Graficando con R

El comienzo

Carlos Iván Espinosa

Octubre de 2016

Contents

Introducción

La representación gráfica de los datos es una actividad bastante común en todas partes vemos graficos 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 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 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.

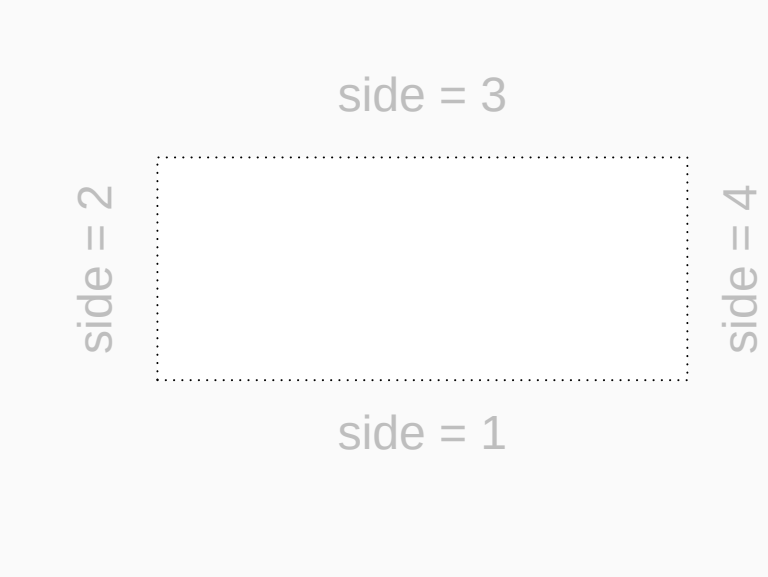

Figure 1: Lienzo con características por defecto

```
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`. 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.

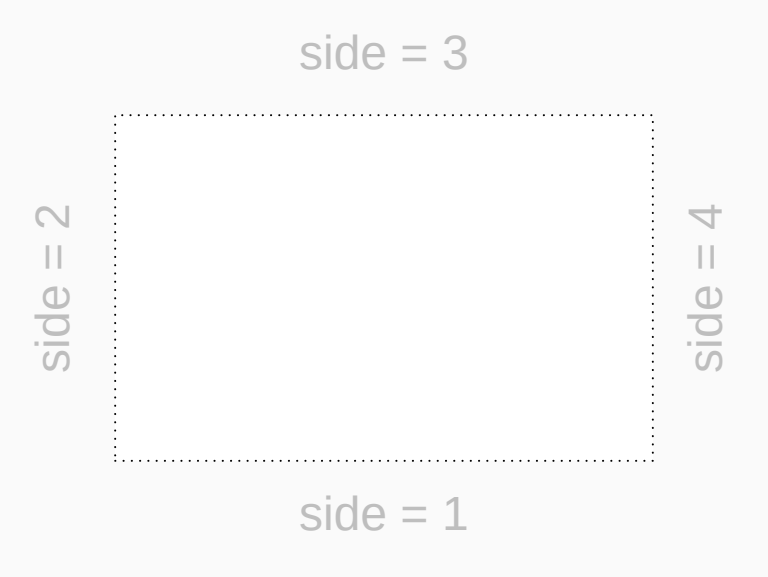

Figure 2: Reduciendo los márgenes del lienzo

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)
#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")
```

