Graficando con R. El inicio

Carlos Iván Espinosa

Octubre de 2016

Contents

Introducción

La representación gráfica de los datos es una actividad bastante común, en todas partes vemos gráficos representando información de todo tipo, los vemos en textos científicos, divulgativos o en informes de organismos públicos o privados. Sin embargo, estos no siempre cumplen con los estándares de una buena gráfica.

El hacer una buena gráfica no es cuestión sencilla y requiere un gran trabajo de abstracción. En esta lección intentaremos dar algunas pautas para poder desarrollar gráficas claras y esperamos que bonitas. Claro como siempre trabajaremos con R.

Pero antes de iniciar con R, es necesario aclaremos algunas cosas importantes. Intentaremos definir algunos principios básicos e importantes que deben ser tomados en cuenta para realizar una buena gráfica, pasaremos luego a entender la anatomía de una gráfica. Con estos insumos si pasaremos a conocer lo que nos ofrece R para hacer gráficas.

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 debemos pensar que el fin último de un gráfico es resumir información de tal forma que muestre tendencias y que permita comparaciones.

Existen muchos principios para realizar una buena gráfica, intentaremos resumir algunos de ellos.

- Captar la atención del lector. Este posiblemente es uno de los retos más grandes cuando estamos haciendo una gráfica, queremos que esta llame la atención y el lector se interese por la misma.

- El gráfico debe presentar la información de la forma más sencilla, clara y precisa. Por lo tanto debemos asegurarnos que no tenemos información de más, si existe algo que borrando no le resta al entendimiento del gráfico bórrelo.
- No inducir al error. Esta es una regla que muchos autores coinciden, los gráficos tienen el fin de presentar información, por tanto deben mostrar lo que esa información dice.
- Los gráficos deben permitir al lector la comparación de datos, mostrar tendencias o diferencias entre las variables.

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 graficas

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.

Antes de empezar, para que se hagan una idea de algunas de las cosas que podemos hacer con R, permitamos a R que nos dé una muestra de los gráficos que se pueden hacer.

```
demo(graphics) # Ejecútela usted
```

Sorprendid@, bueno eso no es nada hay muchas cosas que se pueden hacer con gráficos en R.

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. En el caso de RStudio es una ventana propia. Esta ventana es como el lienzo donde dibujaremos el gráfico.

Cuando yo ejecuto un gráfico, por ejemplo con la función `plot`, la *ventana* (dispositivo gráfico) se abre automáticamente, para cerrar esta ventana puedo utilizar el comando `dev.off()`.

La ventana tiene por defecto algunas características que pueden ser modificadas con la función `par`. Veamos la ventana que sale por defecto.

```
par(bg="grey98")
plot(1:100, type="n", axes=F, xlab = "", ylab="", bg="white")
rect(-10, -10, 120, 120, col="white")
box(lty = 3)
mtext(c("side = 1", "side = 2", "side = 3", "side = 4"),
side = c(1, 2, 3, 4), col = "grey", line = 1, cex = 1.5)
```

```
#dev.off()
```

He modificado una característica, el fondo del lienzo, lo normal es que el lienzo sea de color blanco y yo le he dicho que sea de color *grey98*. Luego volveremos a este punto.

Como se ve en el gráfico inicialmente podemos ver dos partes distintas, la región del plot (el cuadrado en blanco), es la zona donde se dibujaran los datos. El área en gris son los márgenes de cada uno de los lados (side). Cada lado siempre mantendrá ese orden, el lado uno es el inferior, el dos el izquierdo, el tres el superior y el cuatro el derecho. Todos las características de este lienzo pueden ser controladas desde la función `par`.

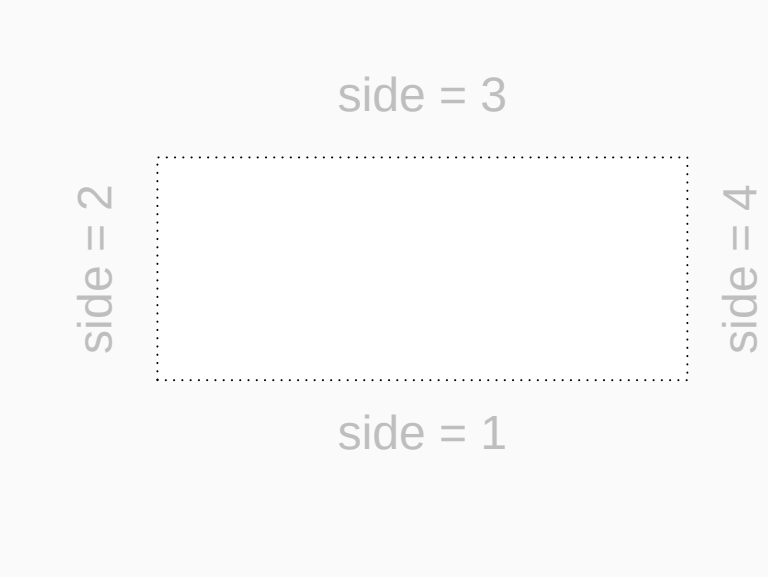

Figure 1: Lienzo con características por defecto

Por ejemplo, los márgenes pueden ser modificados utilizando la función `mar` dentro de `par`. Modifiquemos los márgenes.

```
par(bg="grey98", mar=c(3,3,3,3))
plot(1:100, type="n", axes=F, xlab = "", ylab="", bg="white")
rect(-10, -10, 120, 120, col="white")
box(lty = 3)
mtext(c("side = 1", "side = 2", "side = 3", "side = 4"),
side = c(1, 2, 3, 4), col = "grey", line = 1, cex = 1.5)
```

Podemos comparar con la anterior figura, ahora vemos que los márgenes se han hecho más pequeños. la función `mar` nos permite modificar en orden los lados 1,2,3,4. Si usted necesita modificar con un valor diferente cada lado puede hacerlo.

