Módulo 3. Gráficos básicos con R

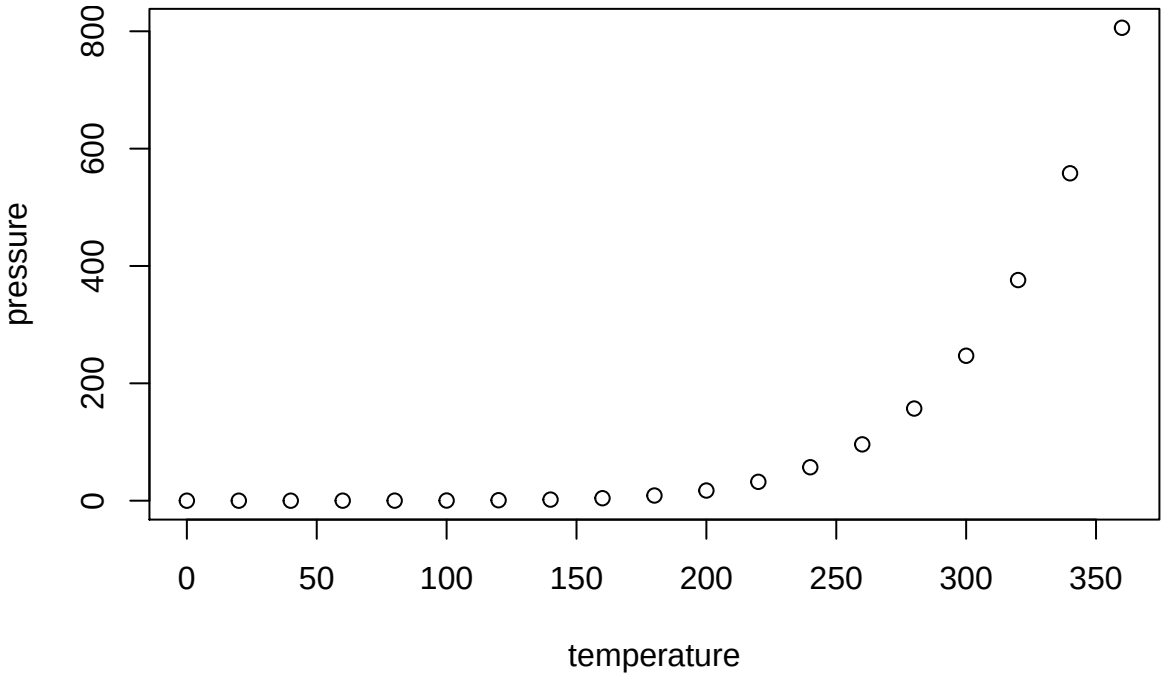
Curso Herramientas para Data Science

Prof. Jose Jacobo Zubcoff Vallejo, PhD
Universidad de Alicante

Módulo 3. Gráficos básicos con R

A partir de unos datos cargados en memoria se pueden hacer diversos tipos de gráficos. En el módulo 1 vimos como hacer un gráfico simple de puntos con los datos sobre presión ("pressure").

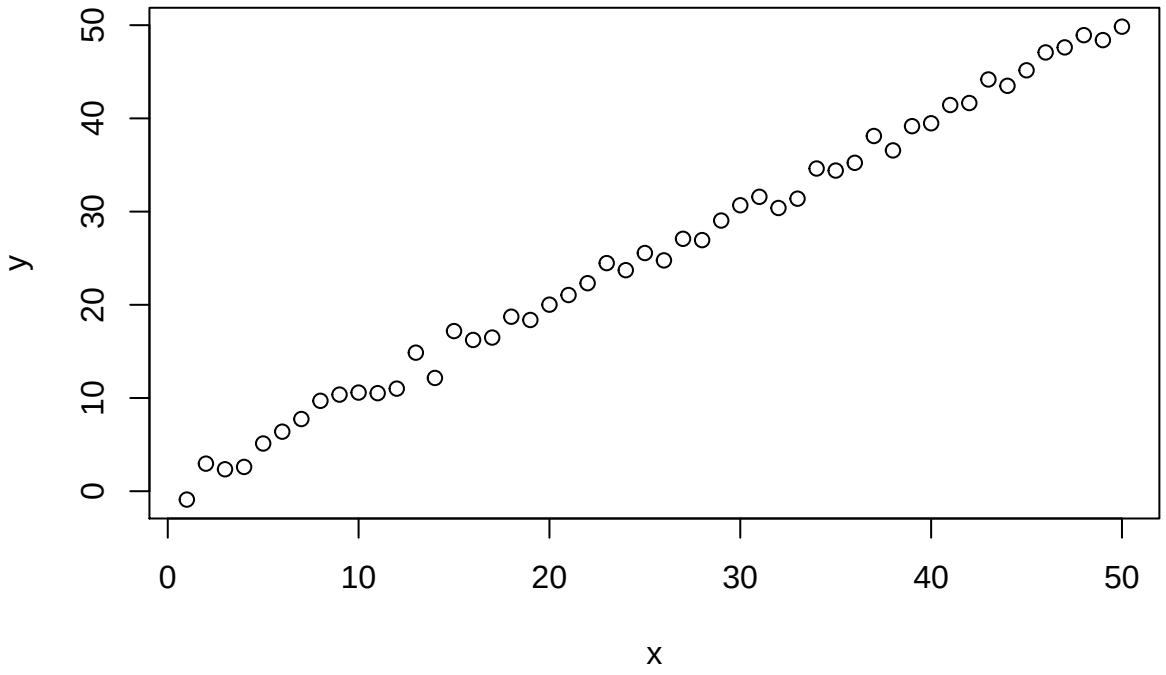
```
plot(pressure)
```



En este módulo, presentaremos funcionalidades y mejoras sobre los comandos básicos para poder controlar a nuestro gusto la forma y colores de nuestro gráfico. Además, para poder guardar los que consideremos necesario.