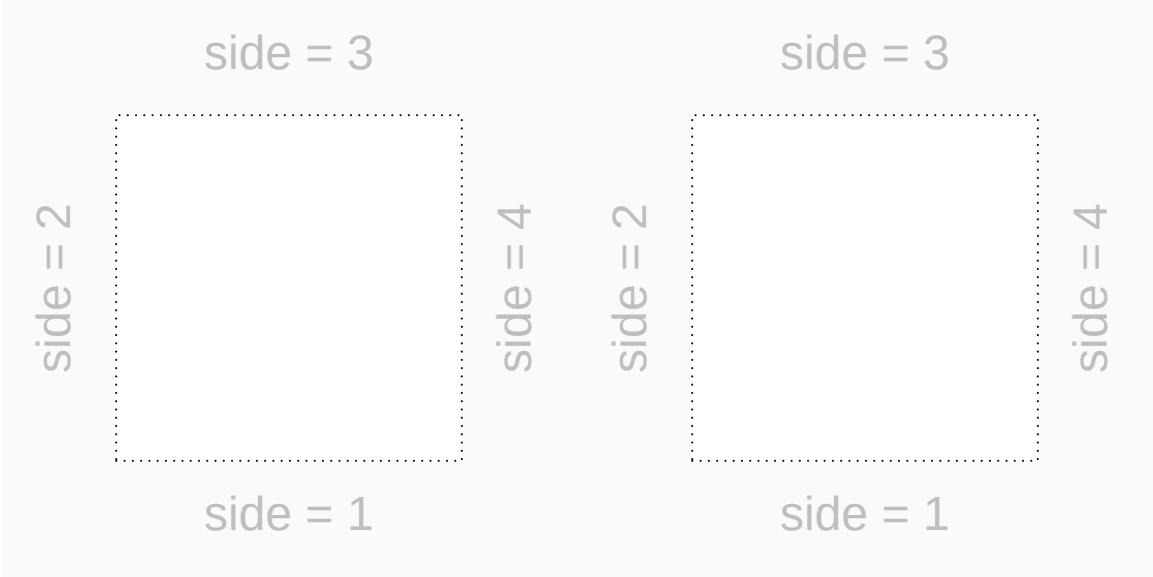

Figure 3: Lienzo con dos gráficos de igual tamaño

```
#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.

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
```

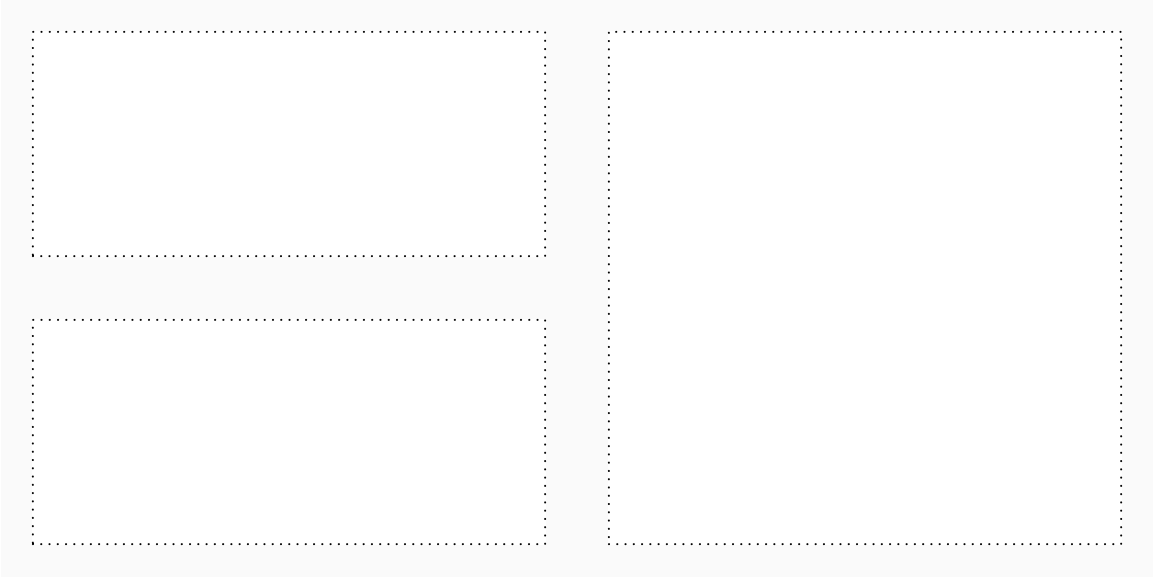

Figure 4: Lienzo con dos gráficos de tamaño variable

```
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.

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 que los hemos manipulado en la función `par`, 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.

Figure 5: Lienzo con dos gráficos de tamaño variable 2

```
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 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". ***lwd***: controla el tipo de línea, continua,

