio de los intervalos.
equidist	Indica si la distancia entre los intervalos son iguales.

```

hist(iris[[2]], col = "gray", border = "white", freq = FALSE,
     main = "", xlab = "", ylab = "")
lines(density(iris$Sepal.Width),col="red",lwd=3)
curve(dnorm(x,mean = mean(iris$Sepal.Width),sd = sd(iris$Sepal.Width)),from = 2,
      to = 6, add = TRUE, col = "blue", lwd = 3)
legend("topleft", col = c("blue","red"), lwd = 3, bty = "n", cex = 0.7,
      legend = c("Estimación densidad normal","Estimación núcleo de
      Sepal.Width"))
rug(jitter(iris[[2]]), col = "gray")

```



## Ejemplo:

```
col1 <- rgb(1,0.1,0.3,0.2)
col2 <- rgb(0,0.2, 1,0.3)
par(mar=c(3,5,3,3))
layout(matrix(c(1,2,3,3), 2, 2, byrow = TRUE))

hist(iris[[4]], breaks = 20, col = col1, main = "Petal.Width",
     xlab = "", ylab = "", labels = TRUE, axes = FALSE)
axis(1, seq(0, 3))

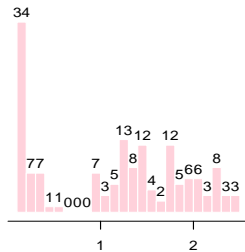
hist(iris[[2]], breaks = 20, col = col2, main = "Sepal.Width",
     xlab = "", ylab = "", labels = TRUE, , axes = FALSE)
axis(1, seq(2, 4.5))

par(mar=c(5,5,0,3))
hist(iris[[4]], breaks = 20, xlim = c(0,4.5), col = col1,
     main = "", xlab = "", ylab = "")

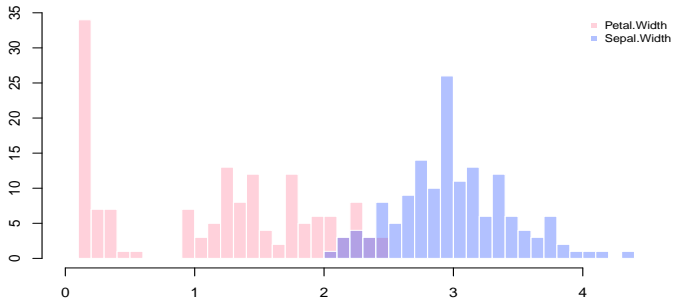
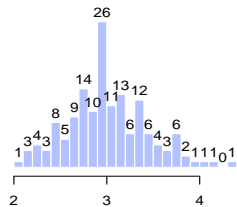
hist(iris[[2]], breaks = 20, xlim = c(0,4.5), col = col2, add = T,
     main = "", xlab = "", ylab = "")

legend(4, 35, legend=c("Petal.Width","Sepal.Width"),
      col=c(col1, col2), pch=15, bty = "n", cex = 0.8)
```

**Petal.Width**



**Sepal.Width**



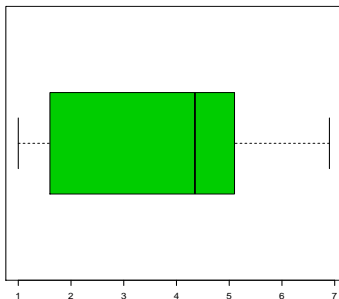
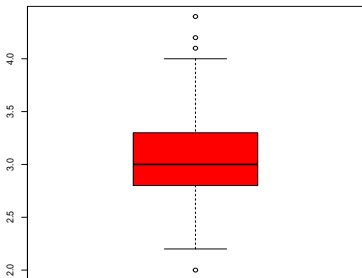
## boxplot()

Un boxplot es un un grafico para una variable cuantitativa que se contruye a partir de Iso cuartiles de la distribucion de los valores de dicha variable.

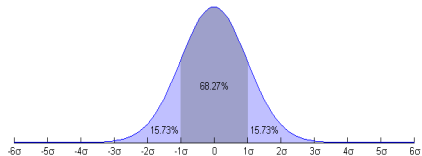
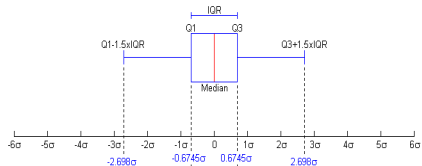
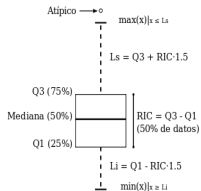
```
boxplot(iris[[2]], col = 2)
fivenum(iris[[2]])
```

```
## [1] 2.0 2.8 3.0 3.3 4.4
```

```
boxplot(iris[[3]], col = 3, horizontal = T)
```

