Gráficos

Carlos Iván Espinosa

13/1/2021

—

output: revealjs::revealjs_presentation: transition: default self_contained: false css: styles.css center: true mathjax: NULL —

Gráficos con R

Carlos Iván Espinosa

Objetivo

Comprender los parámetros gráficos básicos en R y tener la capacidad de desarrollar gráficos personalizados

Contenidos

1. Introducción

La representación gráfica de los datos es una actividad bastante común, en todas partes vemos gráficos representando información de todo tipo.

El hacer una buena gráfica no es cuestión sencilla y requiere un gran trabajo de abstracción.

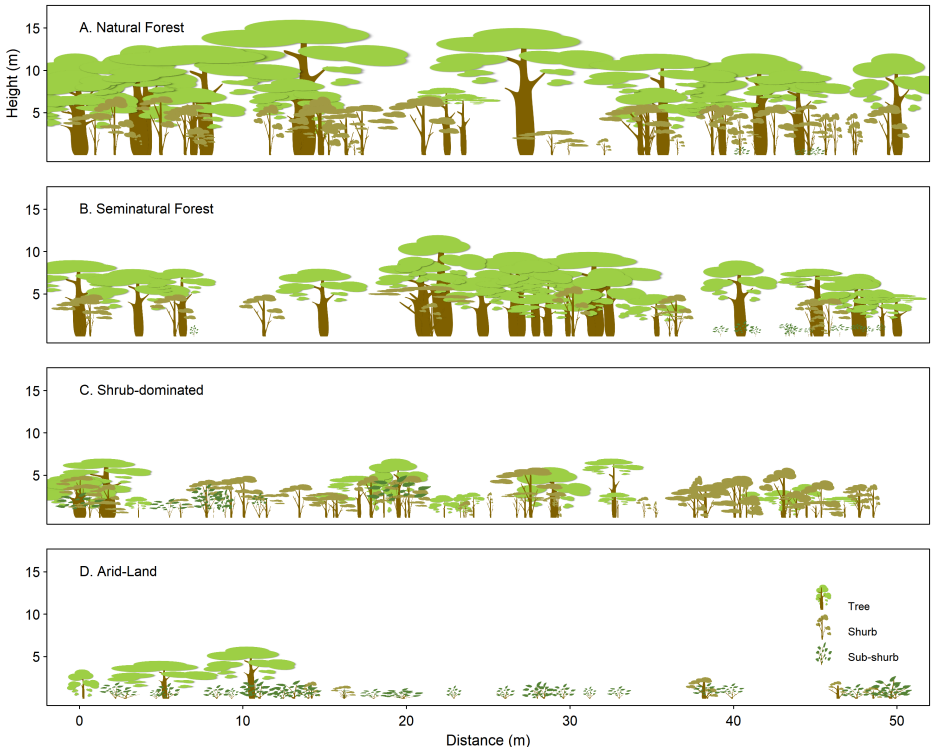
La visualización de datos

Nuestros cerebros están mucho más adecuados a captar imágenes más que números o letras, así que las gráficas es un medio visual para presentar información. De esta forma, el fin último de un gráfico es **resumir** información de tal forma que muestre tendencias y que permita comparaciones.