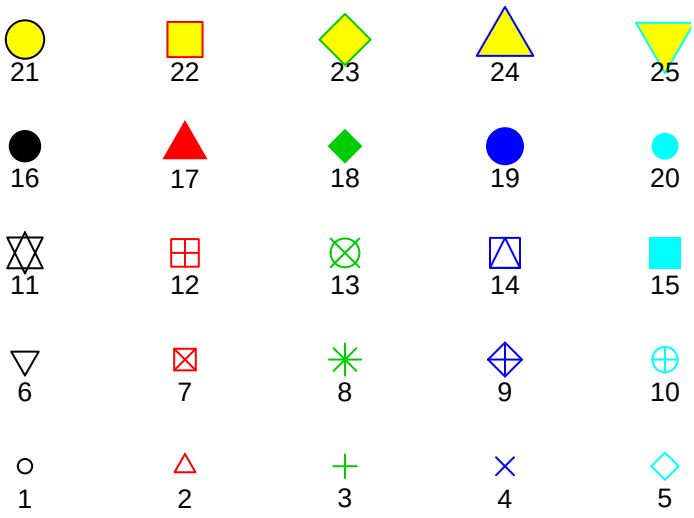

Figure 6: Tipos de símbolos gráficos desplegados

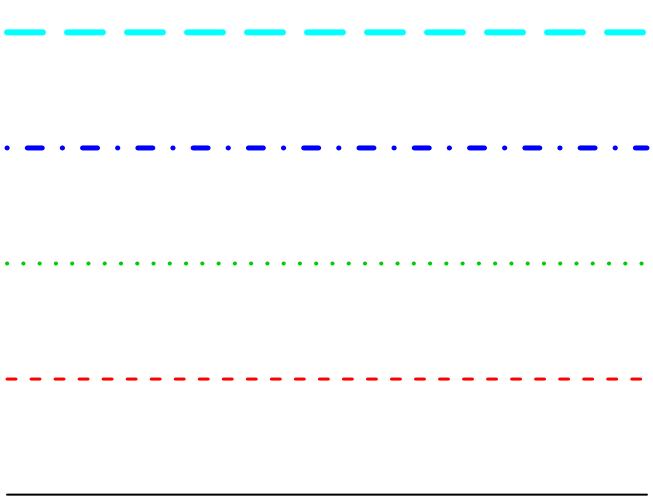

Figure 7: Tipos de líneas desplegadas en gráficos

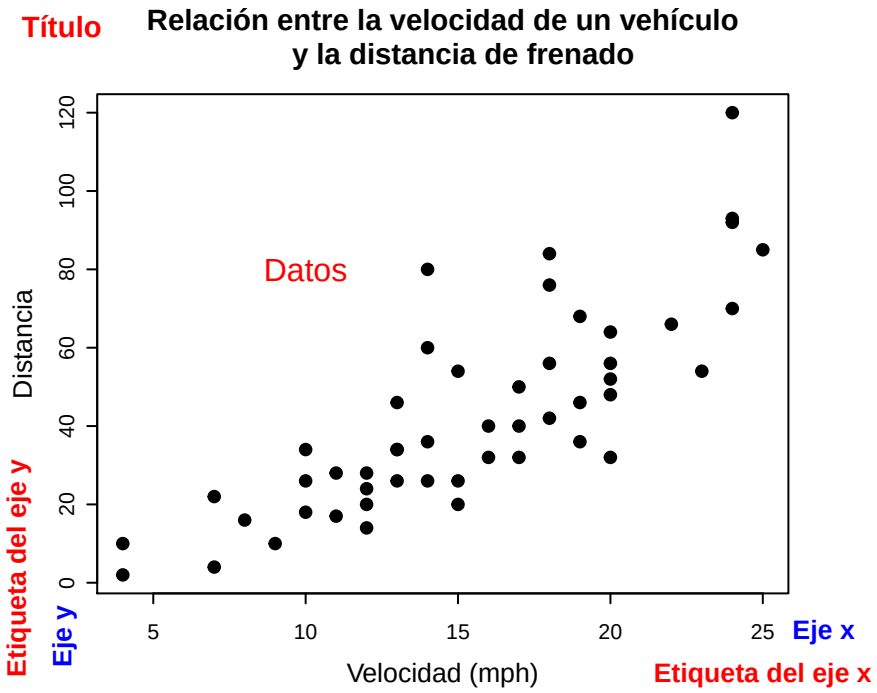

Figure 8: Anatomía de un gráfico

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)
par(mar=c(4,3,3,4), mgp=c(1.5,0.4,0), tck=-0.02)

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")
```

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 podriamos 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.

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

Gráficos de una variable