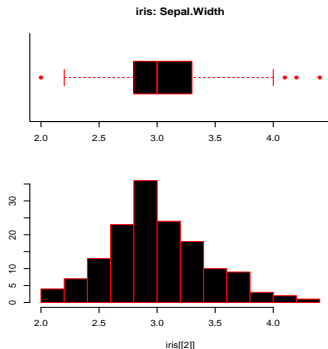
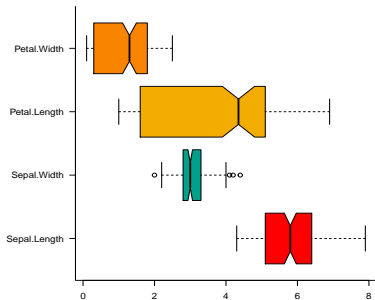


# boxplot()



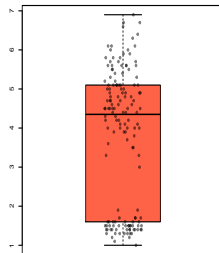
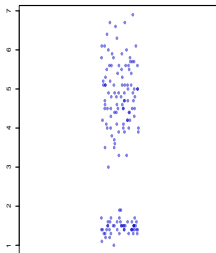
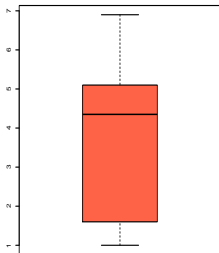
# boxplot()

```
par(mar = c(5.1, 7.1, 4.1, 2.1), bty = "l", bg= "white")
boxplot(iris[,-5], col = wesanderson::wes_palette(name = "Darjeeling1"),
horizontal = T, las = 1, border = 1, notch = TRUE)
par(mfrow = c(2,1))
boxplot(iris[[2]], horizontal = TRUE, col = 1, border = 2,
        main = "iris: Sepal.Width", pch = 16)
par(mar= c(5.1, 7.1, 0, 2.1))
hist(iris[[2]], col = 1, border = 2, ylab = "", main = "" )
```



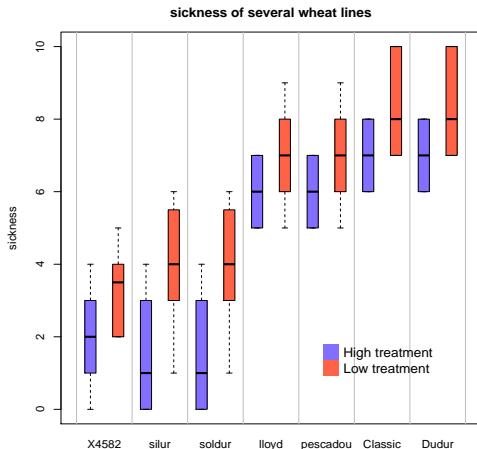
# boxplot()

```
boxplot(iris[[3]], col="tomato", main="")
stripchart(iris[[3]], vertical = TRUE,
  method = "jitter", pch = 20, col = rgb(0,0,0.8,0.5))
boxplot(iris[[3]], col="tomato", main="")
stripchart(iris[[3]], vertical = TRUE,
  method = "jitter", pch = 20, add = TRUE, col = rgb(0,0,0,0.5))
```





# boxplot()



# barplot()

Un barplot es un gráfico de barras que permite visualizar variables cuantitativas.

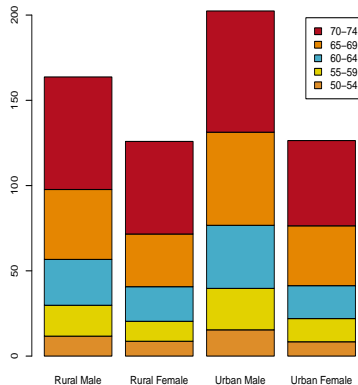
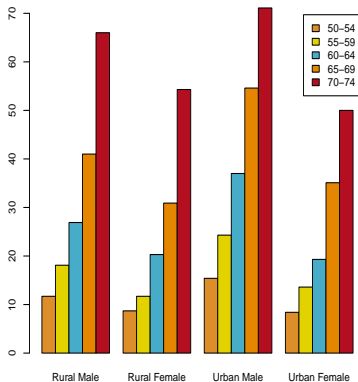
Datos a usar: `datasets::VADeaths`

VADeaths

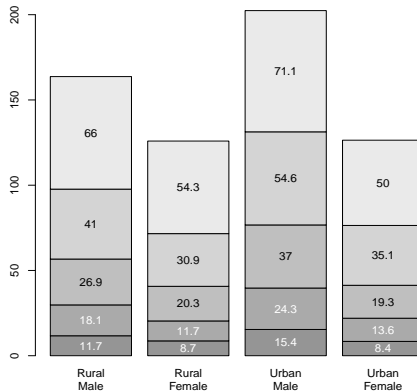
##	Rural	Male	Rural	Female	Urban	Male	Urban	Female
## 50-54	11.7		8.7		15.4		8.4	
## 55-59	18.1		11.7		24.3		13.6	
## 60-64	26.9		20.3		37.0		19.3	
## 65-69	41.0		30.9		54.6		35.1	
## 70-74	66.0		54.3		71.1		50.0	