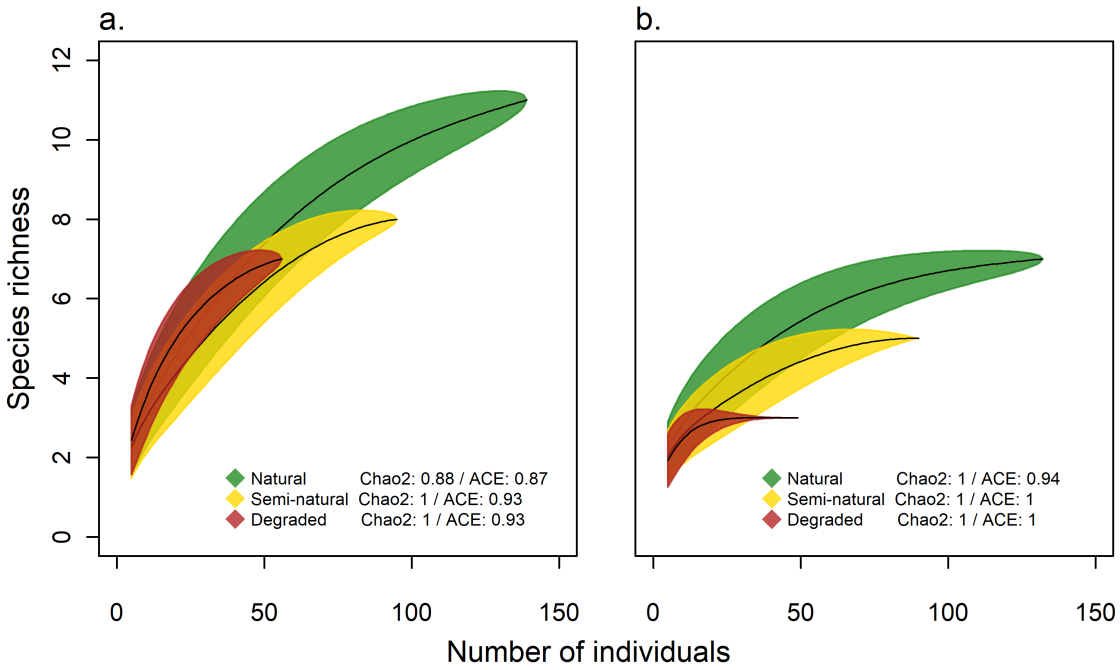
Principios de una buena gráfica

- Captar la atención del lector.
- El gráfico debe presentar la información de la forma más sencilla, clara y precisa.
- No inducir al error.
- Los gráficos deben permitir al lector la comparación de datos, mostrar tendencias o diferencias entre las variables.
- Relación datos-tinta propuest por Edward Tufte