Puede ver todo lo que puede modificar en el lienzo usando la función `par`, ejecute en la consola **?par** y revise las funciones.

Ahora otra de las cosas que puede ser importante es poder poner en un mismo lienzo más de un gráfico. ¿Cómo puedo hacer esto?

Dividiendo el lienzo

La función `mfcol` me permite generar un lienzo con varias partes iguales, puedo decir cuántas filas y cuantas columnas. Si quiero dos gráficos uno debajo de otro lo que debo hacer es `mfcol=c(2,1)` se especifica primero las filas y luego las columnas. Si quiero en cambio un grafico a un lado de otro entonces debo poner `mfcol=c(1,2)`.

```
par(bg="grey98", mar=c(3,3,3,3), mfcol=c(1,2))
#Primer gráfico
plot(1:100, type="n", axes=F, xlab = "", ylab="", bg="white")
rect(-10, -10, 120, 120, col="white")
box(lty = 3)
mtext(c("side = 1", "side = 2", "side = 3", "side = 4"),
side = c(1, 2, 3, 4), col = "grey", line = 1, cex = 1.5)
```

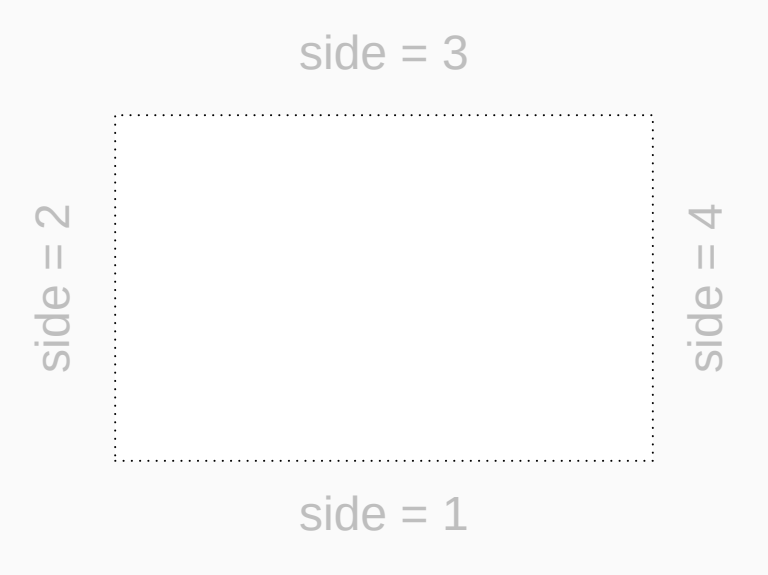

Figure 2: Reduciendo los márgenes del lienzo

```
#Segundo gráfico
plot(1:100, type="n", axes=F, xlab = "", ylab="", bg="white")
rect(-10, -10, 120, 120, col="white")
box(lty = 3)
mtext(c("side = 1", "side = 2", "side = 3", "side = 4"),
side = c(1, 2, 3, 4), col = "grey", line = 1, cex = 1.5)
```

La función `mfcol` es muy usada pero uno de los problemas que tiene es que todas las particiones tendrán el mismo tamaño, muchas veces nos interesa que un gráfico sea más grande que otro. Usaremos la función `layout` que nos permite partir el lienzo como queramos. El argumento principal de esta función es una matriz con números enteros que indican el número de partes que queremos obtener.

