Modificar el tipo de línea (`lty`)

En cualquier figura que contenga una línea en R, ésta normalmente se representa como una línea continua. ¿Qué pasa si queremos que sea discontinua, rayada, punteada o una combinanción de las anteriores? Para eso está el **atributo `lty`**, que significa literalmente *Line type*. Los valores que acepta son numéricos y a continuación os voy a mostrar qué vale cada uno (por ejemplo, `lty = 2`) (Figura 1)

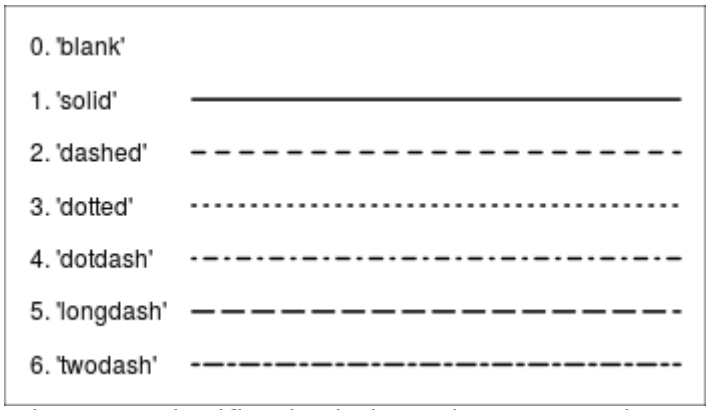

Figura 1: Significado de los primeros 6 valores de lty.

Ejecutando en R Commander cada una de estas líneas de código de modo independiente, obtenemos que lo se observa en la Figura 2. Si no está el atributo, de modo predeterminado se ejecuta `lty=1`.

```
densityPlot( ~ MD.mm., data=Lumley, bw="SJ", adjust=1, kernel="gaussian")
densityPlot( ~ MD.mm., data=Lumley, bw="SJ", adjust=1, kernel="gaussian",
lty=1)
densityPlot( ~ MD.mm., data=Lumley, bw="SJ", adjust=1, kernel="gaussian",
lty=2)
densityPlot( ~ MD.mm., data=Lumley, bw="SJ", adjust=1, kernel="gaussian",
lty=3)
densityPlot( ~ MD.mm., data=Lumley, bw="SJ", adjust=1, kernel="gaussian",
lty=4)
densityPlot( ~ MD.mm., data=Lumley, bw="SJ", adjust=1, kernel="gaussian",
lty=5)
densityPlot( ~ MD.mm., data=Lumley, bw="SJ", adjust=1, kernel="gaussian",
lty=6)
```

En el Vídeo A2.1 os enseño cómo introducir los atributos mencionados en este anexo en R Commander.

