Nota 5 - Graficación

Santiago Casanova y Ernesto Barrios

Graficación Básica

En esta sección aprenderemos conceptos básicos de visualización de información en la paquetería base de R. La función base para graficar es `plot()`. Esta es una fucnión sobrecargada, lo que significa que reconoce el tipo de información que le proporcionamos y nos imprime el resultado correspondiente.

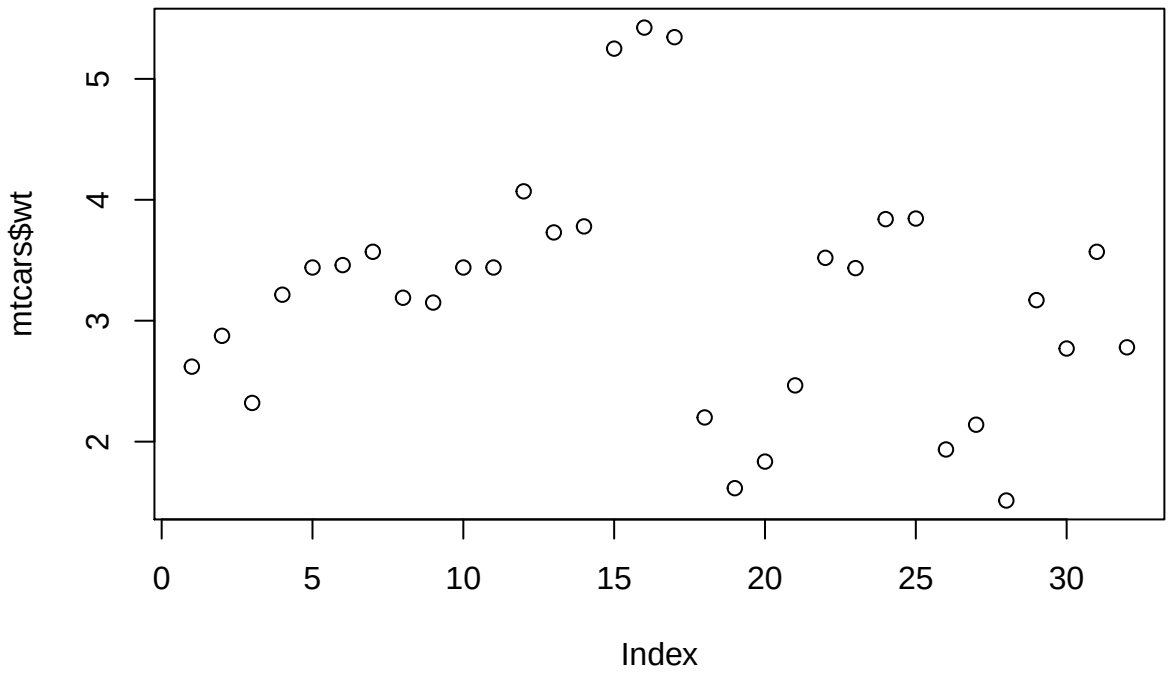
Por ejemplo, siguiendo los ejemplos de la nota anterior (Manipulación de Datos), usemos el dataset `mtcars` para graficar.

```
head(mtcars, 4)
```

```
##                mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4      21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag  21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710     22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
```

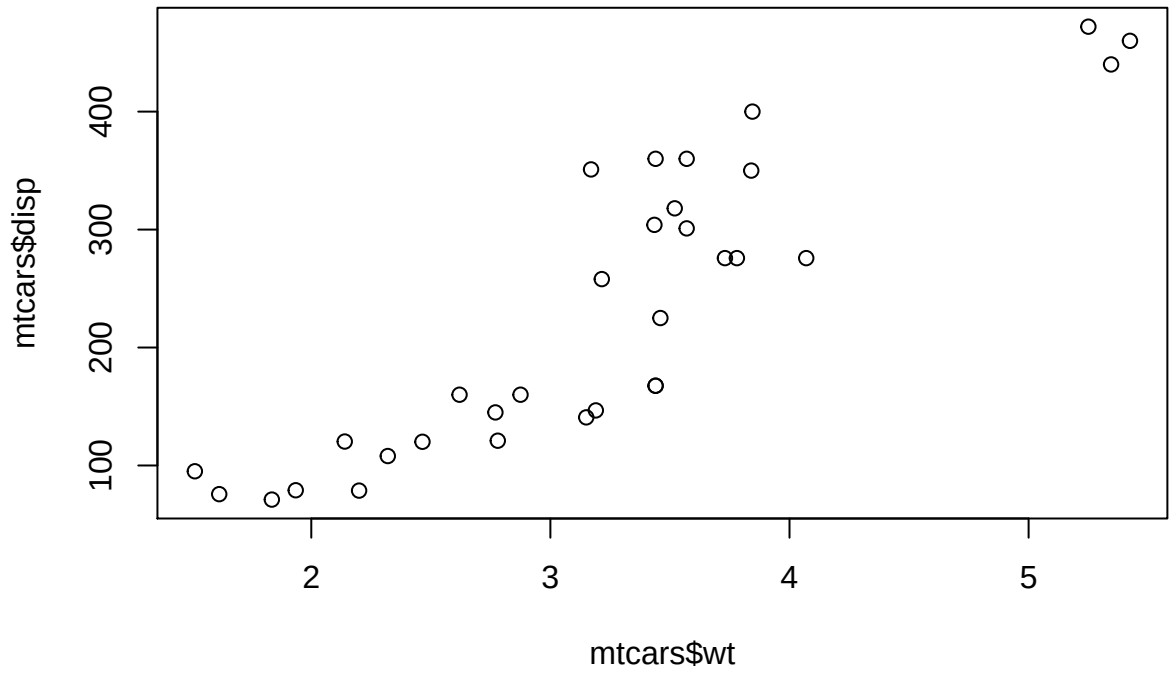
Si le proporcionamos un sólo vector numérico a la función `plot()`, esta nos graficará una figura de dispersión con el vector contra su índice en el vector.

```
plot(mtcars$wt)
```