Para empezar, para visualizar algunos gráficos simples podemos crear unos datos aleatorios. Por ejemplo, crearemos una variable que contenga 50 datos aleatorios a partir de una variable normal (o gaussiana).

```
x <- 1:50            # asignamos a x los valores de 1 a 50
y <- x + rnorm(x)    # rnorm() genera datos aleatorios con probabilidad normal
plot(x,y)            #nube de puntos de x,y
```



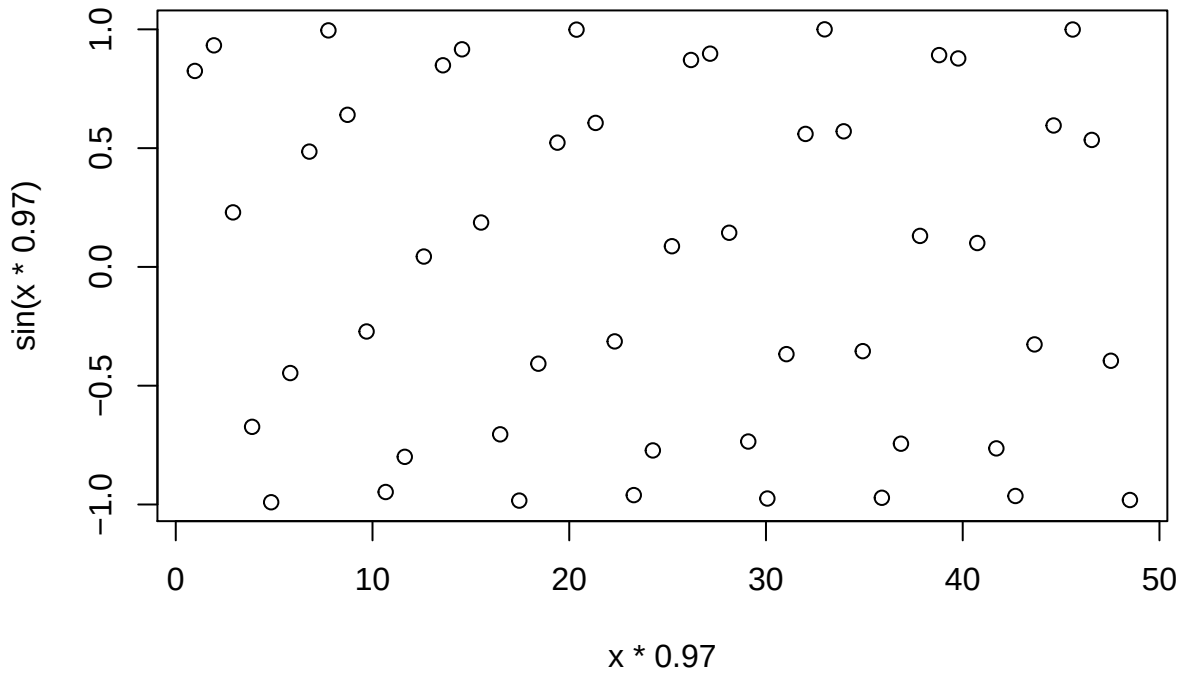
La importancia de la relación de aspecto

Los gráficos sirven para representar datos, por lo que es una herramienta muy potente, y debe usarse para mostrar de una manera visual, y simple, el patrón que hayamos encontrado en los datos.

Para ello debemos controlar algunas cosas, como el color y la forma. Empezaremos por esta última, porque la relación de aspecto puede llegar a ocultar un patrón si no tenemos cuidado de ello.

La ventana donde se muestran los gráficos se puede cambiar de tamaño, pero debemos tener cuidado de mantener siempre la mejor relación de aspecto. Esto lo podemos ver en el siguiente juego visual:

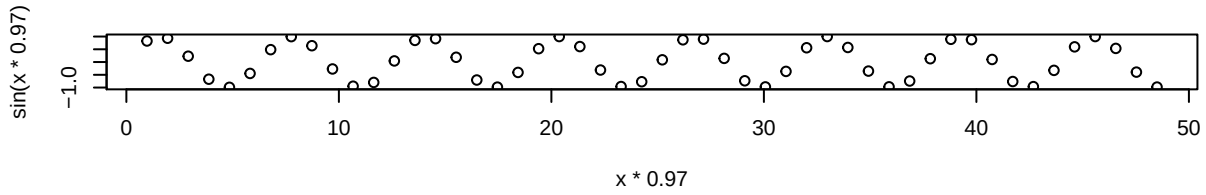
```
plot(x*0.97, sin(x*0.97))
```



En dicho gráfico no parece adivinarse ningún patrón. Quizas si cambiamos de tamaño la ventana aparezca el patrón existente. Podemos hacerlo con el ratón, o bien, podemos forzar que la representación ocupe solo una parte de la ventana usando la función "par(mfrow=c(1,1))" (una única ventana y la gráfica aparece en toda la ventana), pero ajustando ahora a "par(mfrow=c(3,1))" (una única ventana con tres filas para hacer gráficos). Debemos saber que funciona con la ventana de gráficos de la interfaz de R. En otros entornos puede no tener efecto visible (como en guiones destinados a obtener documentos RMarkdown, Shiny o R Notebook).

Ahora representaremos los mismos datos, pero en un tercio de la misma ventana, veremos si el patrón existente sale a la luz:

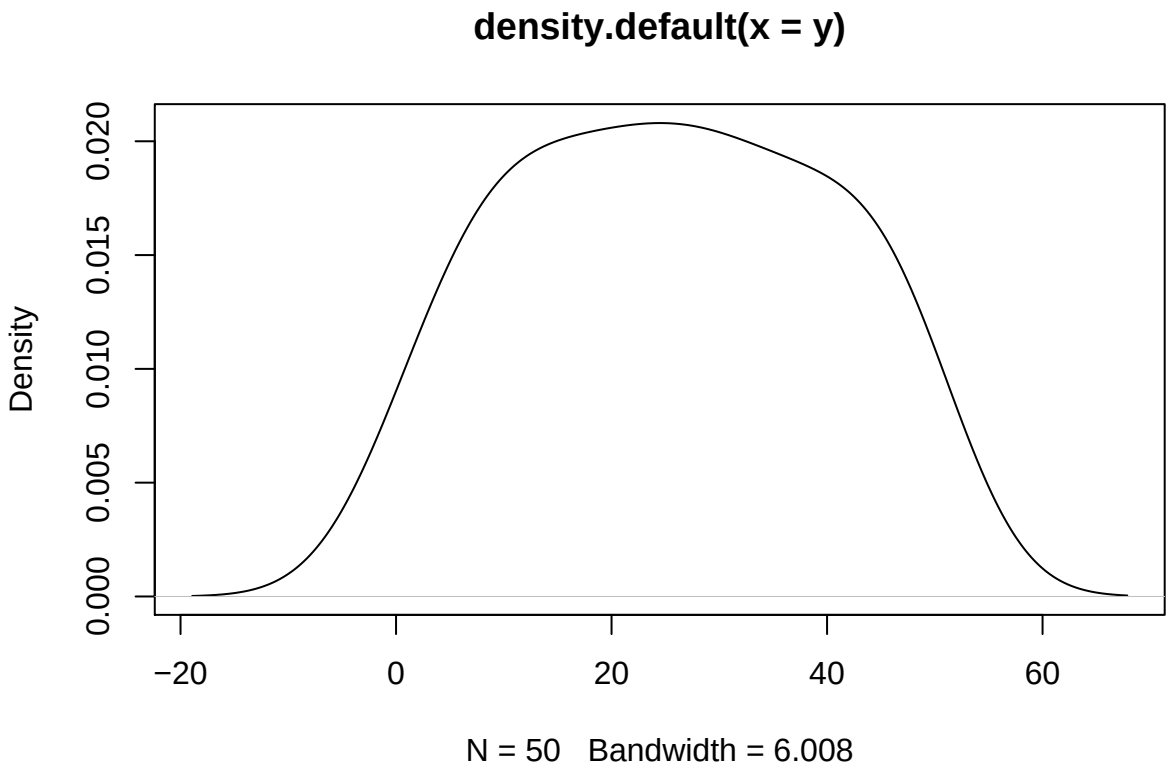
```
par(mfrow=c(3,1))     # preparamos una Ventana de 3 filas, 1 columna para plots
plot(x*0.97, sin(x*0.97))
par(mfrow=c(1,1))     # volvemos a una Ventana de 1 fila, 1 columna para plots
```



Diagramas de densidad

Con la función "plot(density())" podemos representar visualmente la densidad de una variable.

```
plot(density(y))              #Plot de la Densidad de y
```

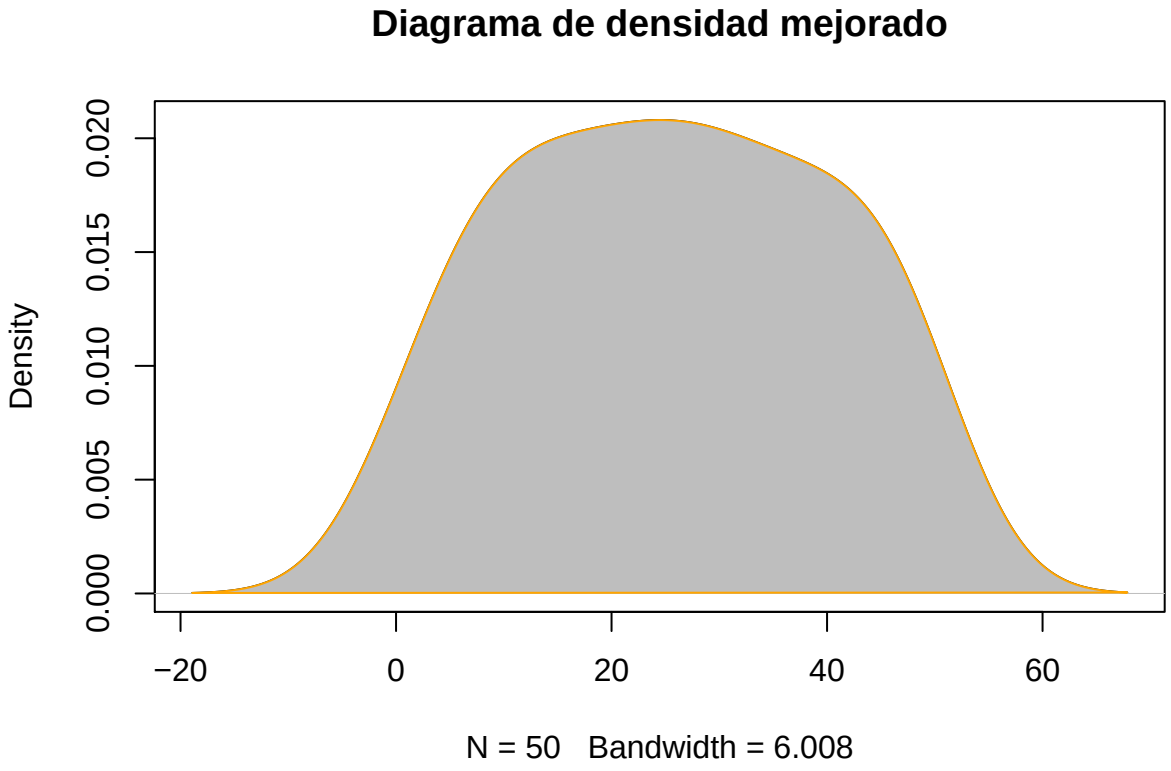



```
density(y)                    #datos numéricos sobre densidad de y
```

```
##
## Call:
```

```
##  density.default(x = y)
##
## Data: y (50 obs.);   Bandwidth 'bw' = 6.008
##
##        x                 y
##  Min.   :-18.920   Min.   :2.221e-05
##  1st Qu.:  2.776   1st Qu.:2.265e-03
##  Median : 24.471   Median :1.376e-02
##  Mean   : 24.471   Mean   :1.151e-02
##  3rd Qu.: 46.167   3rd Qu.:1.959e-02
##  Max.   : 67.862   Max.   :2.080e-02
```