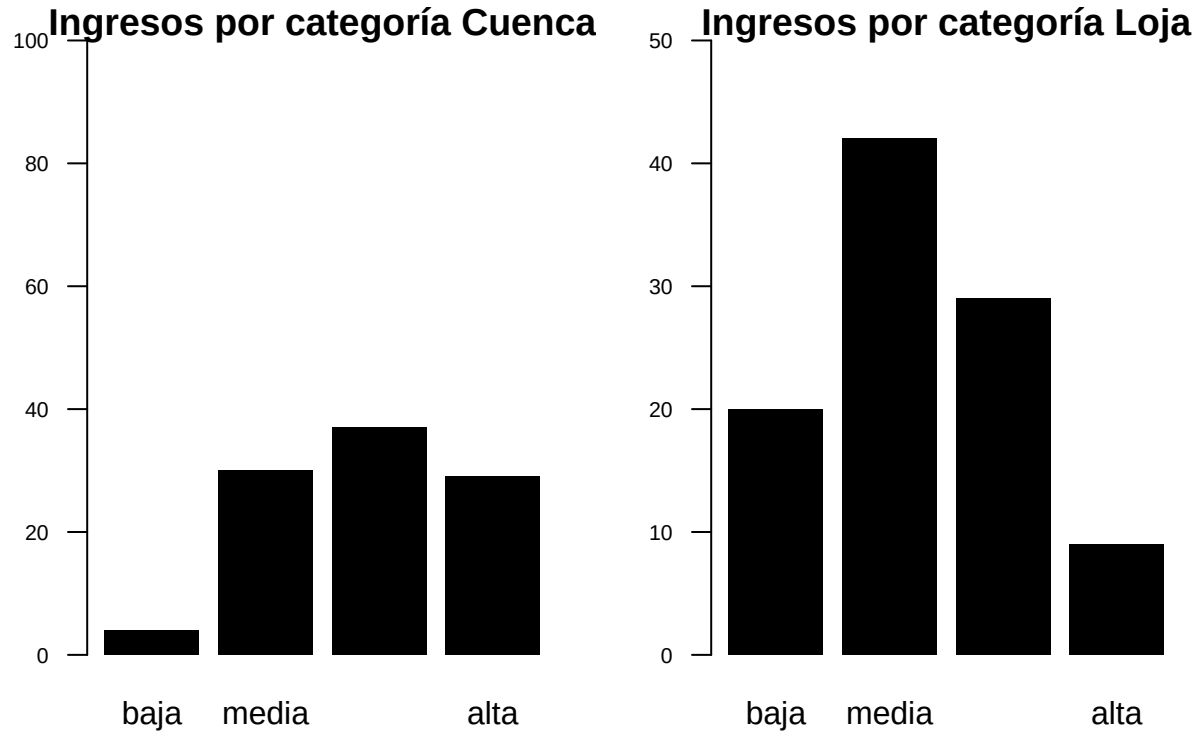
Llamar la atención

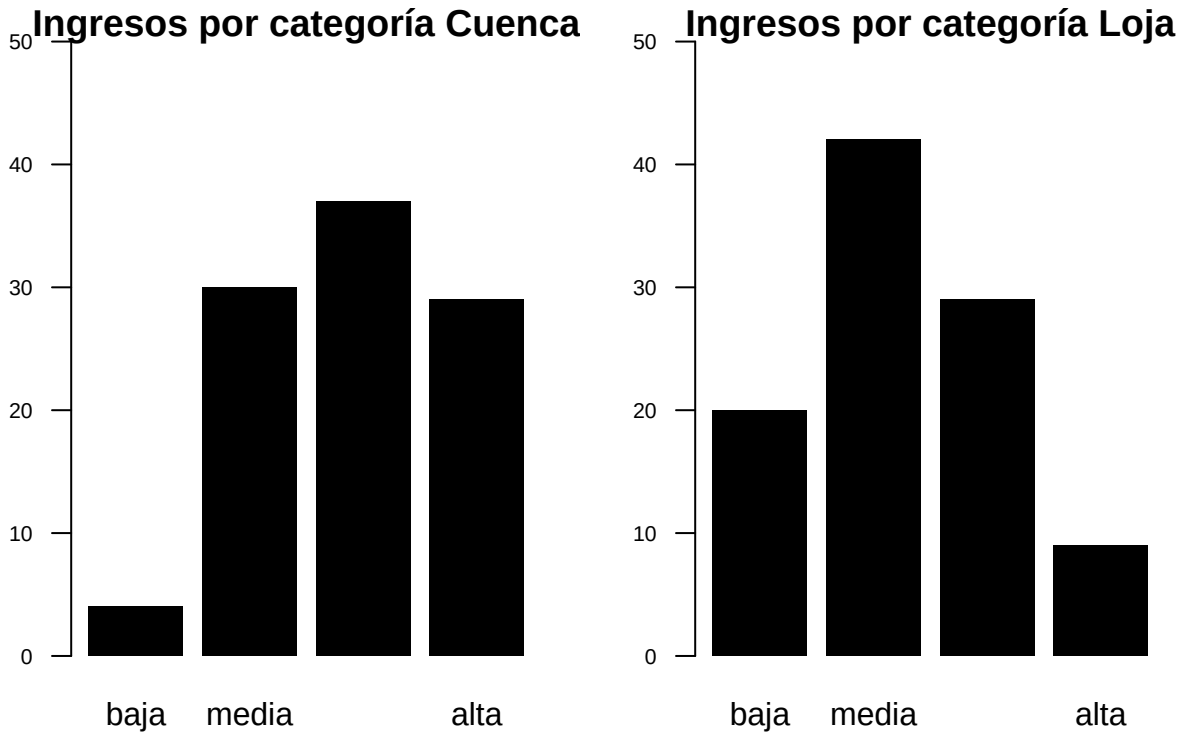


Presentar información clara



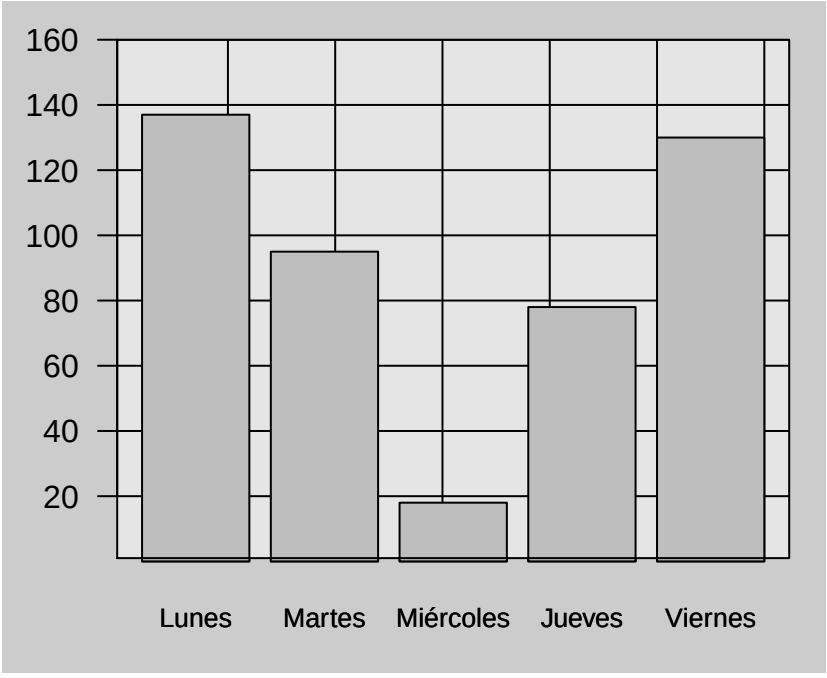
No inducir al error



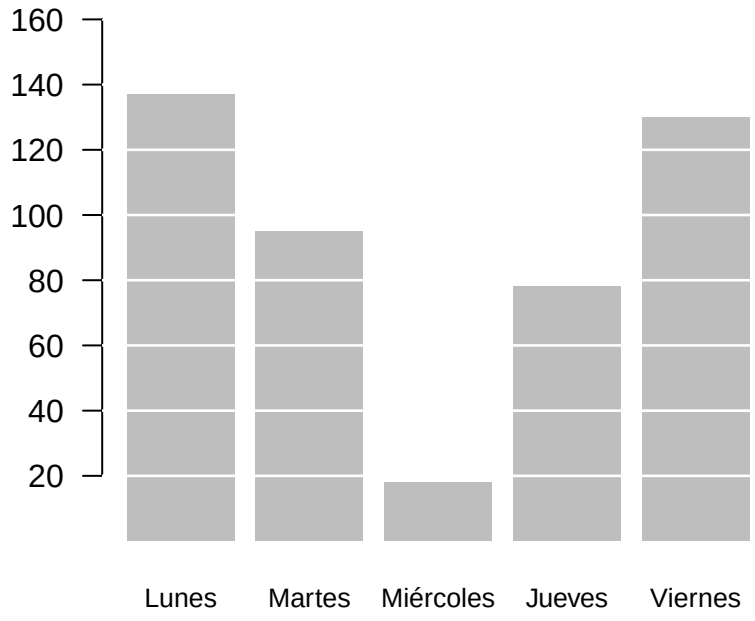


Relación datos - tinta

Bajo ratio Datos-Tinta



Alto ratio Datos-Tinta



Muchas otras reglas han sido discutidas mucho más de lo que uno realmente piensa, pero creo que lo más importante es entender para que quiero hacer un gráfico y asegurarme que este esté dando el mensaje que quiero que dé.

El sistema base de R para gráficas

Una de las mayores fortalezas de R es la versatilidad para generar gráficos de alta calidad, existen muchos paquetes que permiten hacer gráficos de diferentes características como `ggplot` o `lattice`, sin embargo, ahora nos centraremos en el sistema base.