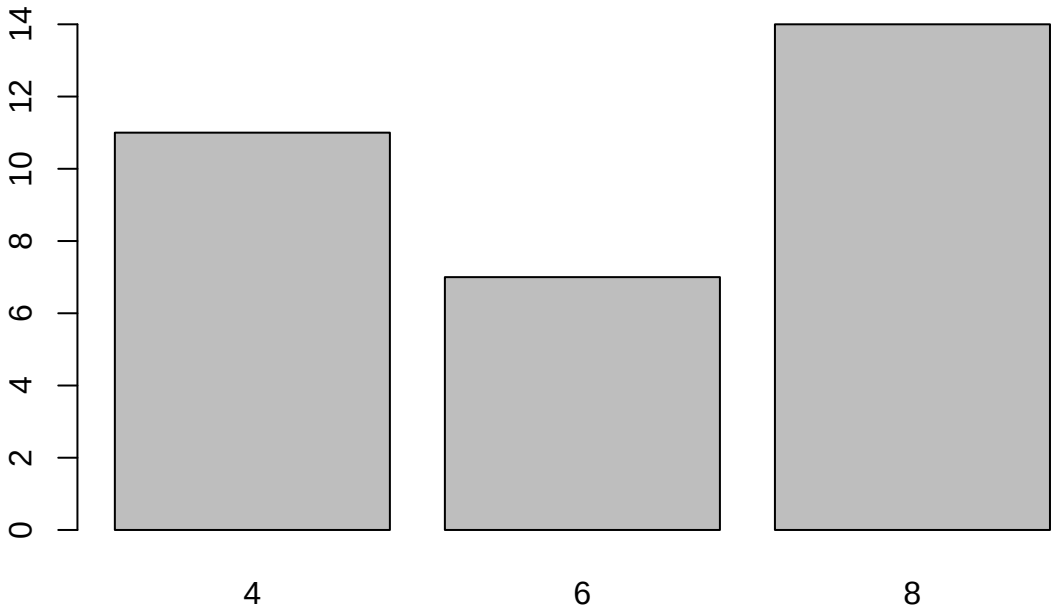
Si le proporcionamos dos vectores nos graficará uno contra el otro en el eje x y eje y.

```
plot(mtcars$wt, mtcars$disp)
```



Ahora veamos que sucede si le damos un vector de factores a la función `plot()`. Para esto tenemos que convertir la columna `cyl` a datos categóricos en lugar de numéricos.