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

#Primer gráfico
plot(1:100, type="n", axes=F, xlab = "", ylab="", bg="white")
rect(-10, -10, 120, 120, col="white")
box(lty = 3)
#Segundo gráfico
plot(1:100, type="n", axes=F, xlab = "", ylab="", bg="white")
rect(-10, -10, 120, 120, col="white")
box(lty = 3)
#tercer gráfico
plot(1:100, type="n", axes=F, xlab = "", ylab="", bg="white")
rect(-10, -10, 120, 120, col="white")
box(lty = 3)
```

Si recordamos en lecciones anteriores hemos visto la función matrix, en esta la primera parte es el vector con los números que se escribirán dentro de la matriz (en este caso *c(1,2,3,3)*) y la segunda parte indica las filas y columnas (en este caso 2 filas y 2 columnas). Si quisiésemos que la gráfica grande este en el lado izquierdo el número 1 debería repetirse (*c(1,1,2,3)*). Inténtalo ahora tú.

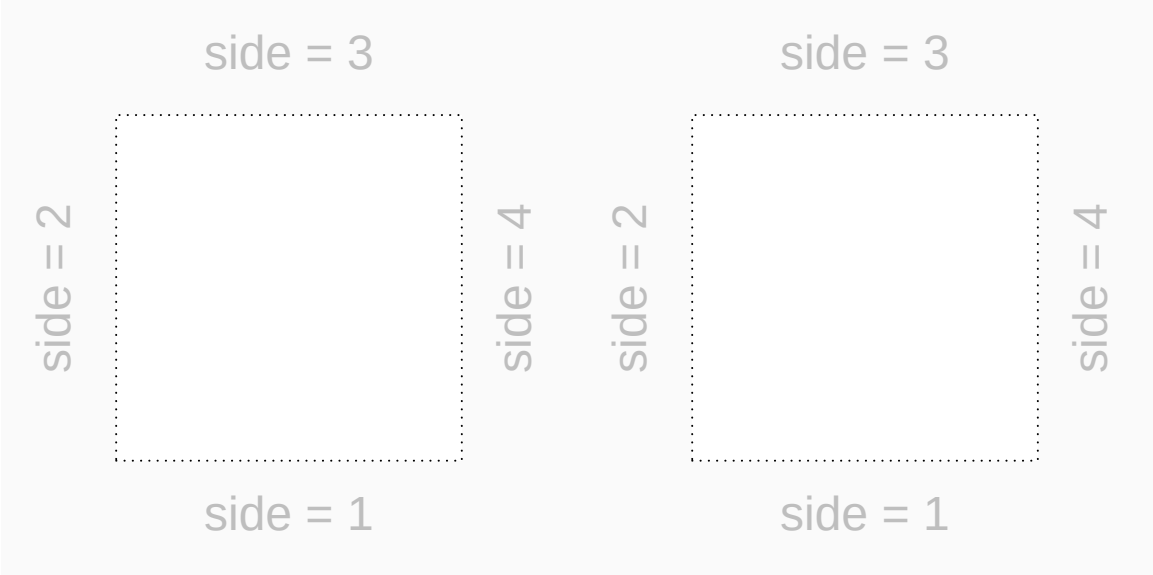

Figure 3: Lienzo con dos gráficos de igual tamaño

Figure 4: Lienzo con dos gráficos de tamaño variable

Figure 5: Lienzo con dos gráficos de tamaño variable 2

Pero qué pasa si queremos por ejemplo hacer los gráficos de diferente tamaño. Podemos utilizar la misma función pero agregamos los argumentos *widths* y *heights*, estos argumentos generan cambios relativos entre las partes, si pongo por ejemplo en widths=c(1,3) los anchos de las gráficas será la segunda columna 3 veces la primera, de forma parecida con heights condicionará los altos de las filas.

```
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")

#Primer gráfico
plot(1:100, type="n", axes=F, xlab = "", ylab="", bg="white")
rect(-10, -10, 120, 120, col="white")
box(lty = 3)
#Segundo gráfico
plot(1:100, type="n", axes=F, xlab = "", ylab="", bg="white")
rect(-10, -10, 120, 120, col="white")
box(lty = 3)
#tercer gráfico
plot(1:100, type="n", axes=F, xlab = "", ylab="", bg="white")
rect(-10, -10, 120, 120, col="white")
box(lty = 3)
```

Como vemos en este caso hemos modificado un poco el lienzo, ahora tenemos una fila continua en la parte superior y partida en la parte inferior, para lograr esta partición hemos incluido el argumento *byrow=T* esto cambia la matriz haciendo que se llene por filas y no por columnas como sucede por defecto. Además hemos incluido el ancho y alto de cada fila y columna. Finalmente, en la función `par` pusimos el argumento *oma* este argumento permite modificar los márgenes del grupo de gráficas.

> Recuerden ***mar*** modifica los márgenes de cada una de las gráficas y ***oma*** el márgen del grupo de gráficas.

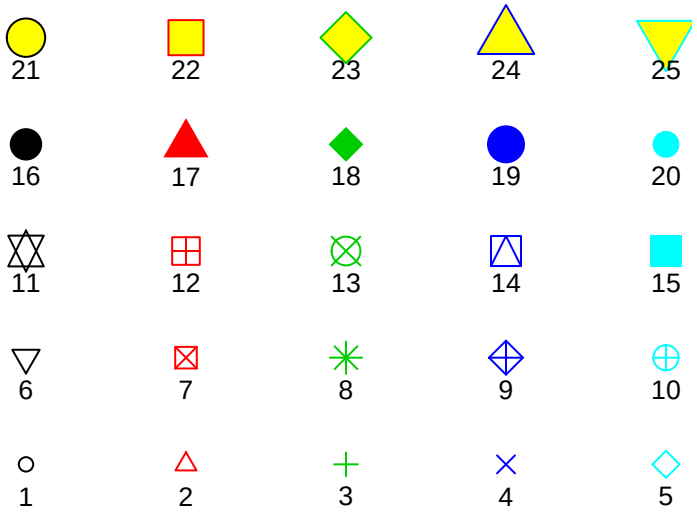

Figure 6: Tipos de símbolos gráficos desplegados

Modifica los márgenes del grupo de gráficas para que comprendas mejor lo que puede hacer este argumento.

Parámetros gráficos

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

Vamos a ver unos pocos parámetros más que nos permiten modificar por ejemplo el tipo de símbolos que graficamos o las líneas, además de los tamaños de estas dos.

Cuando realizamos un plot según sean nuestros datos, éstos serán graficados como puntos o como líneas. Podemos darle una forma, un tamaño y color a estos 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)
```