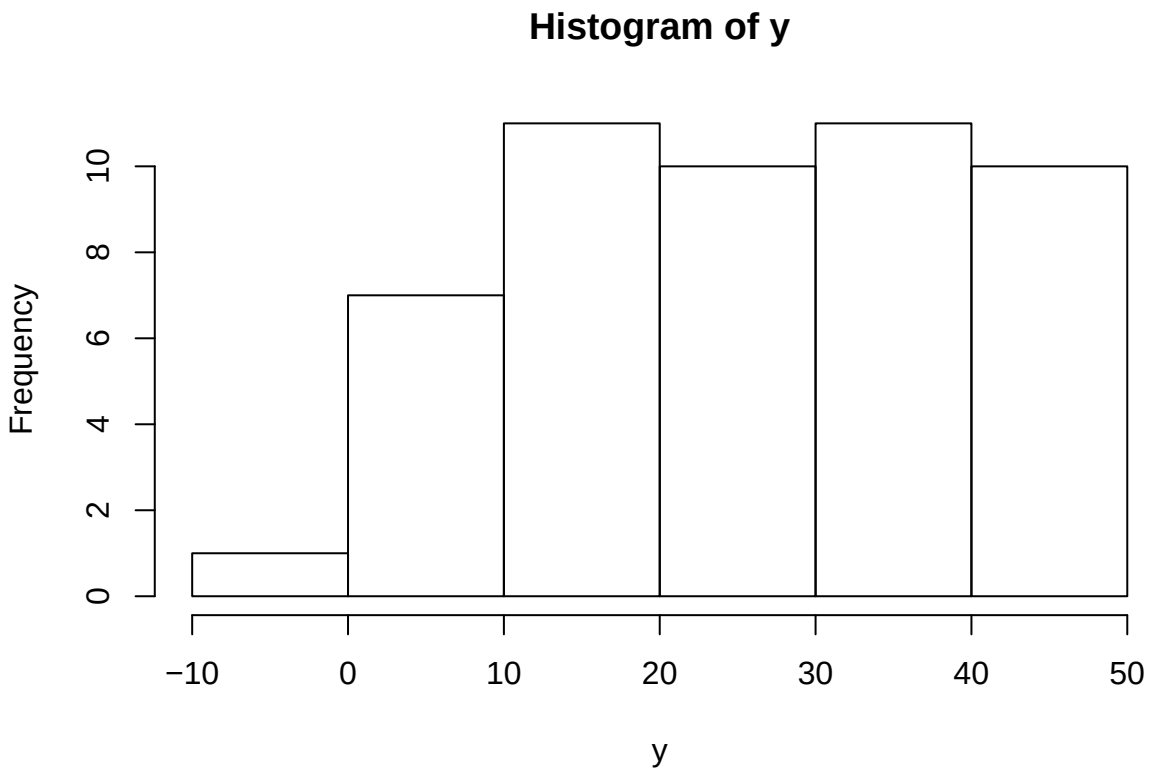
```r
# Diagrama de densidad con fondo de color
#asignamos la densidad de una variable a un vector para luego graficar
d <- density(y)

#dibujamos el diagrama de densidad y agregamos el título
plot(d, main="Diagrama de densidad mejorado")

#ahora pintamos con colores el fondo y el borde
polygon(d, col="grey", border="orange")
```



Histogramas

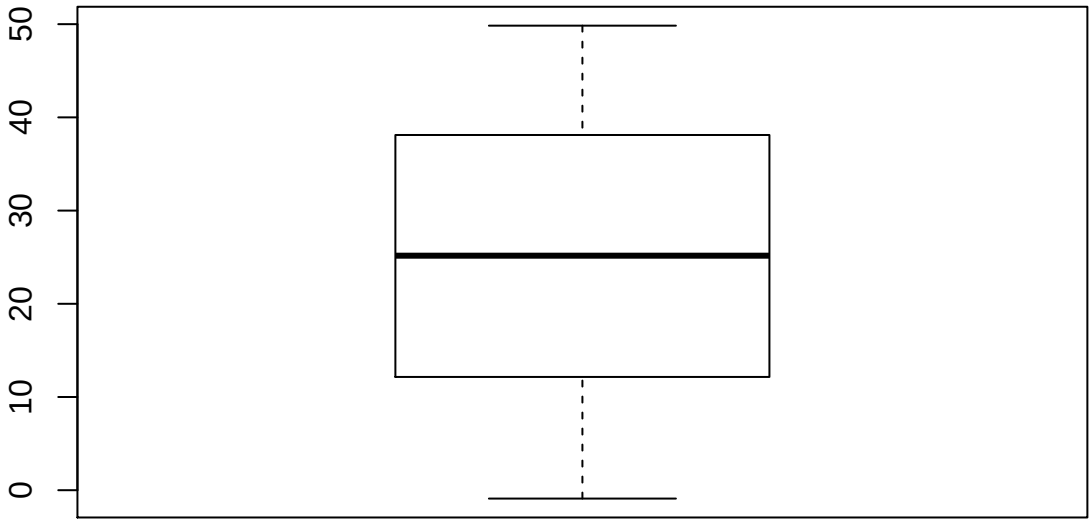
Podemos hacer un histograma muy fácilmente con R. Solo necesitamos pasar la variable que contiene los datos a la función "hist()".

```r
hist(y)              #histograma de y
```



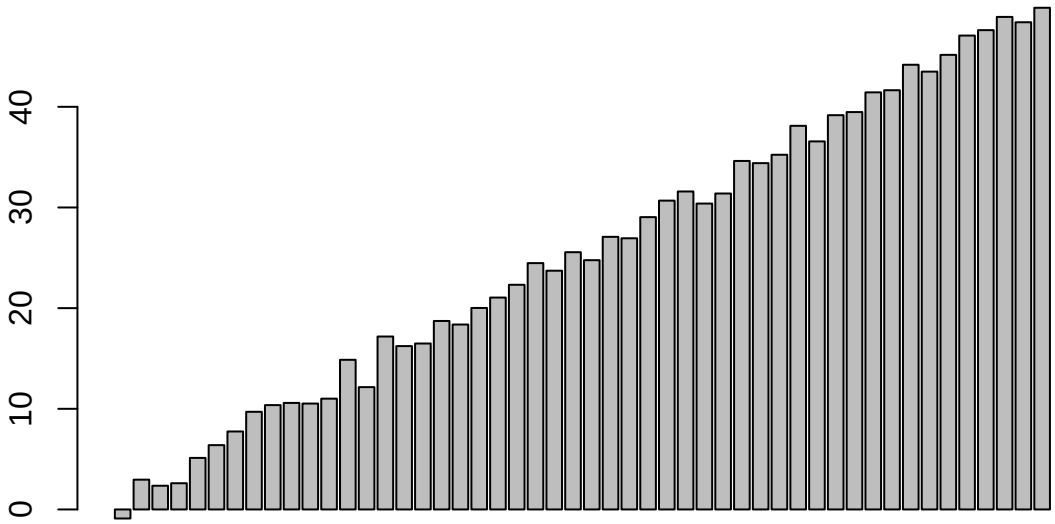
También podemos dibujar un diagrama de cajas o boxplot con la función "boxplot()".

```
boxplot(y)              #Diagrama de cajas de y
```