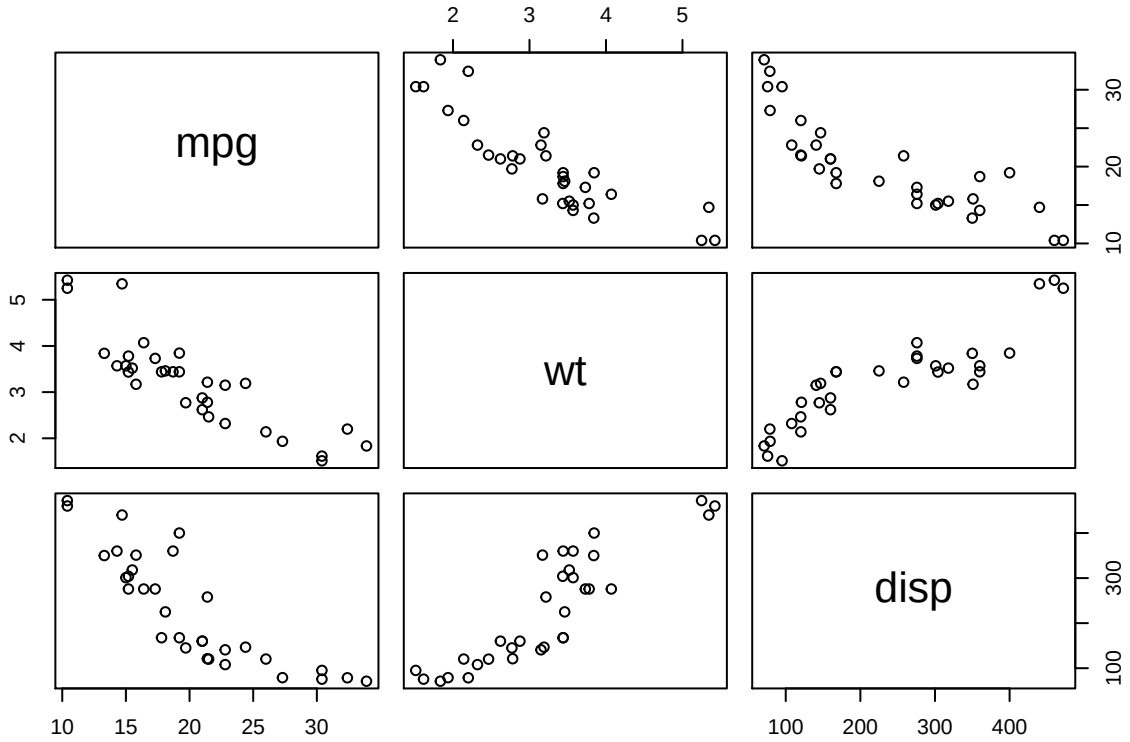
```
plot(as.factor(mtcars$cyl))
```



La salida es un gráfico de barras. Tenemos una barra por cada nivel del factor y la altura representa el número de repeticiones. Nótese que este es diferente a el histograma que veremos más tarde.

Ahora evamos que sucede si le proporiconamos un arreglo a la función `plot()`. En principio esto no suena posible pero veremos que nos dá un resultado muy interesante. Primero vamos a restrngir nuestro arreglo a solo 3 columnas.

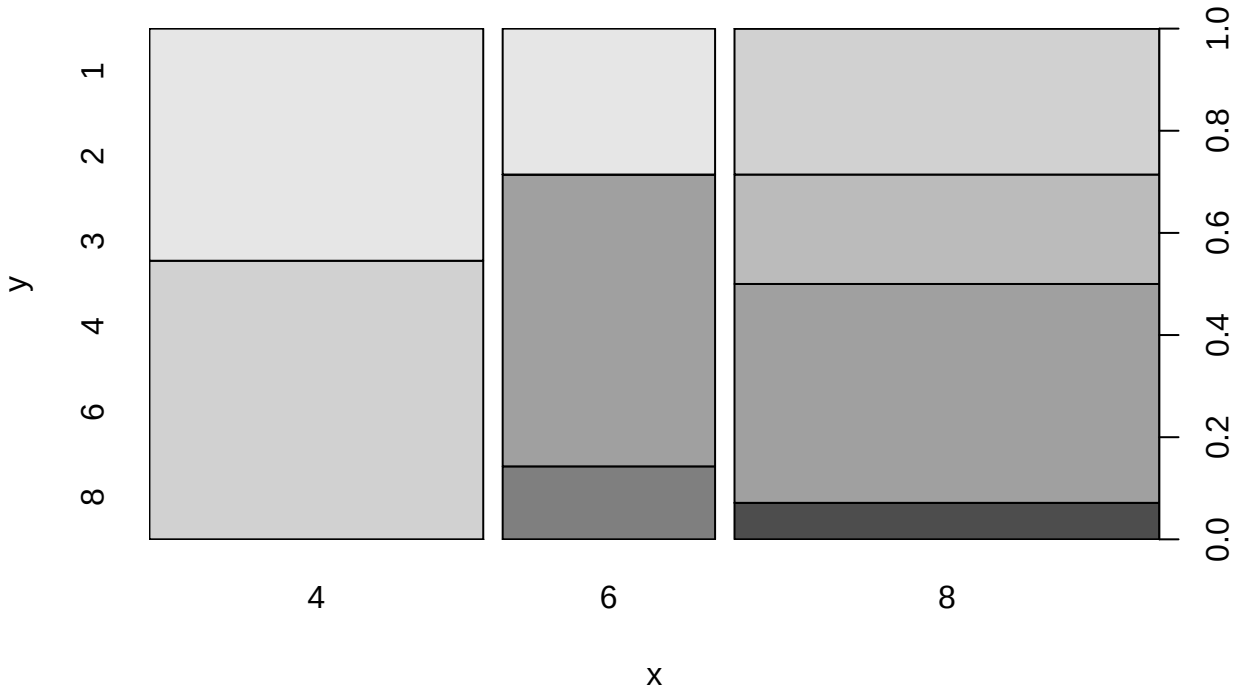
```
arreglo_gráfica <- mtcars[,c("mpg", "wt", "disp")]

plot(arreglo_gráfica)
```



La salida es una matriz de dispersión que compara todas las variables contra todas. Veamos que en términos matriciales es "simétrica".