# barplot()

```
barplot(VADeaths,  
        col = wesanderson::wes_palette(name = "FantasticFox1"),  
        legend = rownames(VADeaths), beside = TRUE)  
barplot(VADeaths,  
        col = wesanderson::wes_palette(name = "FantasticFox1"),  
        legend = rownames(VADeaths), beside = FALSE)
```

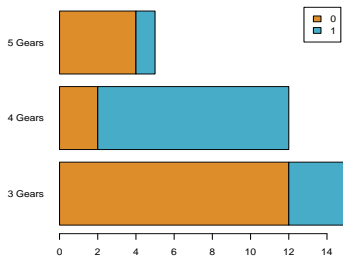
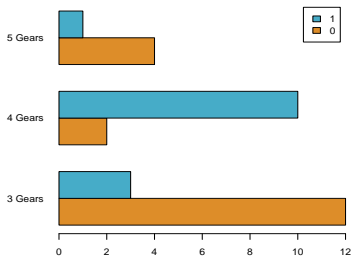


```
midpts <- barplot(VADeaths, col=grey(0.5 + 1:5/12), names=rep("", 4))
mtext(sub(" ", "\n", colnames(VADeaths)),at=midpts, side=1, line=1.5, cex=1)
text(rep(midpts, each=5), apply(VADeaths, 2, cumsum) - VADeaths/2,
      VADeaths, col=rep(c("white", "black"), times=2:3, cex=0.8))
```



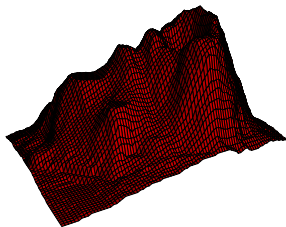
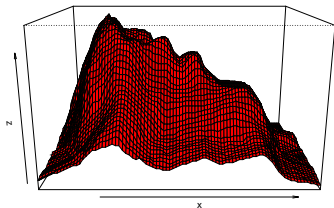
## barplot(): datasets::mtcars

```
par(las = 1, cex = 1.2)
tabla <- with(mtcars, table(vs, gear))
barplot(tabla,
        col = wesanderson::wes_palette(name = "FantasticFox1")[c(1,3)],
        legend = rownames(tabla), beside = TRUE, horiz = TRUE,
        names.arg=c("3 Gears", "4 Gears", "5 Gears"))
barplot(tabla,
        col = wesanderson::wes_palette(name = "FantasticFox1")[c(1,3)],
        legend = rownames(tabla), beside = FALSE, horiz = TRUE,
        names.arg=c("3 Gears", "4 Gears", "5 Gears"))
```



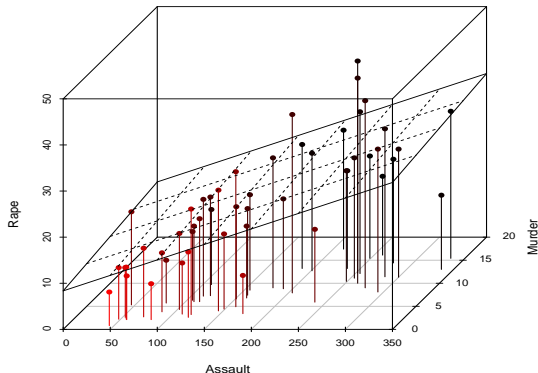
## Perspective Plots con persp()

```
z <- 5 * volcano
x <- 10 * (1:nrow(z))
y <- 10 * (1:ncol(z))
persp(x, y, z, col = 2, scale = FALSE)
persp(x, y, z, theta = 150, phi = 20, col = 2, scale = FALSE,
      ltheta = -120, shade = 0.55, box = FALSE)
```



# Modelo lineal con scatterplot3d()

```
modelo <- lm(Rape ~ Assault + Murder, data = USArrests)
library("scatterplot3d")
grafico <- scatterplot3d(USArrests[,c(2,1,4)],
  type = "h", highlight.3d = TRUE, angle = 45, pch = 16)
grafico$plane3d(modelo, lty.box = "solid")
```



# Estructura de la presentación

## 1 Gráficos

- Gráficos del paquete `graphics`
- Funciones de Alto Nivel
- Funciones de Bajo Nivel
- Colores en R
- Parámetros
- Tipos de gráficos
  - `hist()`
  - `boxplot()`
  - `barplot()`
  - Otros

## 2 Ventanas gráficas

## 3 Division de ventanas gráficas



# Ventanas gráficas

Es posible querer hacer un grafico sin que sobrescriba uno o tambien es posible haer un grafico y que directamente se guarde en una ruta determinada sin que pase por una ventana gráfica.