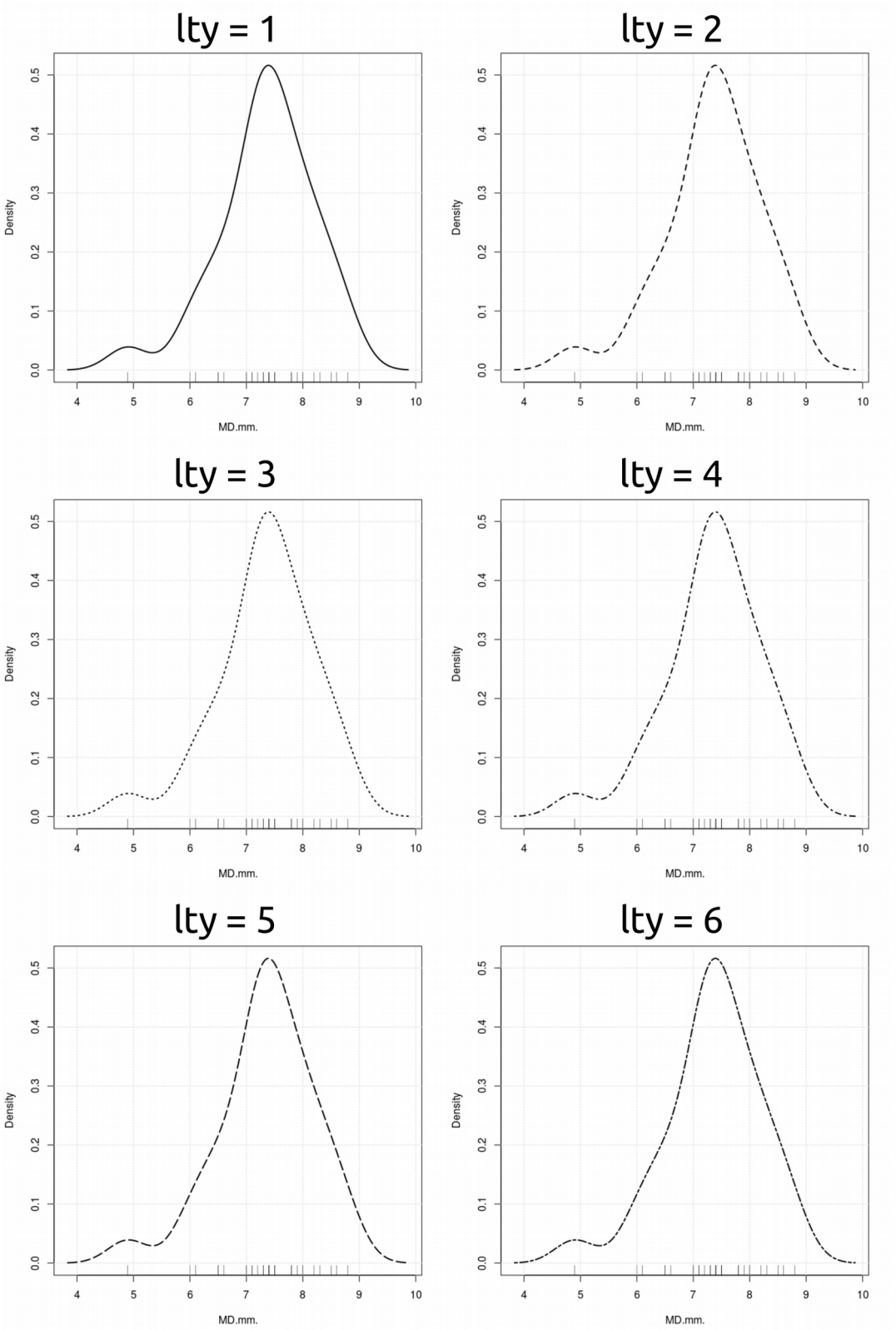

Figura 2: Ejemplos de diferentes grosores de línea utilizando el atributo `lty`.

Modificar el grosor de línea (**lwd**)

Si queremos modificar el grosor de la línea de la figura tenemos que añadir el **atributo `lwd`**, que significa literalmente *Line width* (grosor de línea). Este atributo admite valores numéricos. Ejecutando de modo independiente los siguientes códigos se obtienen las imágenes de la Figura 3. Si no aparece el atributo escrito, de modo predeterminado se ejecuta `lwd=1`.

```
densityPlot( ~ MD.mm., data=Lumley, bw="SJ", adjust=1, kernel="gaussian")
densityPlot( ~ MD.mm., data=Lumley, bw="SJ", adjust=1, kernel="gaussian",
lwd=1)
densityPlot( ~ MD.mm., data=Lumley, bw="SJ", adjust=1, kernel="gaussian",
lwd=5)
densityPlot( ~ MD.mm., data=Lumley, bw="SJ", adjust=1, kernel="gaussian",
lwd=10)
densityPlot( ~ MD.mm., data=Lumley, bw="SJ", adjust=1, kernel="gaussian",
lwd=20)
```