Ahora analizemos otras combinaciones. Si le proporcionamos dos factores:

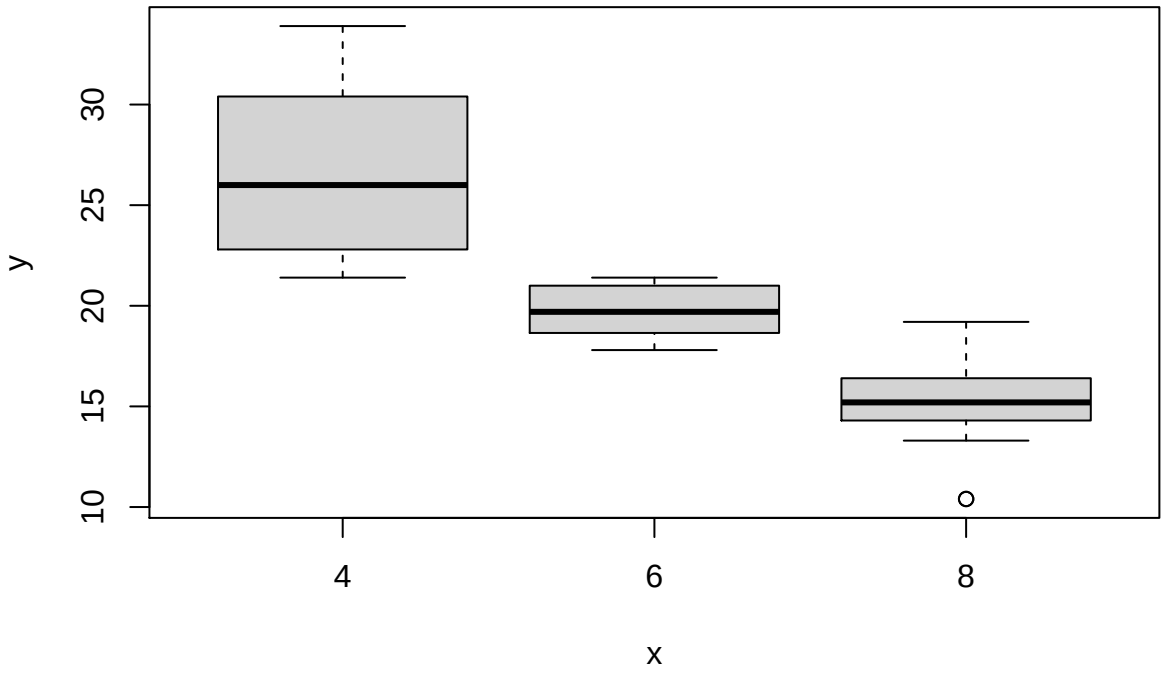
```
plot(as.factor(mtcars$cyl), as.factor(mtcars$carb))
```



Nos da un grafico de mosaico que combina ambos factores.

Para un factor y una variable numérica:

```
plot(as.factor(mtcars$cyl), mtcars$mpg)
```



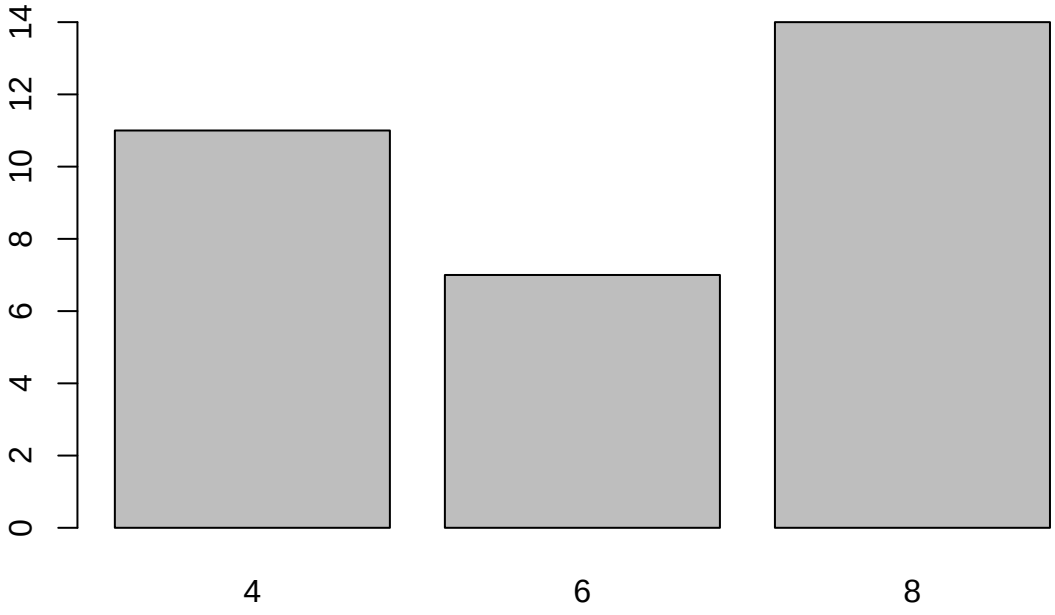
Nos da un gráfico de caja y brazos para la variable numérica, separada por cada nivel del factor.

Graficos específicos

Estos resultados también se pueden obtener con las fucniones específicas `boxplot()` y `barplot()` para dejar en claro cuál es el resultado buscado. Por lo mismo no es necesario convertir a factores los datos antes de graficar en estos casos.