R cuenta con varias funciones para controlar estos dispositivos:

```
windows()      # nuevo dispositivo grafico en windows  
x11()          # nuevo dispositivo grafico en Linux  
macintosh()    # nuevo dispositivo grafico en mac
```

En caso de querer guardar un grafico sin pasar por una ventana se pueden usar alguna de estas funciones:

```
postscript()  
pdf()  
png()
```

Es importante recordar que el ultimo dispositivo abierto es el que queda activo.

# Ventanas gráficas

Es importante recordar que el ultimo dispositivo abierto es el que queda activo.

```
windows()
dev.list()           # muestra los dispositivos abiertos

##      pdf windows
##      2       3

dev.cur()           # muestra el dispositivo activo

## windows
##      3

dev.set(2)          # se cambia de dispositivo activo

## pdf
##      2

dev.off(3)          # si no se especifica numero se cierra el activo

## pdf
##      2
```

# Estructura de la presentación

## 1 Gráficos

- Gráficos del paquete `graphics`
- Funciones de Alto Nivel
- Funciones de Bajo Nivel
- Colores en R
- Parámetros
- Tipos de gráficos
  - `hist()`
  - `boxplot()`
  - `barplot()`
  - Otros

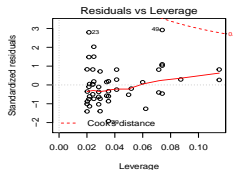
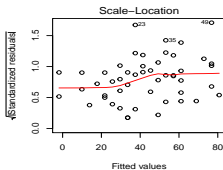
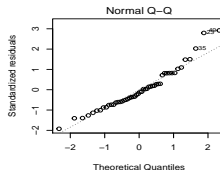
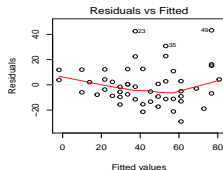
## 2 Ventanas gráficas

## 3 Division de ventanas gráficas

# Division de ventanas gráficas

Con la función `par()` se puede establecer el número de divisiones con el argumento `mfrow = c(2,2)`

```
par(mfrow = c(2, 2))  
plot(lm(dist~speed, data = cars))
```



# Division de ventanas gráficas

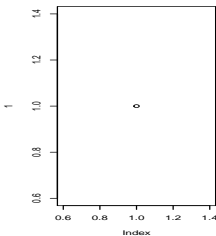
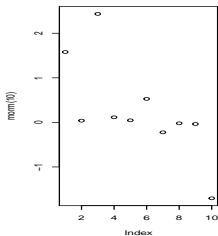
La función `split.screen()` controlar multiples sectores en un dispositivo gráfico.

```
split.screen(c(1, 2))
```

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

```
screen(2); plot(1)
```

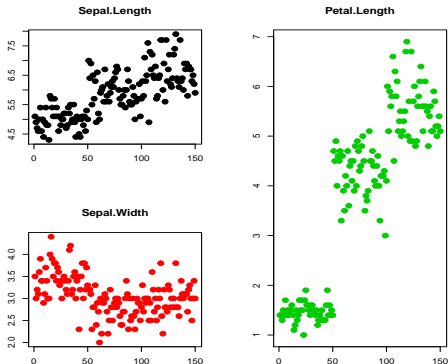
```
screen(1); plot(rnorm(10))
```



# Division de ventanas gráficas

La función `layout()` permite dividir un dispositivo gráfico de múltiples maneras

```
layout(matrix(c(1:3, 3), 2, 2))
for(i in 1:3){
  plot(iris[,i], pch = 20, col = i ,
       cex = 2, xlab = "", ylab = "", main = names(iris)[i])}
```

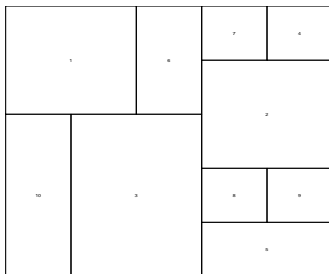


# Division de ventanas gráficas

```
vec <- c(1, 1, 10, 10, 10, 1, 1, 3, 3, 3, 6, 6, 3, 3, 3, 7, 2, 2, 8, 5, 4, 2, 2, 9, 5)
m <- matrix(vec, 5, 5); m
```

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

```
layout(m); layout.show(10)
```



## Final: sitios interesantes!

- The R graph gallery: [aquí](#)
- accidental aRt!: [tumblr](#), [twitter](#)
- htmlwidgets: [aquí](#)
- DiagrammeR [aquí](#)