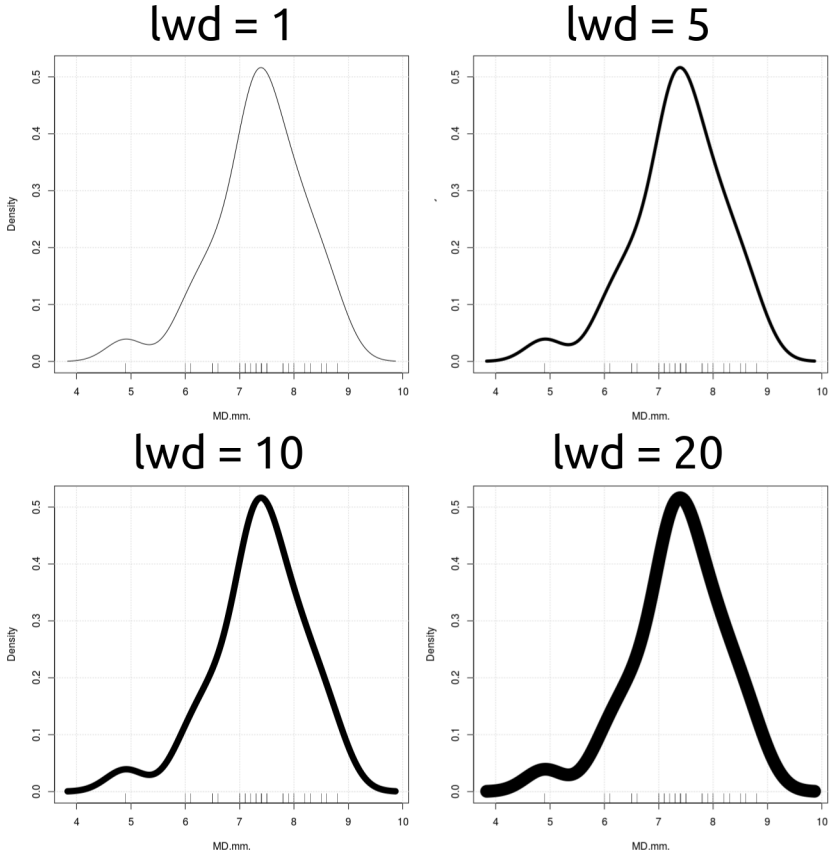

Figura 3: Ejemplos de diferentes grosores de línea utilizando el atributo `lwd`.

Modificar los intervalos numéricos de los ejes *X* e *Y* (**xlim** e **ylim**)

Aunque de modo predeterminado R y R Commander adapta automáticamente los intervalos de los ejes *X* e *Y* a los datos que esté representando gráficamente, en ocasiones puede interesarnos cambiar los intervalos y adaptarlos a nuestras necesidades. En R es muy sencillo añadiendo el atributo `xlim` para el eje *X* e `ylim` para el eje *Y*. En el siguiente ejemplo vamos a representar el eje *X* comprendido entre 0 y 15, mientras que el eje *Y* estará comprendido entre los valores 0 y 1 (Figura 4).

```
densityPlot( ~ MD.mm., data=Lumley, bw="SJ", adjust=1, kernel="gaussian",
xlim=c(0,15), ylim=c(0,1))
```