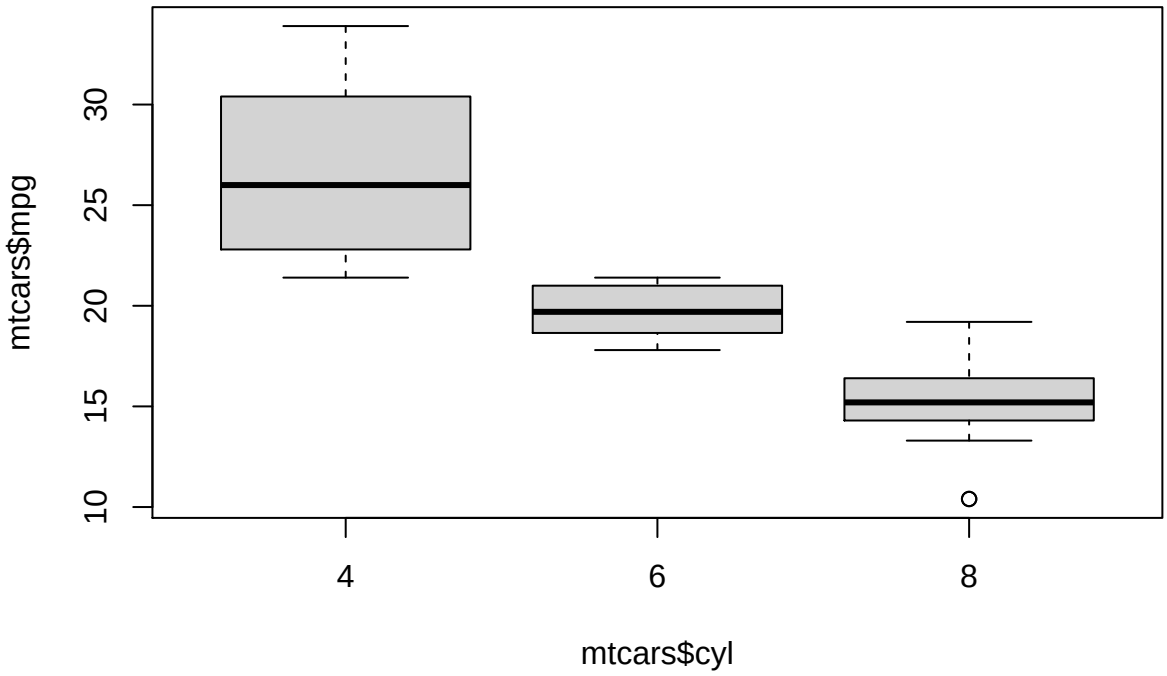
Las siguientes funciones nos regresarán exactamente los mismos resultados que obtuvimos con la función plot.

```
barplot(table(mtcars$cyl))
```



En este caso usamos la función table para que nos diera un recuento de cada valor (con nombres)

```
boxplot(mtcars$mpg~mtcars$cyl)
```



Para este resultado indicamos que queremos que una variable sea agrupada por otra con el símbolo `~`. Veamos también que esta version incluye nombres para los ejes.