El lienzo donde graficar

Cuando estamos realizado gráficos en R, al contrario de lo que hemos visto hasta el momento, R envía el resultado de nuestros códigos a un dispositivo gráfico (una ventana) y no a un objeto.

Esta ventana esta configurada con unas características por defecto. De ahora en adelante le llamaremos a esta ventana lienzo.

Modificando el Lienzo

```
par(bg="grey98", mar=c(3,3,3,3))
```

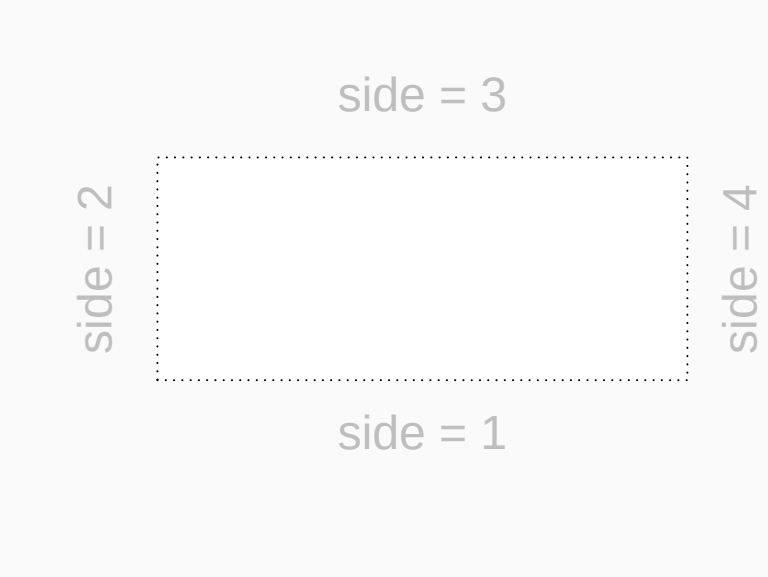
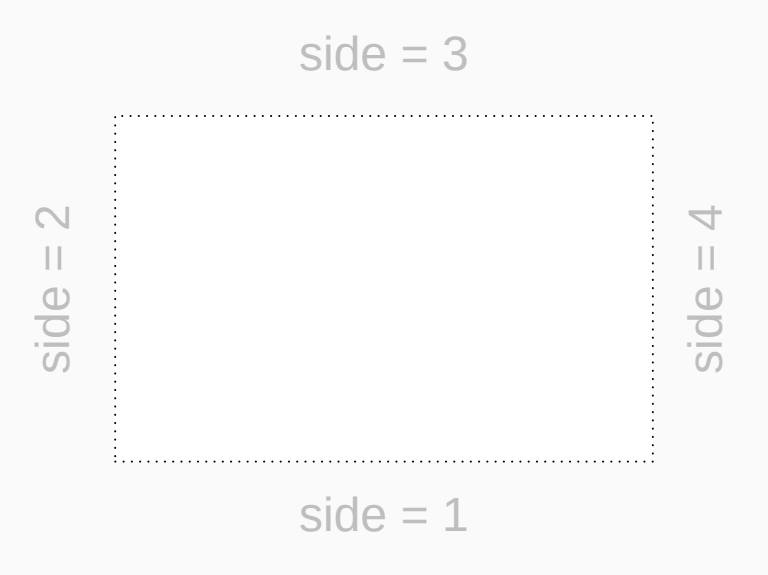


Figure 1: Lienzo con características por defecto



Dividiendo el lienzo

La función `mfcol` me permite generar un lienzo con varias partes iguales, puedo decir cuántas filas y cuantas columnas. `mfcol=c(f,c)`

```
par(bg="grey98", mar=c(3,3,3,3), mfcol=c(1,2))
```