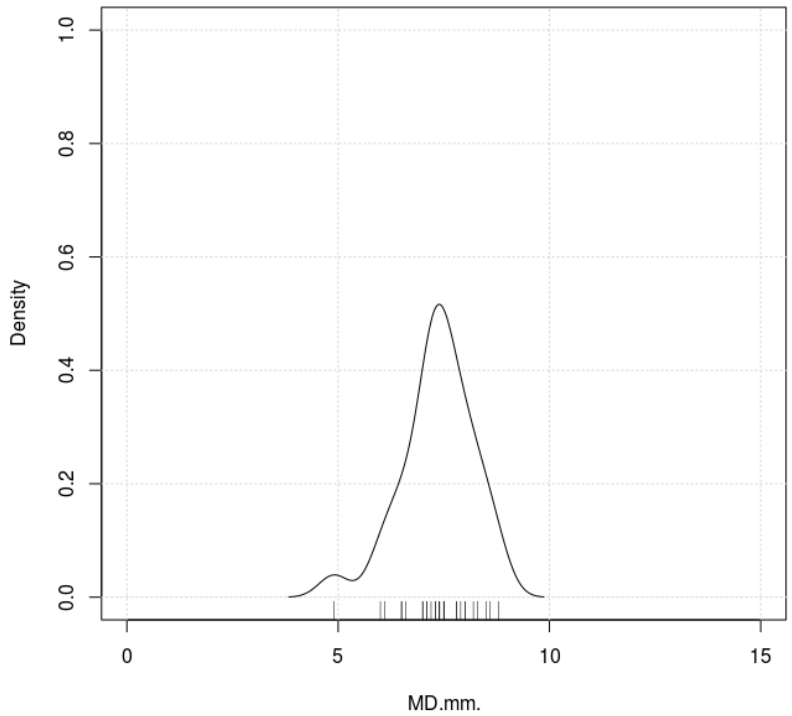

Figura 4: Personalización de los intervalos de los ejes X e Y con los atributos xlim e ylim.

Modificar los colores

Reflejar en cualquier figura un color u otro es relativamente sencillo en R, siempre y cuando te expliquen cómo implementarlo. Por descontado que la introducción de colores hace que cualquier figura sea visualmente mucho más atractiva siempre y cuando esos colores respondan a algo en particular.

Pues bien, en R podemos introducir colores de **dos maneras diferentes**: (1) **introduciendo un nombre** preestablecido o (2) **introduciendo el código hexadecimal** propio del color en cuestión.

Aunque el atributo es ligeramente variable según comandos, pongamos como ejemplo el atributo `col = “nombre_color"` (observar el gráfico inferior derecho de la Figura 11 como ejemplo).

Colores con el nombre del color preestablecido

Colores concretos están definido por un nombre. Veamos la Figura 5 con unos ejemplos.

color	name	color	name
	white		burlywood4
	aliceblue		cadetblue
	antiquewhite		cadetblue1
	antiquewhite1		cadetblue2
	antiquewhite2		cadetblue3
	antiquewhite3		cadetblue4
	antiquewhite4		chartreuse
	aquamarine		chartreuse1
	aquamarine1		chartreuse2
	aquamarine2		chartreuse3
	aquamarine3		chartreuse4
	aquamarine4		chocolate
	azure		chocolate1
	azure1		chocolate2
	azure2		chocolate3
	azure3		chocolate4

Figura 5: Ejemplo de colores con nombres preestablecidos.

Ejemplos de código serían los siguientes, con el atributo `col` incorporado. Atención a las comillas, ¡que son obligatorias!

```
plot(x, y, col = "blue")
plot(x, y, col = “chartreuse")
plot(x, y, col = “lightgoldenrodyellow")
```

Una variante de estos nombres, son códigos numéricos preestablecidos, como se en la Figura 6. Su empleo es idéntico al anterior, teniendo cuidado de las comillas. Veamos estos ejemplos:

```
plot(x, y, col = "84")
plot(x, y, col = “515")
plot(x, y, col = “294")
```