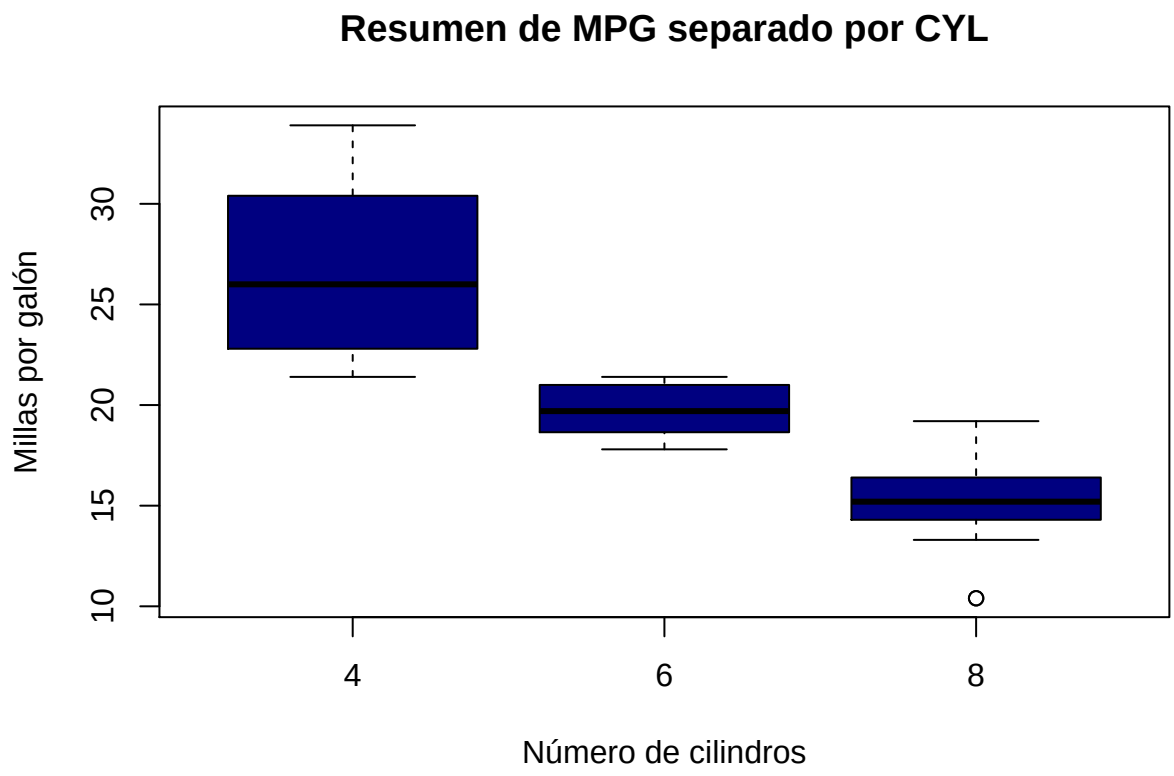
Personalización de gráficos

Todas las funciones de graficación tienen una serie de parámetros que podemos modificar para darle diferentes colores o etiquetas a los datos. Los principales son los siguientes:

`col`: el color de los datos `main`: el título de la figura `xlab/ylab`: el título de cada eje

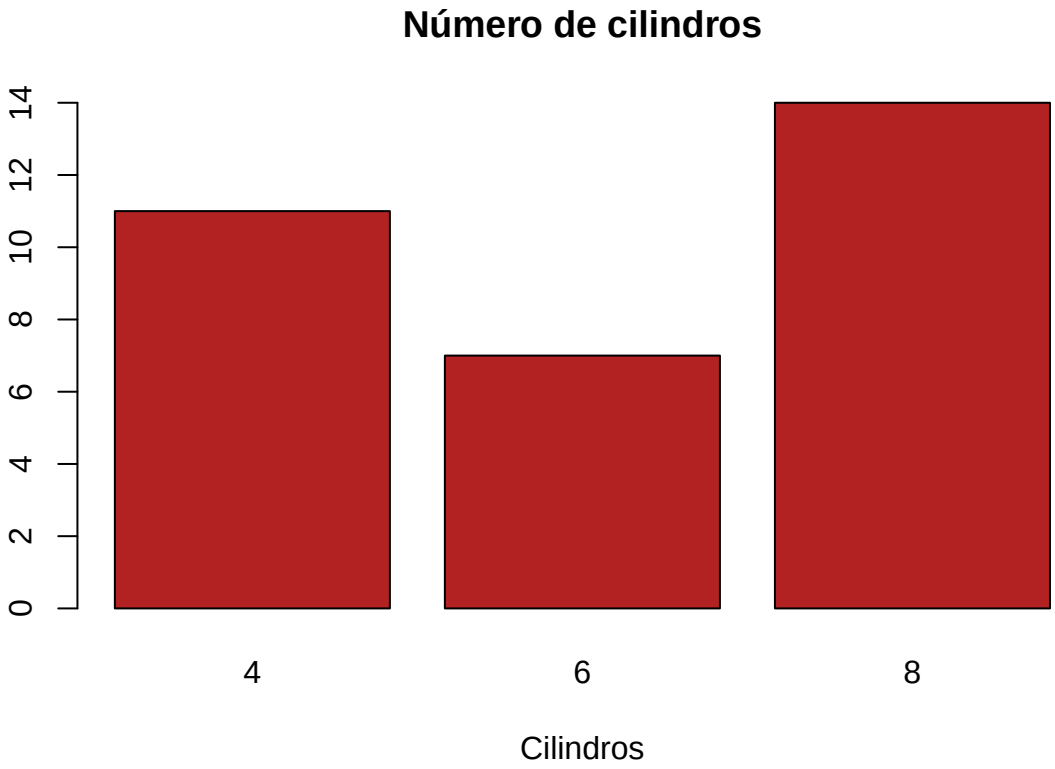
Probémoslos con el grñafico anterior.

```
boxplot(mtcars$mpg~mtcars$cyl, col = "navy", main = "Resumen de MPG separado por CYL", xlab = "Número d
```



Otro ejemplo completo:

```
barplot(table(mtcars$cyl), col = 'firebrick', xlab = 'Cilindros', main = 'Número de cilindros')
```