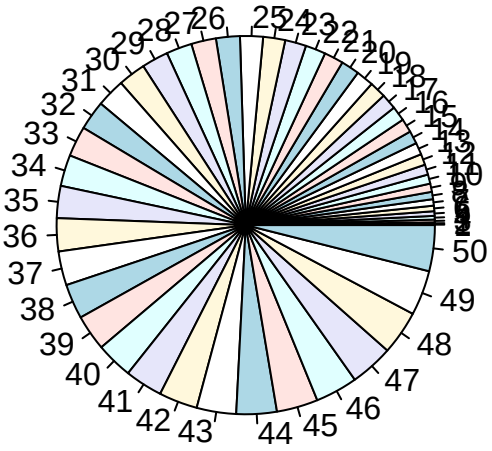
La función para dibujar diagramas de barra en R es "barplot()".

```
barplot(y)              #Diagrama de barras de y
```



También podemos dibujar diagramas tipo tarta, o quesitos en R con la función "pie()".

```
pie(x)                    #Diagrama de tartas de x
```

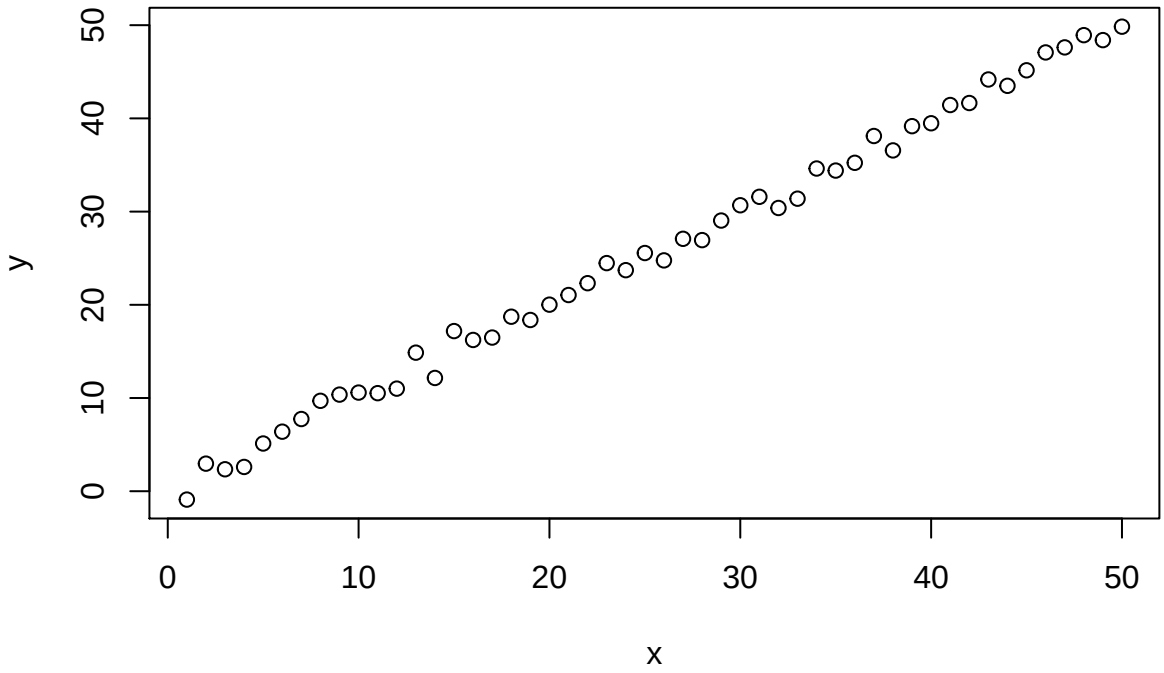


Formas en los gráficos

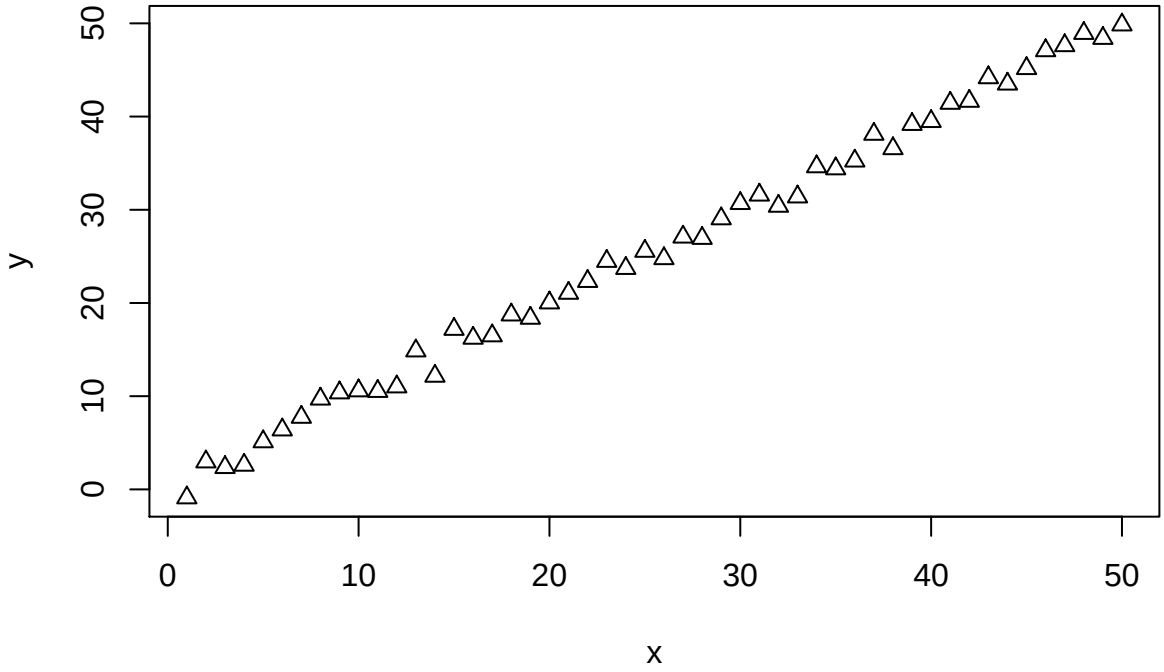
Para controlar las formas de la representación gráfica se puede usar el parámetro "pch=" que en algunas funciones permite elegir la forma.