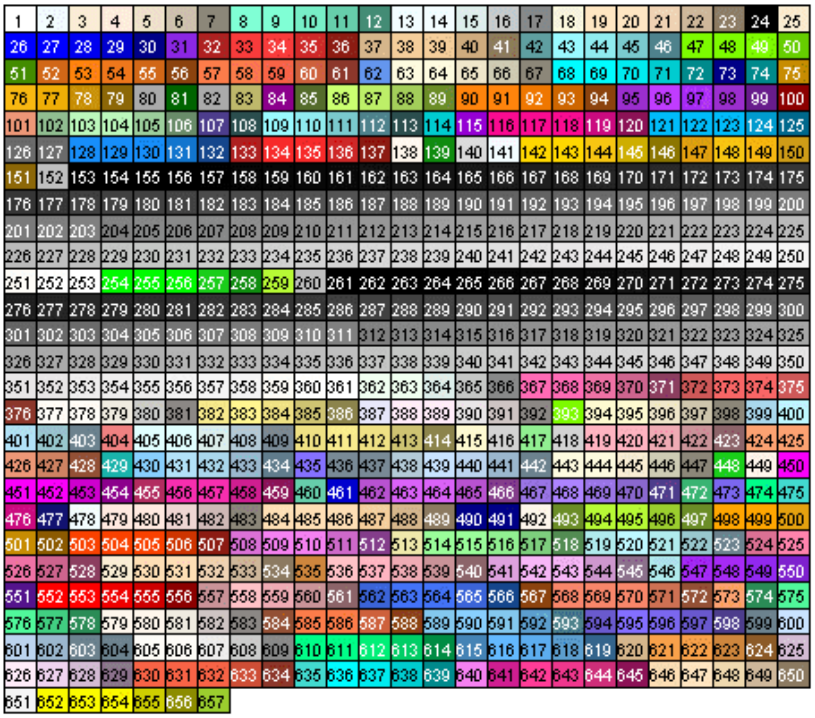

Figura 6: Códigos numéricos preestablecidos que hacen referencia a colores específicos.

Colores con códigos hexadecimales

Primero, no hay que asustarse con *códigos hexadecimales*. Viendo la Figura 7, que puede ser obtenida de cualquier software de edición de fotografías o imágenes (en el ejemplo he usado GIMP 2.8), podemos obtener la **paleta de colores**. Si nos fijamos

atentamente, hay una sección que pone *HTML notation*. Ese código (hexadecimal, porque tiene 6 caracteres) es el que estamos buscando, correspondiéndose en nuestro ejemplo con 51da67. He de decir que el título de *HTML notation* puede variar entre el software que se utilice, no siendo así para el código del color, que es universal. Considero importante también mencionar que los caracteres del código **pueden escribirse en mayúscula o minúscula**, no alterando así el color final, por lo que sería lo mismo escribir 51da67 que 51DA67.

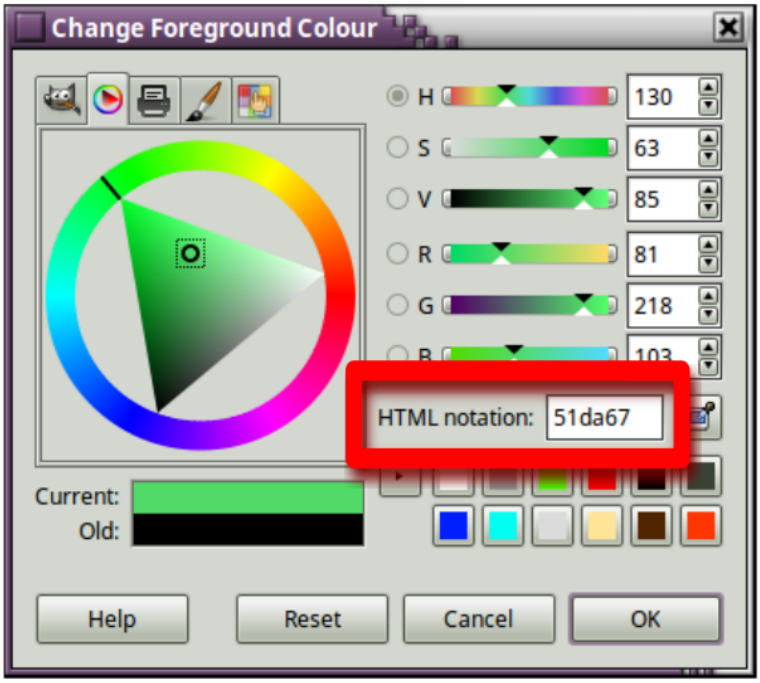

Figura 7: Dónde localizar el código hexadecimal de cualquier color. En este caso he usado GIMP 2.8 para sacar la paleta de colores.