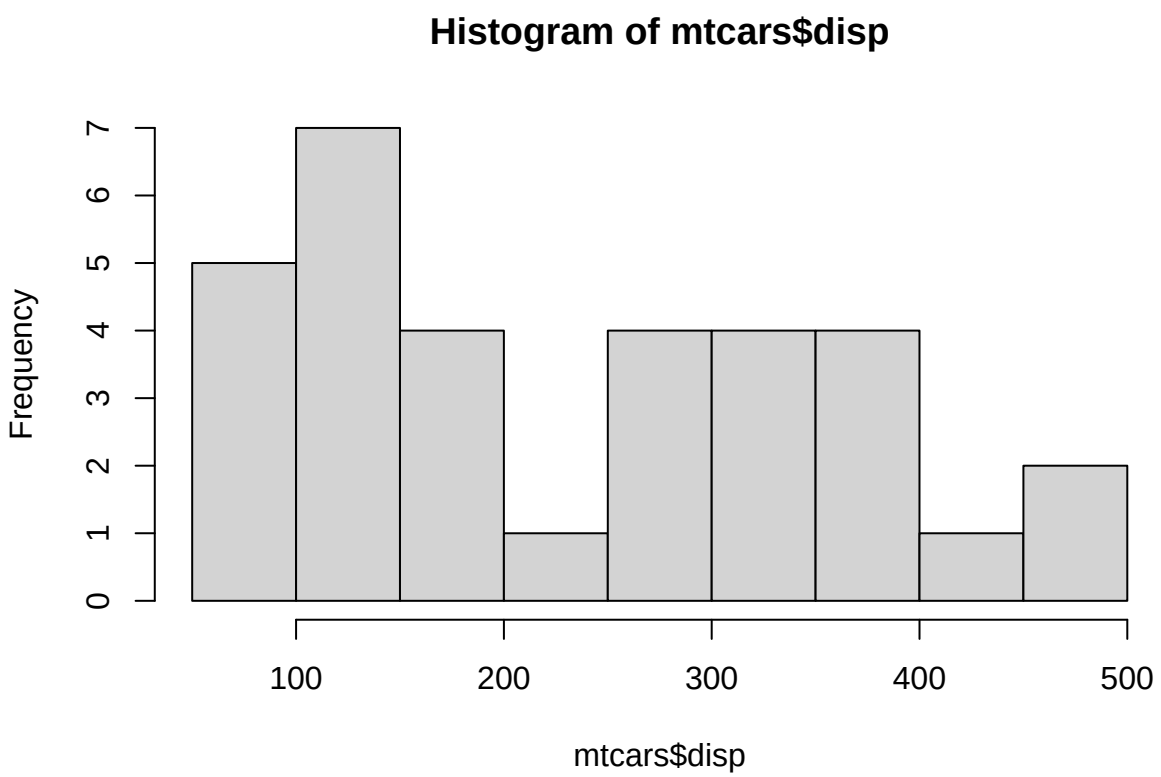


Histograma

El histograma es visualmente similar al gráfico de barras pero cumple otro propósito. El histograma toma una serie de datos (continuos o discretos) y los agrupa en "cubetas" o particiones regulares. El eje y representa el número de observaciones que caen en cada cubeta y por lo tanto esta visaulización es útil para representar densidades.

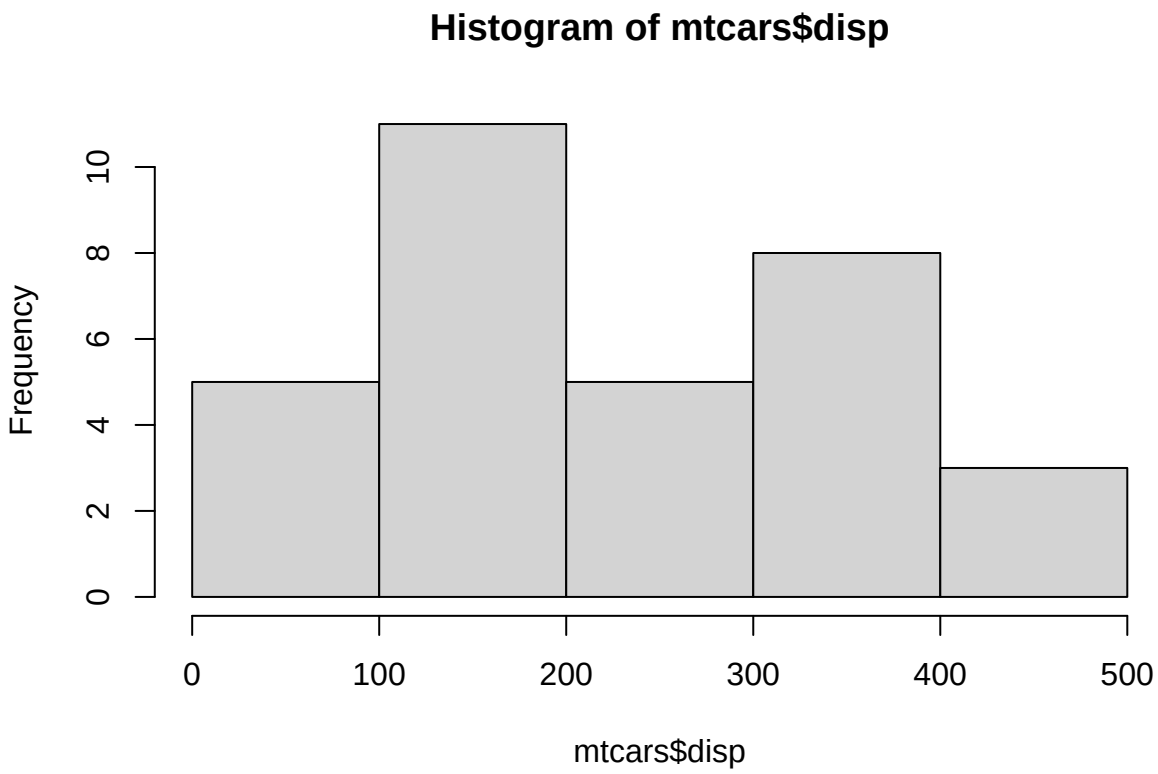
Usamos la función `hist()` para obtener un histograma.

```
hist(mtcars$disp)
```