Algunos ejemplos del uso del parámetro "pch=":

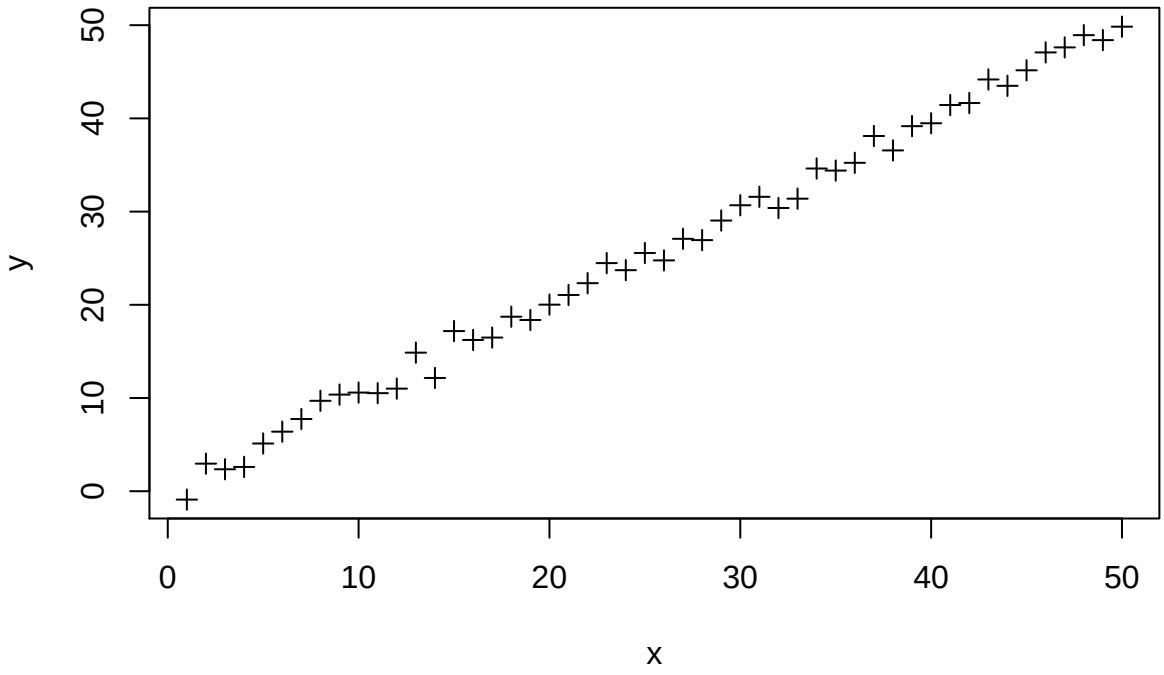
```
plot(x, y, pch=1)  # círculos
```



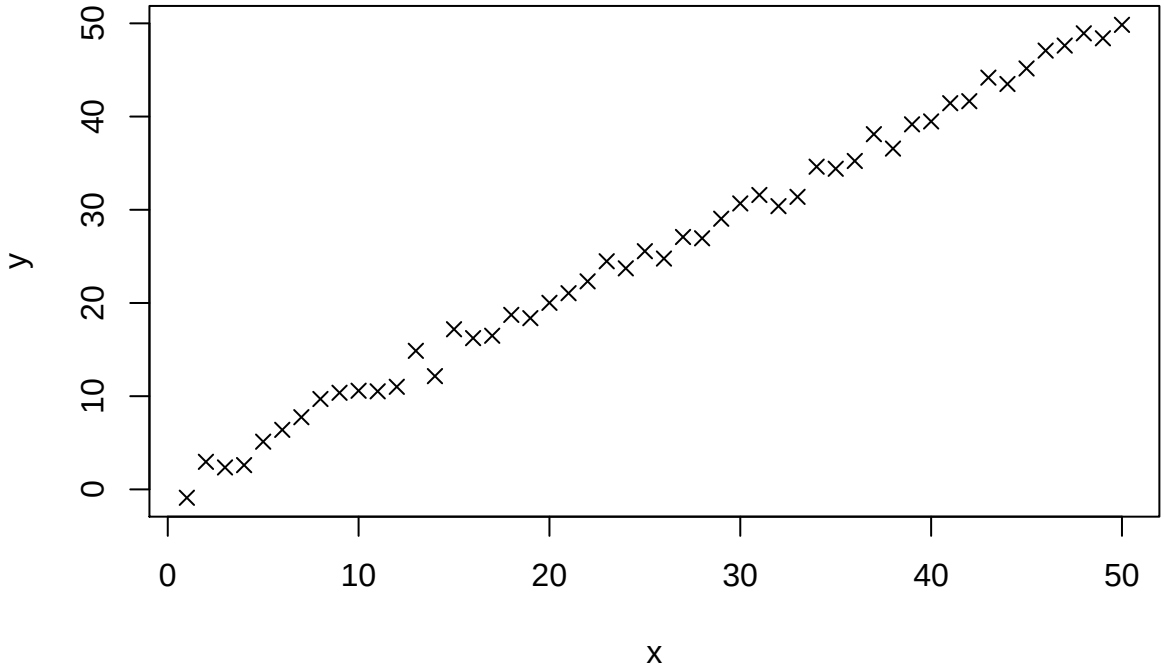

```
plot(x, y, pch=2)  # triangulos
```