Veamos un par de ejemplos sobre el uso de estos códigos hexadecimales sobre el código de R. Es importante mencionar que las comillas se mantienen y es obligatorio introducir al principio el símbolo de almohadilla (#):

```
plot(x, y, col = "#51DA67")
plot(x, y, col = “#cb3622")
plot(x, y, col = “#596CCC")
```

Introduciendo transparencias en los colores

Es un caso particular, pero supongamos que tenemos varios puntos en un diagrama de dispersión que se solapan en gran parte y no vemos todos los puntos. **Podemos añadir transparencias a los códigos hexadecimales**. Esto se consigue **añadiendo el porcentaje de opacidad que queremos justo después del código hexadecimal**. El

intervalo que se maneja es entre 00 y 99. Cuanto mayor sea este valor, mayor será la opacidad y menor la transparencia.

Veamos el siguiente código, donde he marcado una opacidad del 25% (o una transparencia del 75%), resaltándola en negrita justo donde hay que implementarla. Algunos atributos de este código ya los hemos visto (`xlim` e `ylim`) y otros los veremos a continuación (`cex` y `pch`). En la Figura 8 se puede observar el resultado con varios porcentajes.

```
plot(Dataset$x, Dataset$y, cex = 10, pch = 19, xlim = c(-5,8), ylim =
c(0,15), col = "#51da6725")
```

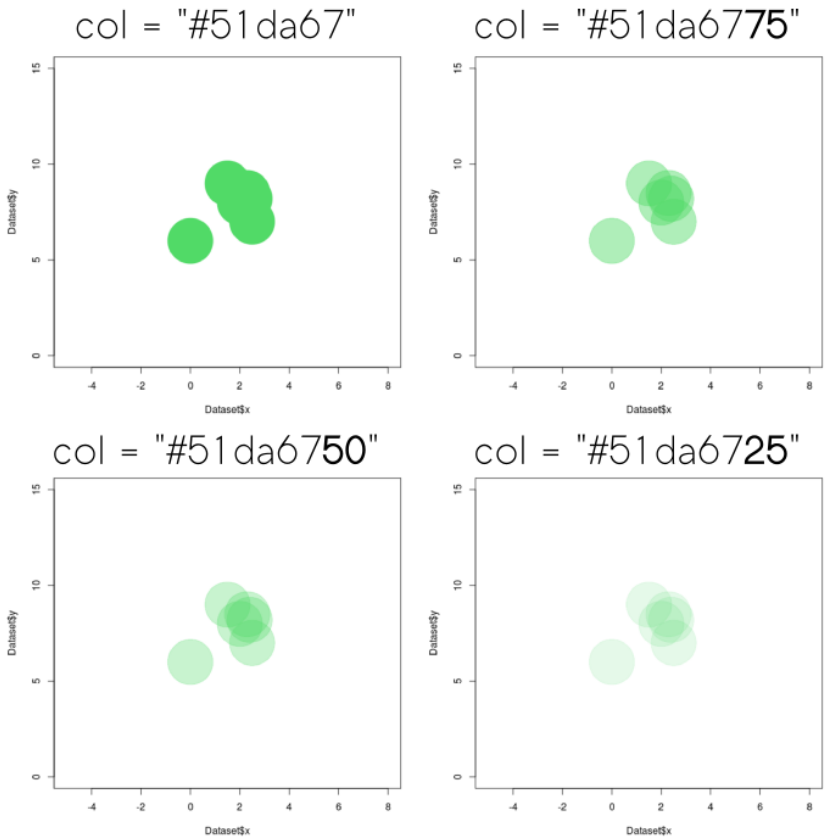

Figura 8: Ejemplos de diferentes transparencias sobre unos mismos datos (100, 75, 50 y 25).