Dividiendo el lienzo

```
layout(matrix(c(1,2,3,3), 2, 2))
par(mar=c(1,1,1,1), bg="grey98")
```

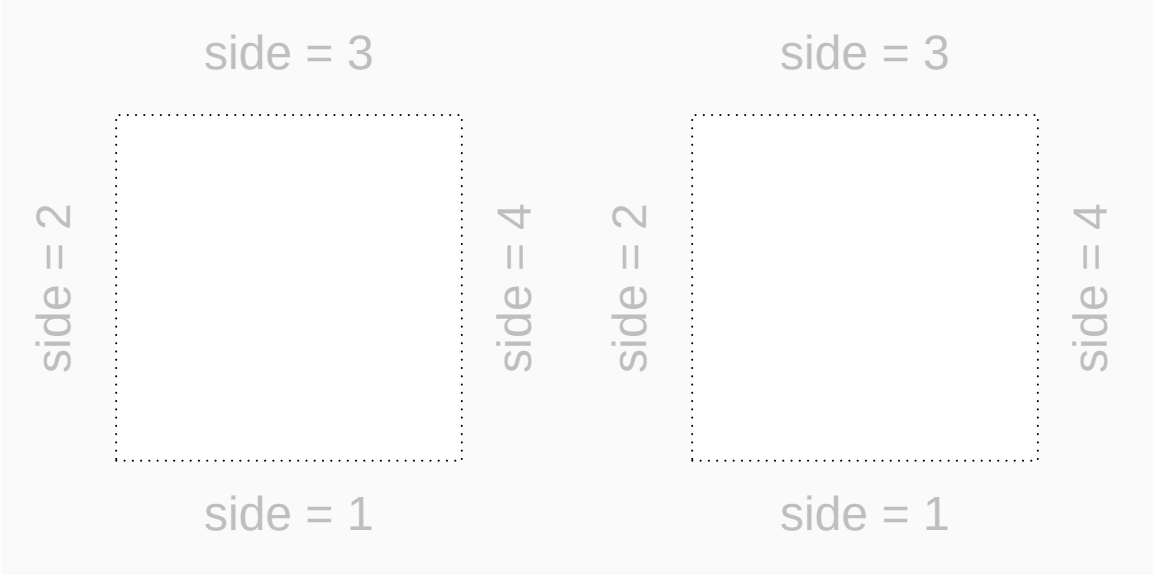


Figure 2: Lienzo con dos gráficos de igual tamaño

Figure 3: Lienzo con dos gráficos de tamaño variable

Controlando el tamaño

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE),
       widths = c(2,3), heights = c(1.5,3))
par(mar=c(1,1,1,1), oma=c(3,3,1,1), bg="grey98")
```

Figure 4: Lienzo con dos gráficos de tamaño variable 2

Parámetros gráficos

Parámetros gráficos

Aunque no lo sabían, ya hemos utilizado algunos parámetros gráficos en la función `par`, estos nos han servido para manipular el lienzo. Usamos *mar* para cambiar márgenes de las gráficas, *oma* para el margen del conjunto de gráficas, además, *bg* para poner un color al fondo de la gráfica, y *mfcol* para dividir el lienzo en varias partes iguales.

```
par(mar=c(1,1,1,1), oma=c(3,3,1,1), bg="grey98", mfcol=c(2,2))
```

Los puntos

```
x <- rep(1:5, 5)
y <- sort(rep(1:5, 5))
par(mar=c(1,1,1,1))
plot(x,y, pch=1:25, col=1:5, cex=seq(1,3,length.out = 25), bg="yellow",
     axes=FALSE, xlab="", ylab="", ylim=c(0,6))
text(x,y-0.3, cex=0.8)
```

Otros puntos

En la gráfica inicial se uso figuras como puntos. Veamos como podemos hacer esto.

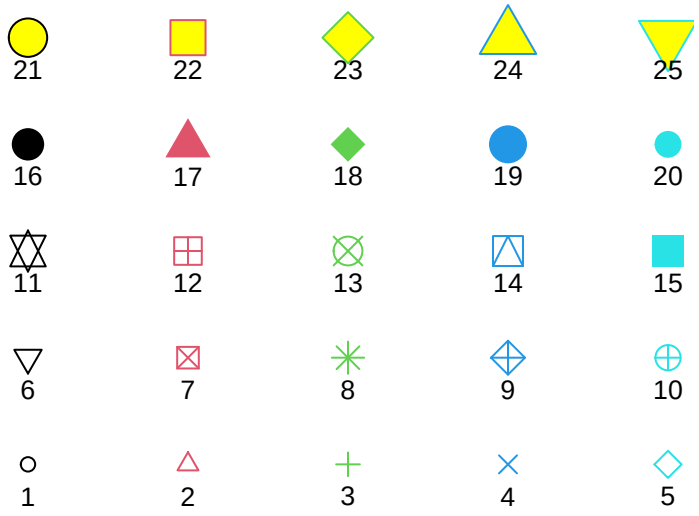


Figure 5: Tipos de símbolos gráficos desplegados

Primero vamos a leer nuestra imagen.