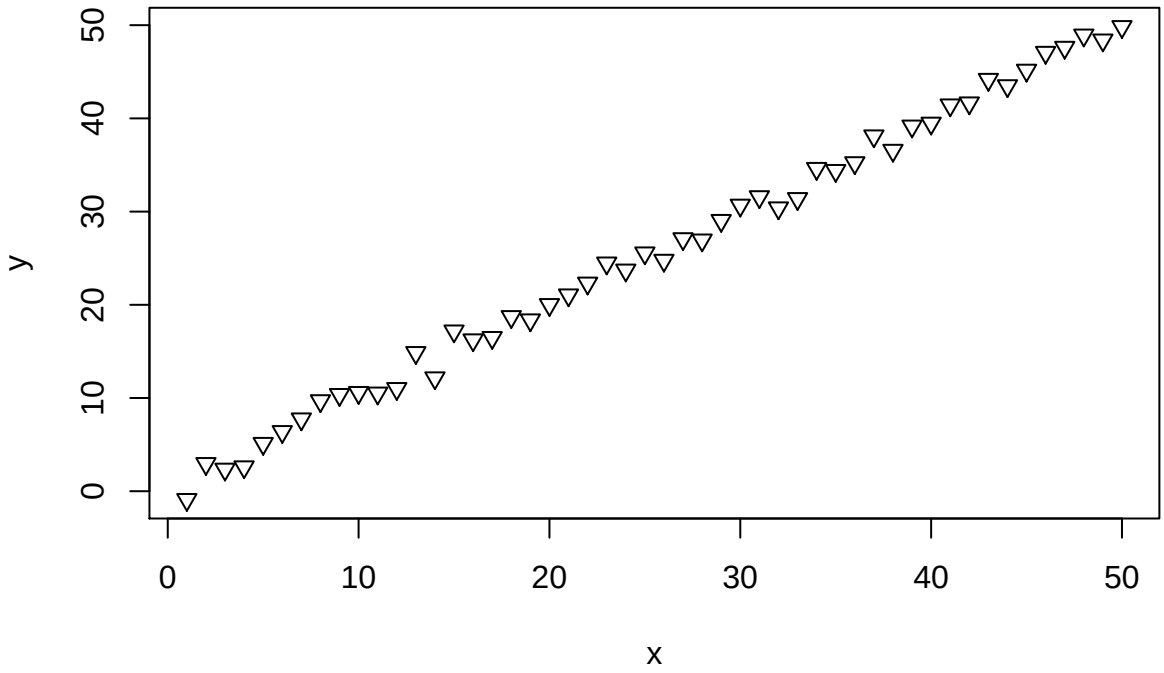

```
plot(x, y, pch=3)  # cruces
```



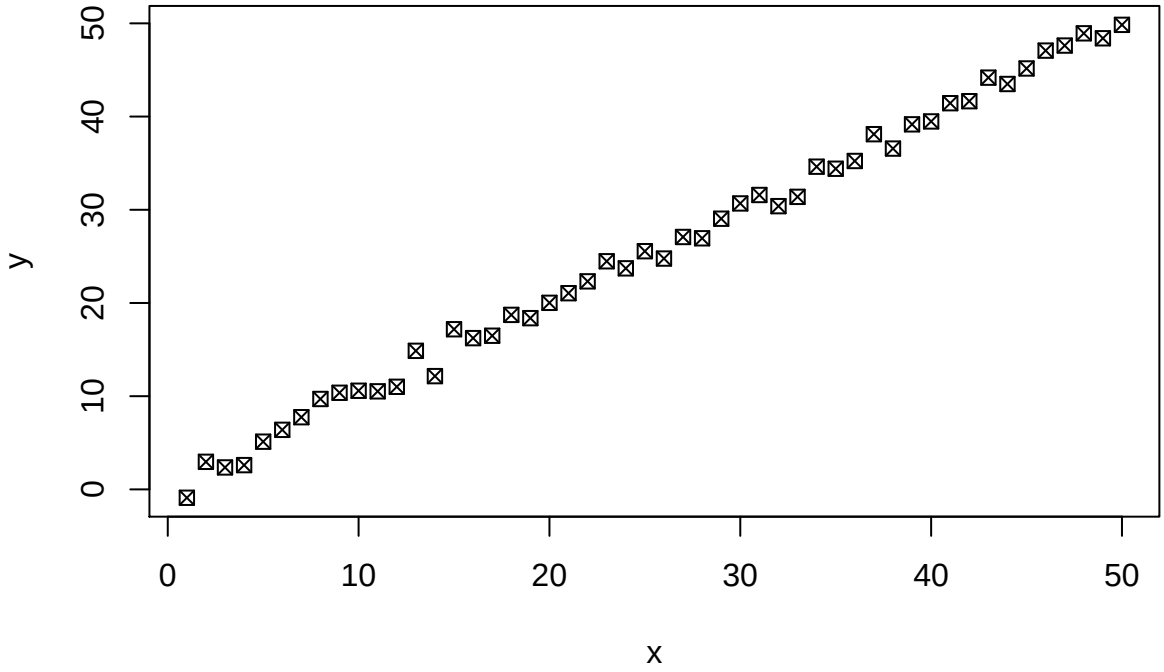

```r
plot(x, y, pch=4)  # con x
```



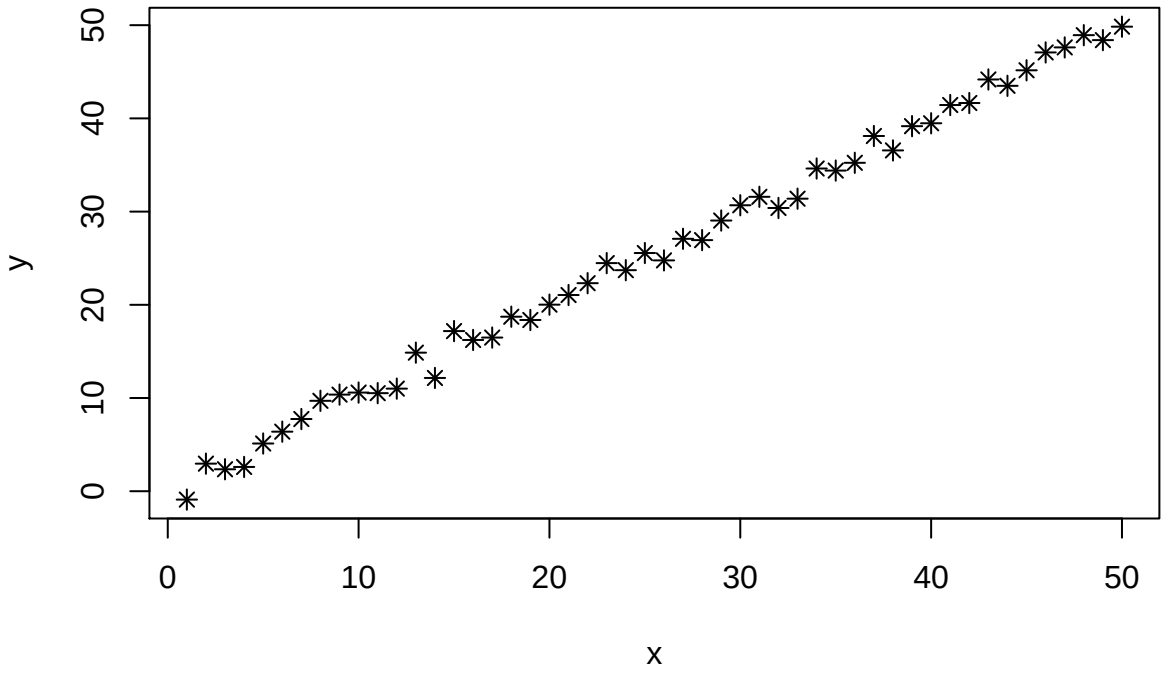

```r
plot(x, y, pch=6)  # triangulos invertidos
```