Entendamos lo que hemos hecho. Iremos por cada argumento utilizado, explicando cual es la función de cada uno de estos. ***pch***: este argumento nos permite modificar el tipo de símbolo gráfico desplegado, cada número del 1 al 25 corresponde un tipo de símbolo (el símbolo y su número correspondiente se ve en el gráfico anterior). ***col***: este argumento controla el color que tomará el símbolo, en nuestro ejemplo hemos escogido los números del 1 al 5 que corresponden a los colores; negro, rojo, verde, azul y cyan. Podemos también describir

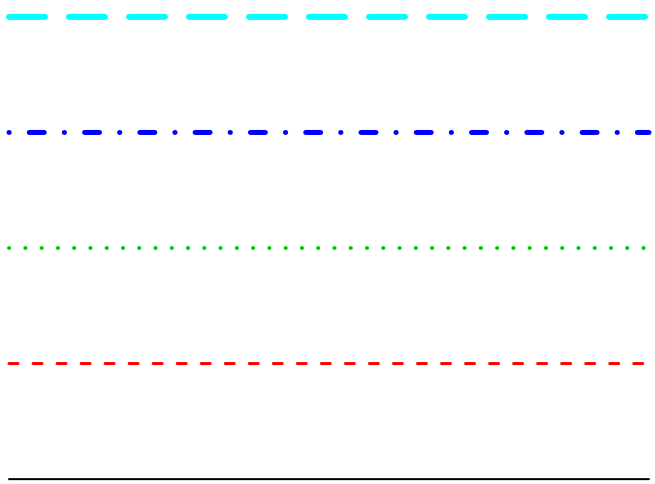

Figure 7: Tipos de líneas desplegadas en gráficos

el color que quiero poniendo el nombre en inglés, ej. "black". Para ver los colores disponibles digite colors() en la consola y ejecútelo. ***cex***: este argumento controla el tamaño de los símbolos, por defecto el tamaño es 1, podemos poner valores menores o superiores. En la figura generamos un vector entre 1 y 3 con 25 particiones, para que los símbolos se desplieguen con tamaños crecientes entre 1 y 3. ***bg***: este argumento controla el fondo de los símbolos 21 a 25, podemos utilizar cualquier color. Más adelante conoceremos el resto de argumentos.

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

```
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="")
```

En este caso el argumento ***type="l"***: define el tipo de gráfico en este caso le hemos dicho que se dibuje una línea ("l"), podemos graficar puntos (símbolos) ("p") o ambos "b" o podemos graficar líneas verticales a cada dato desde el eje central usando **type="h"**. ***lwd***: controla el tipo de línea, continua, entrecortada, puntos, puntos y líneas, etc. ***lwd***: define el tamaño de la línea, por defecto es 1. Podemos poner un valor para todas o como en el ejemplo un valor para cada línea.

Anatomía de una gráfica

Como hemos visto R es muy versátil y podemos hacer una gráfica combinada que nos presente información complementaria. Ahora pasaremos del lienzo a la gráfica, pero antes de iniciar los gráficos es necesario conocer cuáles son las partes de una gráfica.

Utilizaremos datos de R para hacer una gráfica y ver sus partes.

```
data(cars)
tiff(filename = "antomia.tiff", width = 1600, height = 1300,
     pointsize = 13, units = "px", res = 300)
par(mar=c(4,3,3,4), mgp=c(1.5,0.4,0), tck=-0.02)
```

```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("Título",side = 3,at = 2, font=2, line=1.3, cex=0.9, col="red")
mtext("Eje x",side = 1,at = 27, font=2, line=0.4, cex=0.8, col="blue")
mtext("Etiqueta del eje x",side = 1,at = 25, font=2, line=1.5, cex=0.8, col="red")
mtext("Eje y",side = 1,at = 2, font=2, line=0.3, cex=0.8, col="blue", las=2)
mtext("Etiqueta del eje y",side = 1,at = 0.5, font=2, line=-3.5, cex=0.8, col="red", las=2)
text(10,80, "Datos", col="red")
dev.off()
```

```
## pdf
##   2
```

Los ejes

Los ejes definen el espacio o rango de una gráfica y lo más común es que una gráfica tenga dos ejes, los cuales denominamos X e Y. El eje X se ubica por convenio en la horizontal. En el eje x ponemos las variables independientes o explicativas, mientras que en el eje y colocamos las variables de respuesta. En el caso del gráfico anterior la velocidad de frenado influye en la distancia de frenado de los vehículos.