Modificar los títulos de los ejes, del título principal y del subtítulo

En R Commander es muy sencillo personalizar las leyendas de los ejes X e Y, así como del título principal de la figura. Prácticamente la totalidad de las figuras que se pueden hacer en R Commander usando la interfaz gráfica trae por defecto en la pestaña de **Opciones** (*Options*) la posibilidad de personalizar estos títulos y leyendas (*Plot Labels*, *x-axis label*, *y-axis label* y *Graph title*) (Figura 9).

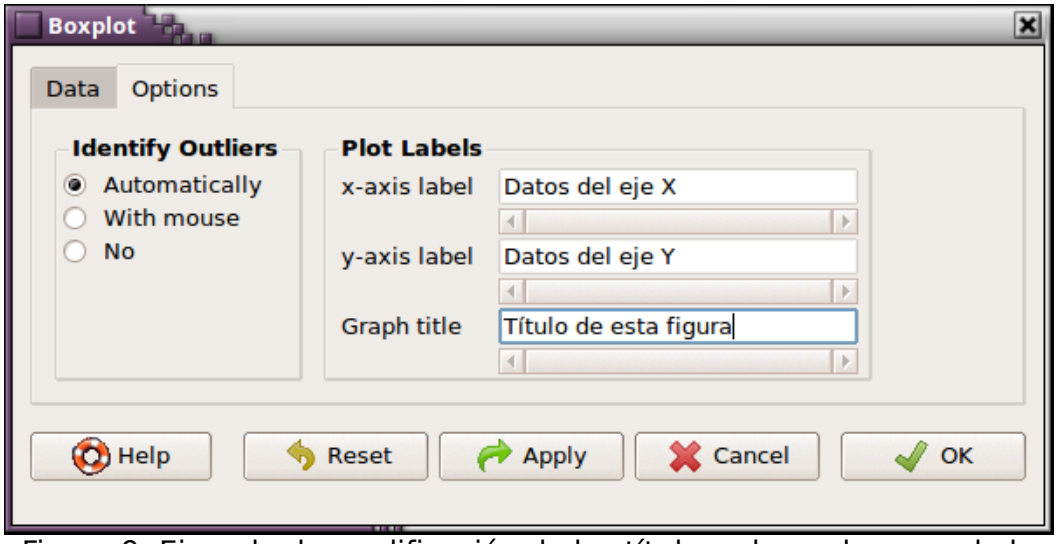

Figura 9: Ejemplo de modificación de los títulos y leyendas usando la interfaz gráfica R Commander.

Si nos fijáramos en el código exacto que se ha ejecutado, observaríamos cuáles son los atributos exactos que hay que escribir para personalizar estos parámetros:

```
Boxplot( ~ BL.mm., data=Lumley, id.method="y", xlab="Datos del eje X",
ylab="Datos del eje Y", main="Título de esta figura", sub="Subtítulo")
```

Como vemos en el código anterior los responsables de personalizar los nombres de los ejes y el título principal son `xlab`, `ylab` y `main`, respectivamente, seguidos de nuestro nombre entre comillas. Si queremos añadir un subtítulo a la figura, tenemos que añadir `sub="Lo que queramos aquí"`.

Modificar los iconos (**pch** y **cex**)

Cambiar el tipo de icono (pch)

Si estuviésemos representando un diagrama de dispersión, en vez de líneas, tendríamos puntos en el espacio. Si quisiéramos cambiar los iconos, tendríamos que añadir el atributo `pch`, pudiendo adquirir valores numéricos como se muestra en la Figura 10.