```
library(png)
##Leerlo del disco duro
coli <- readPNG("Bird.png")

#Leerlo de internet y guardarlo
pngURL <- "http://www.pngall.com/wp-content/uploads/4/Single-Flying-Bird.png"
download.file(pngURL, "birdD.png", mode = "wb")
```

Desde internet

```
##Leerlo directamente del internet
library(magick)
```

```
FALSE Linking to ImageMagick 6.9.11.34
FALSE Enabled features: cairo, freetype, fftw, ghostscript, lcms, pango, rsvg, webp
FALSE Disabled features: fontconfig, x11
```

```
birdD <- image_read('http://www.pngall.com/wp-content/uploads/4/Single-Flying-Bird.png')
```

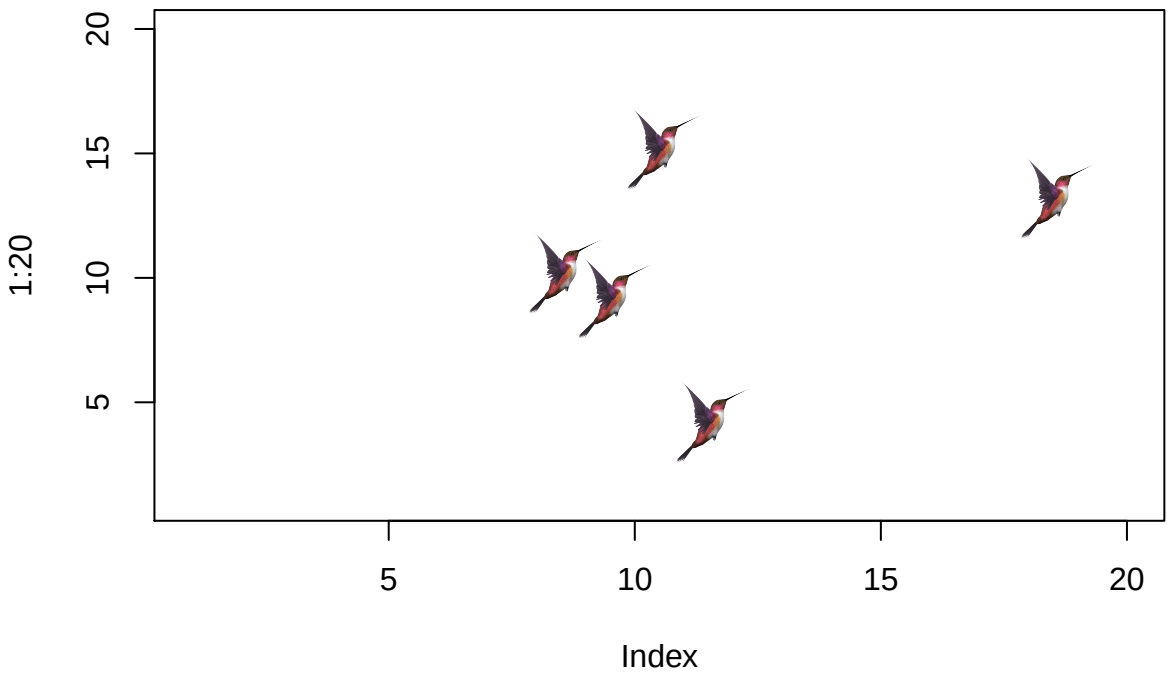
Usando la imagen

```
set.seed(25)
x <- sample(1:18, 5)
set.seed(23)
```

```r
y <- sample(1:18, 5)

plot(1:20, type="n")

for(i in 1:5) rasterImage(birdD, x[i], y[i],x[i]+3, y[i]+4.5,)
```



Las líneas

También podemos dar formato a las líneas, modificando el tipo de línea y su tamaño. Veamos cómo podemos hacer esto.