Cada eje nos permite ver la variación de nuestros datos, vemos que la velocidad de los vehículos varía entre 3 y 25 mph (parece raro no!, los datos son de 1920), y la distancia de frenado esta entre 0 y 120 ...., *se dieron cuenta* hemos encontrado un error en la etiqueta del gráfico, no sabemos la distancia en que esta medido. Bueno se los cuento la distancia esta medida en pies. Como podemos ver en el gráfico R por defecto ajusta los ejes a los datos, siempre es bueno no tener espacio en blanco, sin embargo, usted podría querer marcar un límite de los ejes para hacer esto podemos utilizar el argumento **xlim** o **ylim** para modificar el eje x o y respectivamente. Si quisiéramos incrementar el eje "Y" hasta 150 podríamos utilizar el argumento de la siguiente forma: **ylim=c(0,150)**.

Las etiquetas de los ejes

Como nos dimos cuenta hace un momento tener unas etiquetas correctas es fundamental puesto que estas nos permiten comprender nuestros ejes. !Cuidado¡ debemos pensarlas bien, ya que tampoco se vale tener unas etiquetas muy grandes. Recuerda la información debe ser clara y concisa. La mayor parte de ocasiones las etiquetas de los ejes deben contener la unidad de medida en la cual se miden las variables.

> Aprovecho para darte un consejo, siempre es mejor mantener los ejes con valores pequeños. Si tenemos unos datos en cientos de miles, es mejor representar los ejes en cientos y en la etiqueta poner el dato en miles. Veamos un ejemplo: si tengo una etiqueta de "número de células cancerígenas" con unidades del eje entre 1000 y 100000, es mejor cambiar a una etiqueta "Miles de células cancerígenas" con unidades de entre 1 y 100.

El título

El título es una parte importante del gráfico, al igual que las etiquetas debemos intentar que el título sea corto y que rescate lo más importante de la gráfica. Muchas veces se omite el título del gráfico cuando utilizamos una etiqueta de la figura. Sin embargo, recuerde que si no tenemos la etiqueta debemos colocar un título.

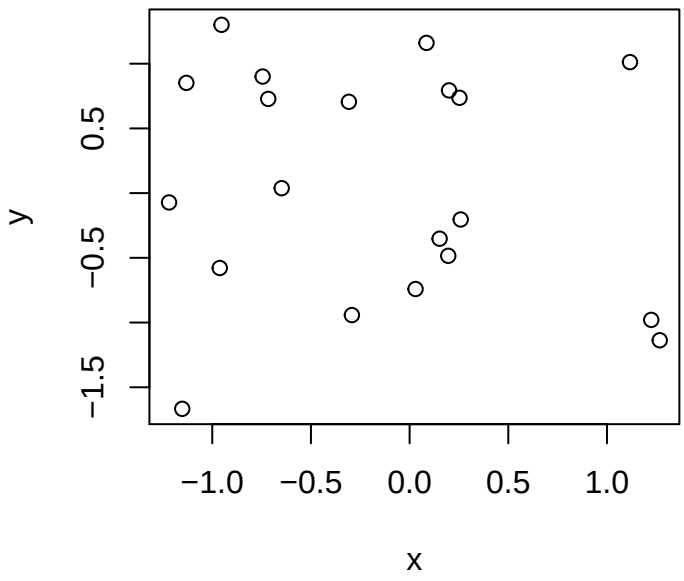

Figure 8: Uso de plot, el primer paso

Ejercicio 1

Bueno vamos a parar un poco y ver lo que otra gente ha hecho con gráficos. Este ejercicio es sencillo y espero divertido.

Piense en una temática específica que le interese, ahora va a buscar 6 gráficos de esa temática, 3 gráficos deben ser buenos gráficos y mire tres más que usted considere que no son buenos gráficos. Coloque los gráficos en una hoja de word y describa porque considera que los gráficos son buenos o malos.

Una gráfica con R

Vamos mostrar algunas de las funcionalidades de R para hacer gráficos y dar formato general. Vamos a trabajar con un gráfico bivariado para lo cual vamos a generar dos series de datos al azar.

El primer paso hacer el gráfico para esto usamos la función `plot` y agregamos las dos series de datos que queremos graficar.

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

#Graficamos
plot(x,y)
```

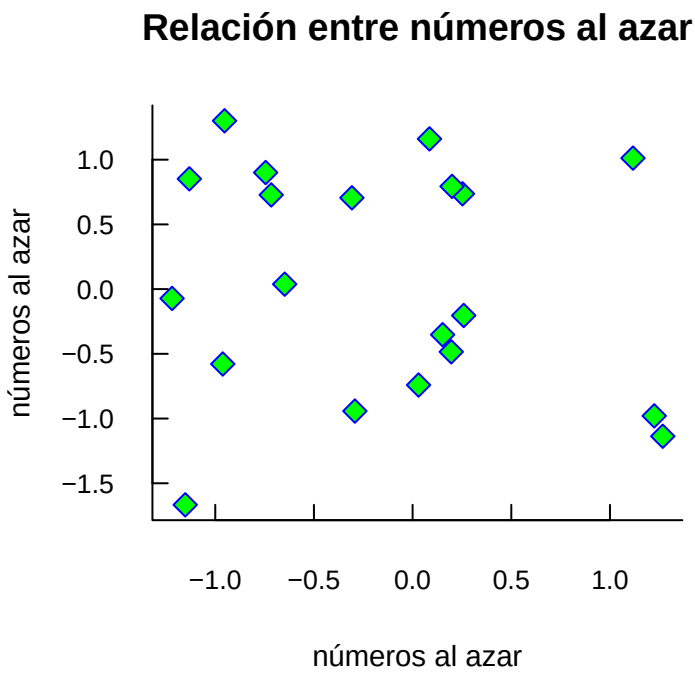

Figure 9: Personalizando el grafico en R (1)

Como vemos el gráfico por defecto no está del todo mal, ha generado unos ejes ajustados a los datos, ha dado unas etiquetas a cada eje y puesto que son datos de dos variables ha elegido puntos para representar los gráficos. Pero podríamos mejorar la presentación de este gráfico. Por ahora no hemos hablado de los datos (lo más importante), sino que nos centraremos en el formato.

Recuerde que hay algunos principios que deberíamos tenerlos presente. Lo primero es tener un título, y unas etiquetas explicativas para cada eje. Además vamos a cambiar a unos símbolos que llamen más la atención.