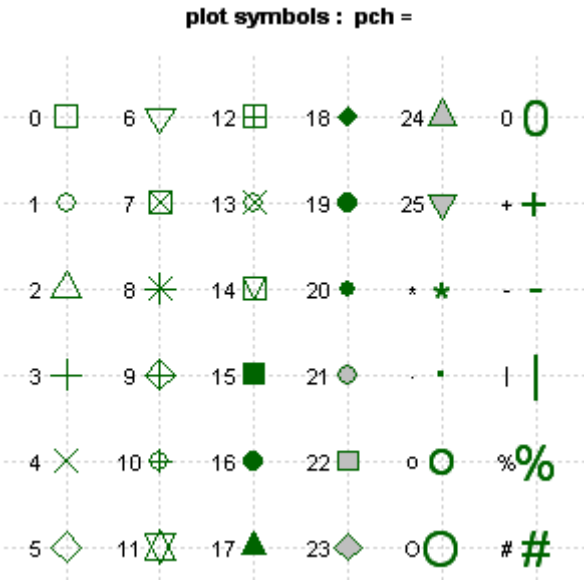

Figura 10: Valores de pch para cada uno de los símbolos.

Veamos un par de líneas de código de ejemplo:

```
plot(x, y, pch = 8)
plot(x, y, pch = 22, col ="blue", bg="red")
```

En el primer ejemplo vemos que el icono que muestra los datos es un asterisco (`pch = 8`).

En el segundo caso vemos que es un cuadrado (`pch = 22`), pero observamos que existen dos atributos de color situados después (`col` y `bg`, de *background*). Y aquí viene una parte curiosa. Los valores comprendidos entre el 21 y 25 (ambos inclusive) permiten no solo cambiar el color del borde del icono, sino también el fondo del icono. De ahí que sean los únicos iconos en la Figura 10 que tengan el fondo en gris. Por lo tanto, en ese

código de ejemplo, `col = “blue"` hace referencia al color del borde del icono, y `bg = “red"` se refiere al relleno del icono.

Cambiar el tamaño del icono (cex)

Es un atributo muy fácil de entender. Si en el código no aparece, de modo predeterminado se ejecuta cex = 1. Si queremos aumentar (o disminuir) su tamaño, tenemos que modificar ese valor numérico. Cogiendo los dos ejemplos del apartado anterior...

```
plot(x, y, pch = 8, cex = 0.5)
plot(x, y, pch = 22, col ="blue", bg="red", cex = 3)
```

… vemos que en el primer código se divide por la mitad el tamaño original del icono, mientras que en el segundo se triplica su tamaño.

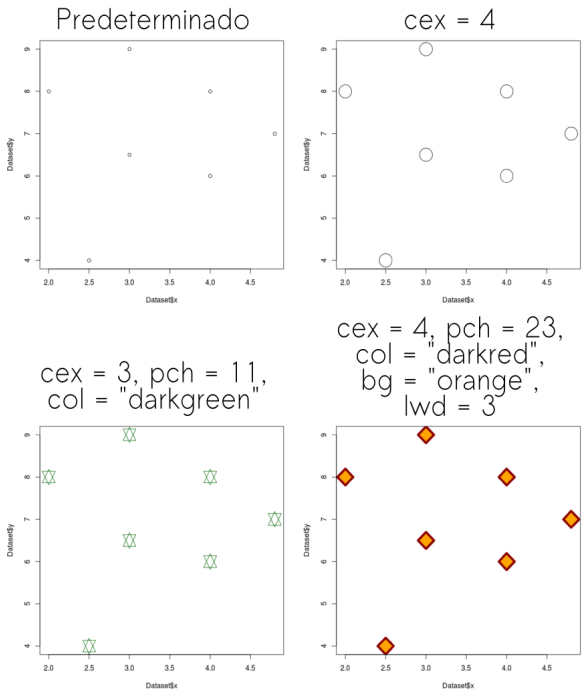

Figura 11: Ejemplos de aplicación de los atributos `cex`, `pch`, `col`, `bg` y `lwd` en la representación de puntos en un diagrama de dispersión.