```r
lin <- matrix(c(rep(1,5), rep(2,5), rep(3,5), rep(4,5), rep(5,5)), 5,5)
par(mar=c(1,1,1,1))
matplot(lin, type="l", lty=1:5, lwd=seq(1,3,length.out = 5),
        axes=FALSE, ylab="")
```

Anatomía de una gráfica

```r
plot(cars, xlab="Velocidad (mph)", ylab = "Distancia",
     main="Relación entre la velocidad de un vehículo
     y la distancia de frenado", cex=0.8, pch=19, cex.main=0.9,
     cex.axis=0.7, cex.lab=0.85)
mtext("Eje x",side = 1,at = 27, font=2, line=0.4, cex=0.8, col="blue")
```

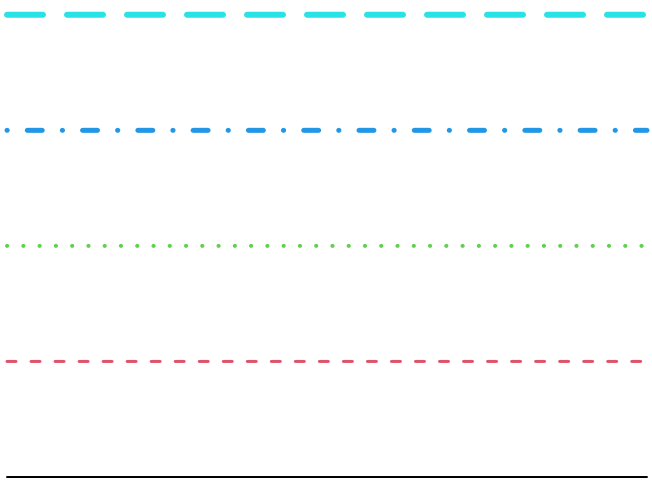


Figure 6: Tipos de líneas desplegadas en gráficos

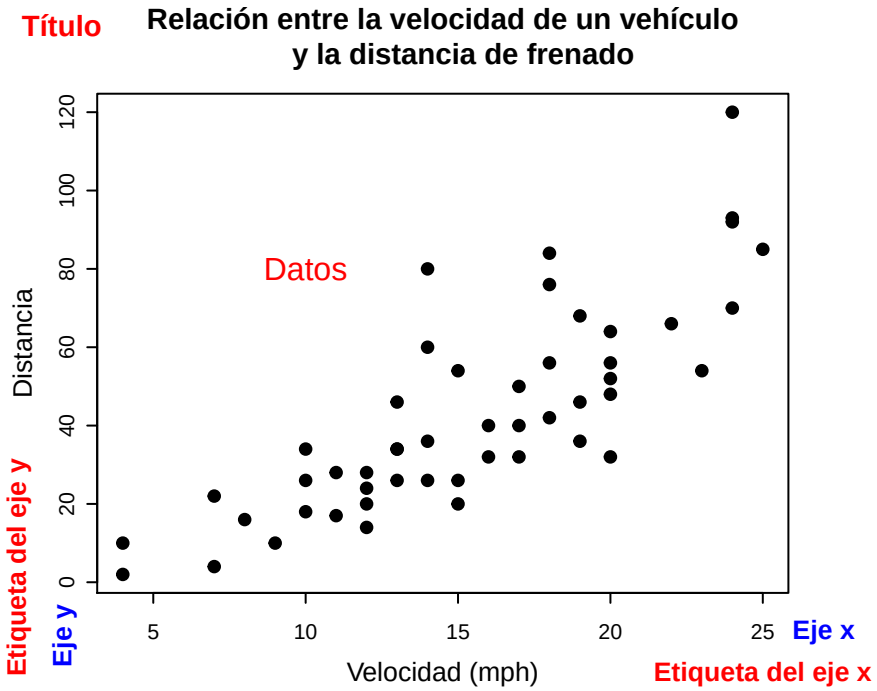


Figure 7: Anatomía de un gráfico

Los gráficos

- Si estamos interesados realizar un gráfico con dos variables contínuas (un gráfico de dispersión) usamos plot
- Si queremos realizar un gráfico con dos variables una continua y una discreta usamos boxplot.
- Si nos interesa realizar un gráfico de valores únicos de variables discretas usamos barplot o plot con type = "h"

Graficando con plot

```
set.seed(3)
x <- rnorm(20)
y <- rnorm(20)

#Graficamos
plot(x,y)
```

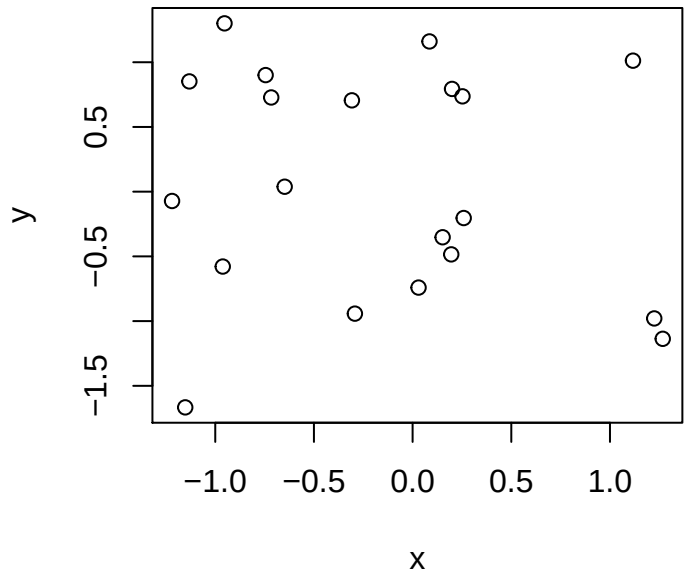


Figure 8: Uso de plot, el primer paso

Graficando con boxplot