```
plot(x, y, xlab="números al azar", ylab="números al azar", pch=23, col="blue", bg="green", bty="l", tcl=
     cex=1.3, cex.axis=0.8, cex.lab=0.9)
```

Parece que ha mejorado, pueden intentar cambiando el símbolo utilizado, seleccionen un símbolo en base de la figura de símbolos. Cambie el color, utilice `colors()` para ver que colores hay disponibles.

Que argumentos hemos utilizado. Con **xlab** y **ylab** podemos indicar a R lo que queremos que se escriba como etiqueta del eje x y el y respectivamente. El argumento **pch** y **col** ya los hemos explicado antes es para cambiar el símbolo y el color. El argumento **bty** permite controlar el tipo de caja del área del gráfico, "l" dibuja el primer eje y el segundo eje, nos da como resultado una L, podría poner "7" y se grafican los ejes 3 y 4, "c" los ejes 1, 2 y 3. El argumento **tcl** modifica las marcas de graduación del gráfico, por defecto es **-0.5**, nosotros hemos cambiado a **0.4** y por eso se han dibujado dentro de la caja. El argumento **main** indica el título. **las** indica la orientación de los ejes, por defecto es "las=0" que indica que el eje x se escribe horizontalmente y el eje y se escribe verticalmente. Podemos cambiar a "las=1" la opción que elegimos para que el eje "x" y "y" se escriban horizontalmente. Finalmente, "las=2" me permite poner el eje x verticalmente y el eje y horizontalmente. Como vimos antes el argumento **cex** permite controlar el tamaño en general, en este caso cex solo controla el tamaño del símbolo, "cex.axis" y "cex.lab" controla el tamaño del eje y el tamaño de la etiqueta del eje. Podemos utilizar "cex.main" para controlar el tamaño del título.

Hemos mejorado pero aún me gustaría modificar algunas otras cosas como el fondo del lienzo, la distancia a

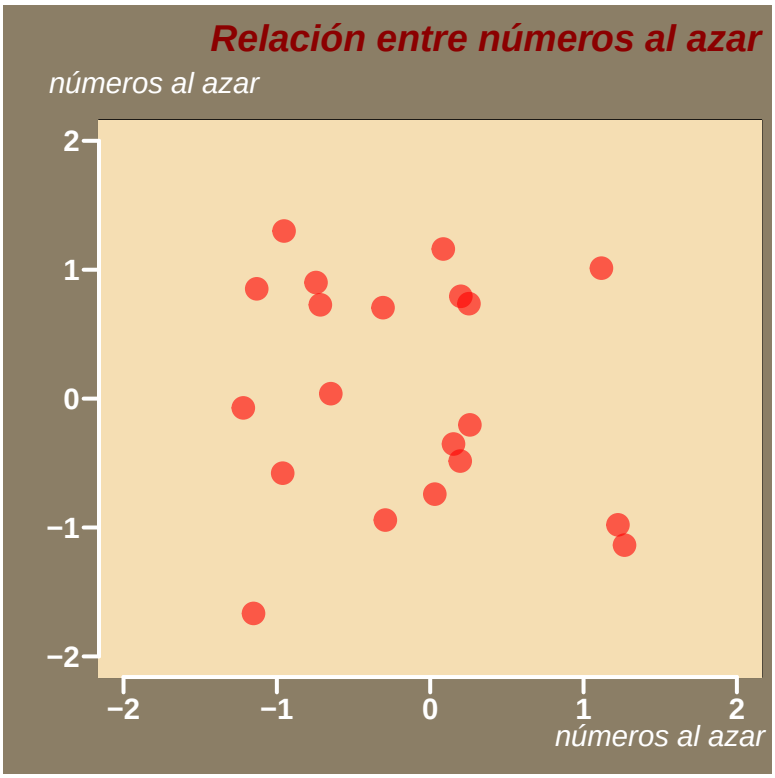

Figure 10: Personalizando el grafico en R (2)

la que se escriben el título y las etiquetas de los ejes. Vamos a ver cómo cambiar este formato.