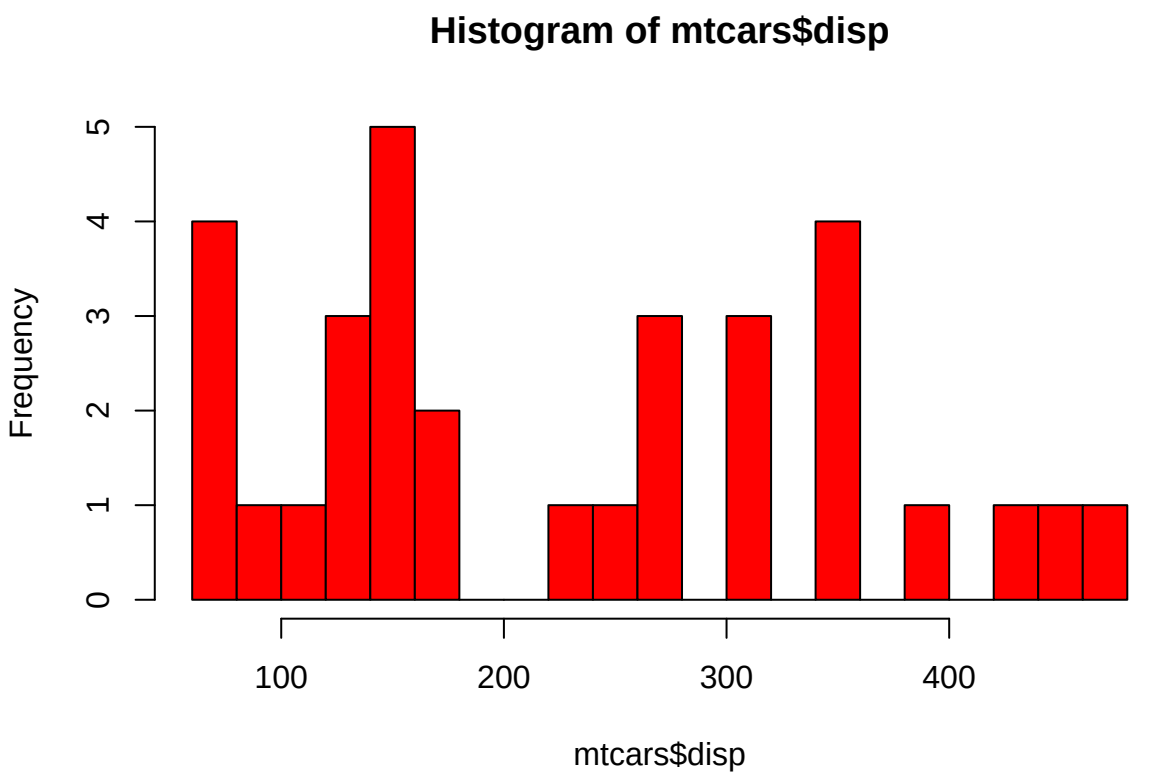


Si sólo le damos nuestro vector como parámetro, `R` ya nos da un título general y para cada eje dscribiendo el propósito del histograma. Además de los parámetros estéticos (xlab, col, etc.) tambiñen podemos pedirle un número diferente de cubetas con `breaks`.

```
hist(mtcars$disp, breaks = 5)
```



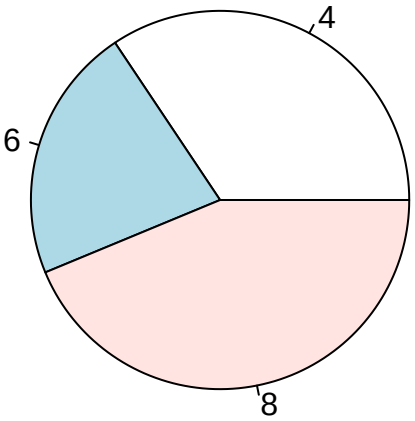

```r
hist(mtcars$disp, breaks = 20, col = "red")
```



Pie

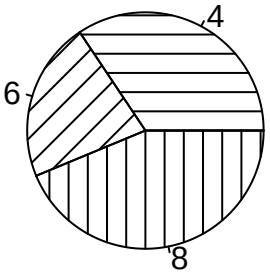
Por útlimo analizaremos cómo crear un gráfico de pie, aunque por lo general no es recomendado utilizarlo en la práctica. Esto es tan sencillo como usar la función `pie()` y proporcionarle un vector de datos discretos agrupados.

```
pie(table(mtcars$cyl))
```



Una característica especial de los gráficos de pie en `R` es la propiedad `density` que, en conjunto con la propiedad `angle`, cambia de colores a lineas cada sección del pie. Además, es modificable el radio,

```
pie(table(mtcars$cyl), density = 10, angle = c(0,45,90), radius = 0.5)
```



Por último nótese que `R` construye el gráfico en el sentido contrario a las manecillas del reloj, empezando a 90º. Esto también se puede cambiar con la propiedad `clockwise`.