```
x <- sample(letters[1:4], 20, replace=TRUE)
y <- rnorm(20)
```

```r
#Graficamos
boxplot(y~x)
```

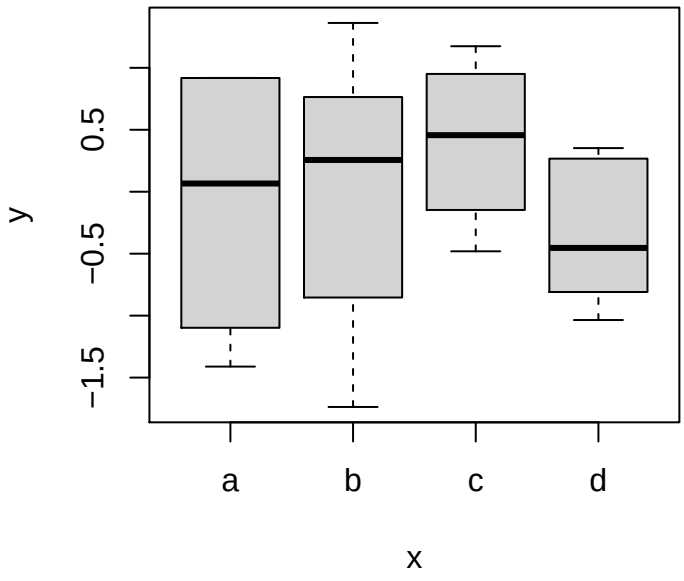


Figure 9: Uso de boxplot, el primer paso

Graficando con barplot

```r
x <- sample(letters[1:4], 20, replace=TRUE)
y <- rnorm(20)

#Graficamos
barplot(tapply(y,x, mean))
```

Graficando valores medios plot

```r
x <- sample(letters[1:4], 20, replace=TRUE)
y <- rnorm(20)

#Graficamos
plot(tapply(y,x, mean), type="h")
```

Ejercicio

Genere una gráfica basado en los datos que tiene. Use las diferentes funciones que hemos generado para diseñar su gráfica.

Puede acceder a a lección en: https://ciespinosa.github.io/Graficando/

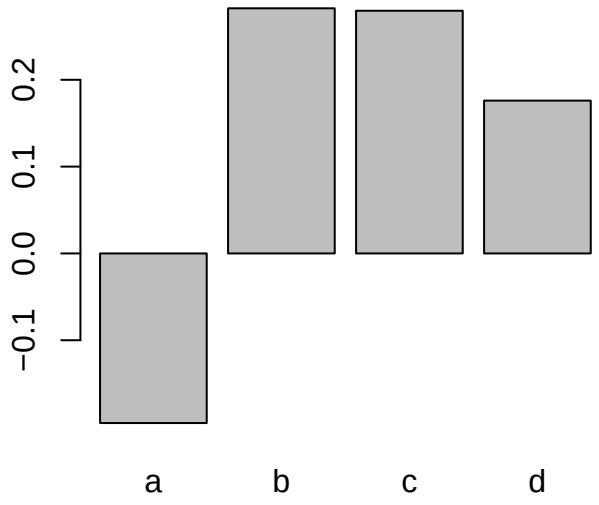


Figure 10: Uso de boxplot, el primer paso

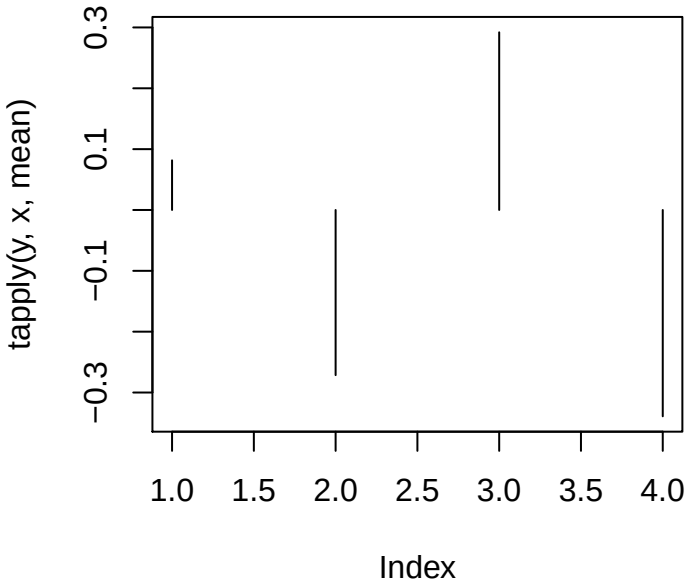


Figure 11: Uso de plot, el primer paso