```
par(bg="wheat4", mar=c(2.5, 2.5,3, 0.25))#cambiamos el fondo y los márgenes

plot(x, y, type="n", xlab="", ylab="", xlim=c(-2, 2),
     ylim=c(-2, 2), xaxt="n", yaxt="n") #ejecutamos el plot con type="n",
                                        #se despliega unicamente la caja (box)
rect(-3, -3, 3, 3, col="wheat") #una caja con de otro color
points(x, y, pch=19, col=rgb(1,0,0,0.6), cex=1.5) #agregamos los puntos
axis(side=1, c(-2,-1,0,1,2), tcl=-0.4,
     labels=FALSE, col="white", lwd=2) #agregamos el primer eje, de color blanco
axis(side=2, -2:2, tcl=-0.4, labels=FALSE, col="white", lwd=2)
title("Relación entre números al azar", font.main=4, adj=1,
      cex.main=1.2, line=1.8, col.main="darkred") #Agregamos el título
mtext("números al azar", side=1, line=1, at=1.5,
      cex=0.9, font=3, col="white")#Agregamos etiqueta del eje 1
mtext("números al azar", line=0.5, at=-1.8,
      cex=0.9, font=3, col="white")#Agregamos etiqueta del eje 2
mtext(-2:2, side=1, las=1, at=-2:2, line=0.3,
      col="white", cex=0.9, font=2) #Incluimos las leyendas del eje 1
mtext(-2:2, side=2, las=1, at=-2:2, line=0.5,
      col="white", cex=0.9, font=2) #Incluimos las leyendas del eje 2
```

Intentaré explicar rápidamente lo que hemos hecho. - Lo primero es que graficamos con argumentos de tipo (type) y ejes (xaxt, yaxt) nulo (“n”), por lo que no se dibujan ni ejes ni puntos. Además, las etiquetas de los ejes sin datos (xlab=“”, ylab=“”) por lo que R no escribe etiquetas.

- Con la función `rect` graficamos un cuadrado mayor al límite del gráfico de -3 a 3 y le ponemos un color distinto al fondo.
- Con la función `points` agregamos los datos con unos símbolos determinados y les ponemos un color. Usamos la función `rgb` que nos permite dar un color y hacerlo semitransparente.
- Con la función `axis` incluimos el eje 1 y 2 (side=1 o 2), le damos un vector que usará para poner las marcas, el tamaño de las marcas lo controlamos con tcl (tcl=-0.4). Le damos color (col), le decimos que no incluya las leyendas (labels=FALSE) y controlamos el ancho de la línea del eje (lwd=2).
- Con la función `title` podemos incluir el título y controlar algunos parámetros. El argumento "font.main" permite modificar el texto con cursivas y negritas como en el gráfico (font.main=4), cursivas (font.main=2), negritas (font.main=2) o sin cursivas y sin negritas (font.main=4). El argumento adj=1 justifica el texto a la derecha, podemos cambiarlo al centro con adj=0.5, o a la izquierda con adj=0. El argumento line=1.8 permite alejar o acercar el título del gráfico.

Me parece que quedo muy bien el gráfico que opinan ustedes?. Lo podrían hacer mejor, estoy seguro de que si, inténtalo.

Ejercicio 2

Modifica los argumentos de esta gráfica, cambia los colores los símbolos y todo lo que quieras. Intenta hacer un gráfico que se acople a lo que necesitas y te gusta.

Gráficos de una variable

Bueno creo que hemos logrado hacer cosas interesantes pero solo hemos jugado, como les decía lo más importante en un gráfico son los datos. Hasta ahora no nos habíamos preocupado de ello, pero es hora de hacerlo.

```
library(readxl)
ame <- read_excel("AMEBIASIS_LOJA.xlsx",sheet = 1, na = "NA")#cargamos los datos
pr.edad <- tapply(ame$`Edad en años`, ame$Parroquia, mean)
                                        # obtenemos la media de edad por parroquia
```

Una vez que tenemos los datos de la media de edad con amebiasis es posible que nos interese ver los datos en una gráfica.

```
plot(pr.edad, ylab="Edad Promedio", xaxt="n", xlab="", pch=19)
```

Le hemos pedido que no grafique el eje x, ya que realmente el eje x no es ninguna variable, lo que estamos graficando es una sola variable, por lo que lo lógico es que este eje no se grafique. Esta gráfica nos permite ver la dispersión de los datos, ya podemos ver que la edad más recurrente en la que se enferman de amebiasis es entre 20 y 30 años. Sin embargo, una gráfica mejor para resumir una sola variable cuantitativa como la media de edad sería un diagrama de caja, también conocido como diagrama de caja y bigote, o boxplot en inglés.

```
boxplot(pr.edad)
```

Ahora es mucho más claro que la mediana de nuestros datos están en torno a los 30 años. Además vemos que algunas parroquias muestran valores raros superiores a los 50 años. Una buena forma de presentar esta información es mezclar este boxplot con los datos brutos. Veamos lo que podemos hacer.

```
boxplot(pr.edad)
stripchart(pr.edad, method = "jitter", jitter = 0.1, add =
TRUE, vertical = TRUE, pch=19)
```

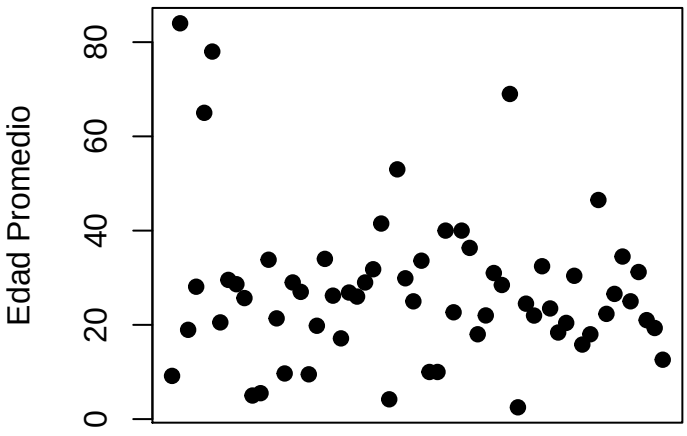

Figure 11: Ejemplo de gráfico con una sola variable

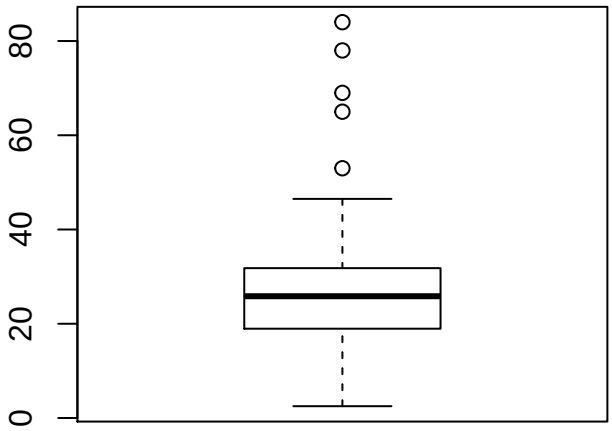

Figure 12: Ejemplo de boxplot con una sola variable (1)

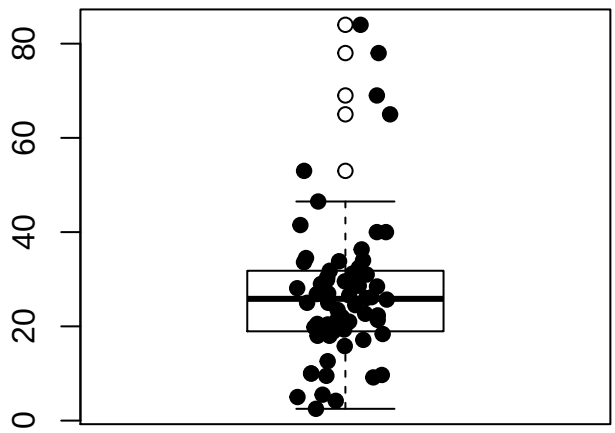

Figure 13: Ejemplo de boxplot con una sola variable (1)

Bueno que les parece, ¿bonito no? Aunque hay muchos elementos que incluir aún para que esto pueda llamarse gráfico de verdad.

Otra interesante opción es realizar un histograma con esta variable. El histograma nos muestra la frecuencia de los datos a lo largo de rangos equitativos de la variable. Al igual que el boxplot nos ayuda ver la distribución de los datos.

```
hist(pr.edad)
```

Ejercicio 3

Nos interesa describir la distribución de edad de las personas que han padecido Amebiasis.

Incluya los elementos que considere necesarios y póngale el formato de color y ejes que le parezca al boxplot y al histograma del promedio de edad por parroquia de las personas con amebiasis. Debe partir el lienzo en dos partes, podrían ser desiguales para poner en el mismo lienzo las dos gráficas. Espero tener un gráfico para enmarcar, seguro lo podrá hacer.

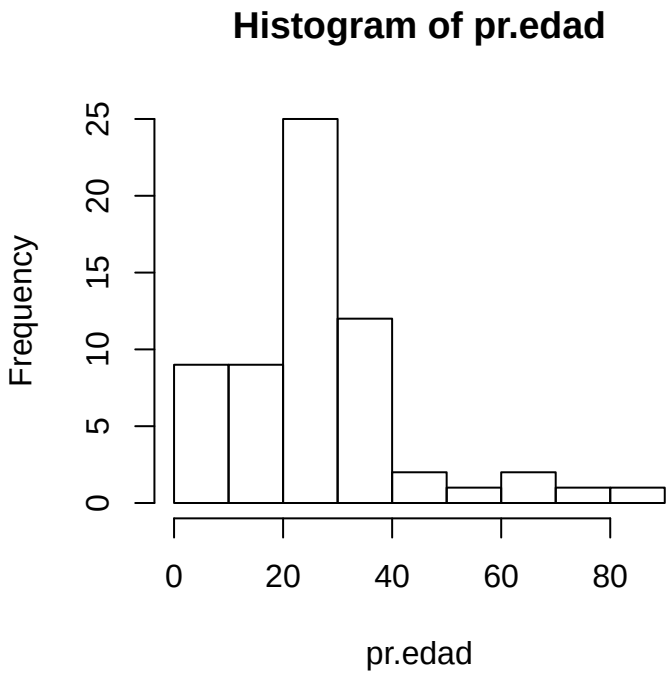

Figure 14: Ejemplo de histograma