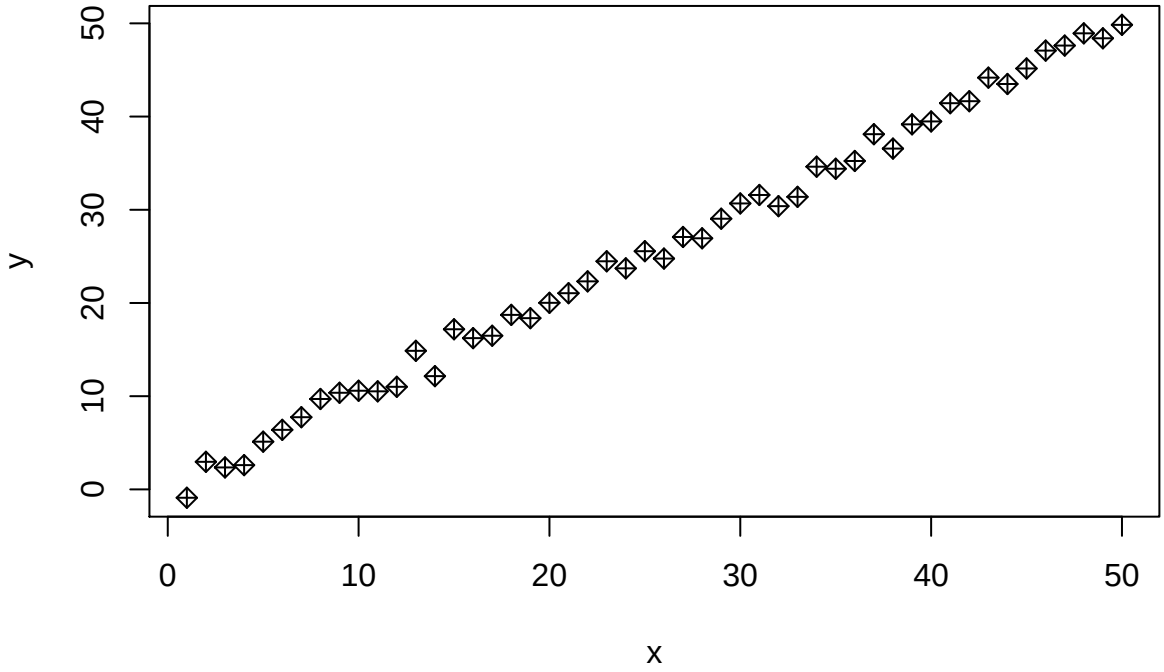

```
plot(x, y, pch=7)  # cuadrados con x
```



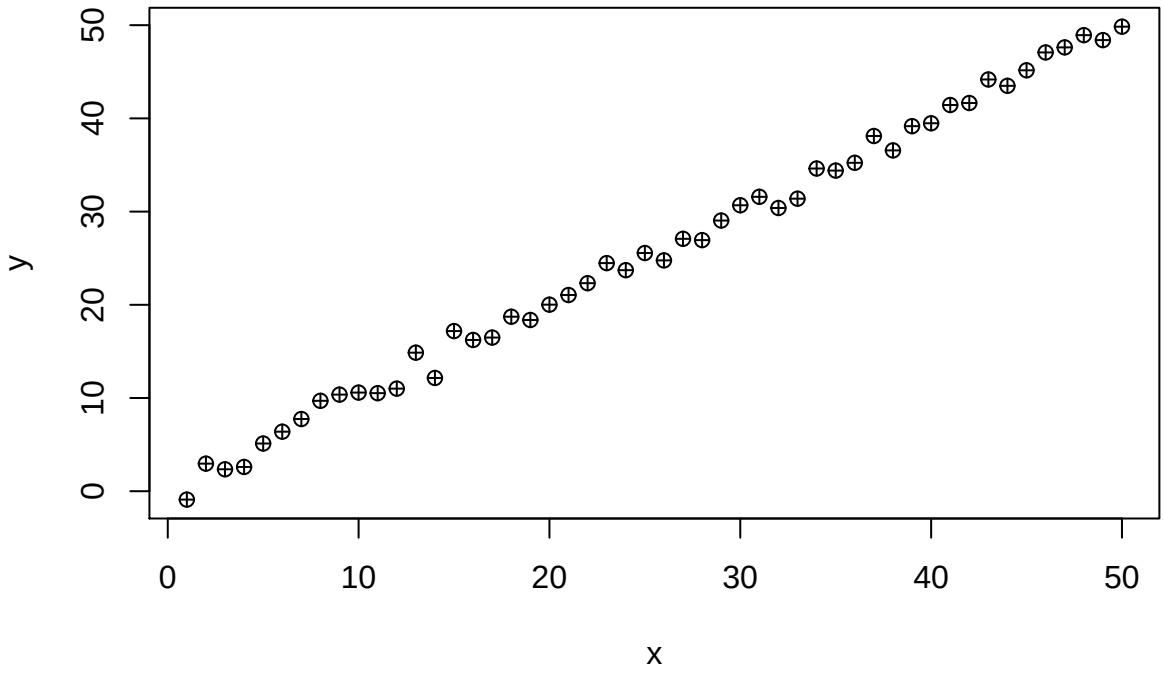

```
plot(x, y, pch=8)  # asteriscos
```



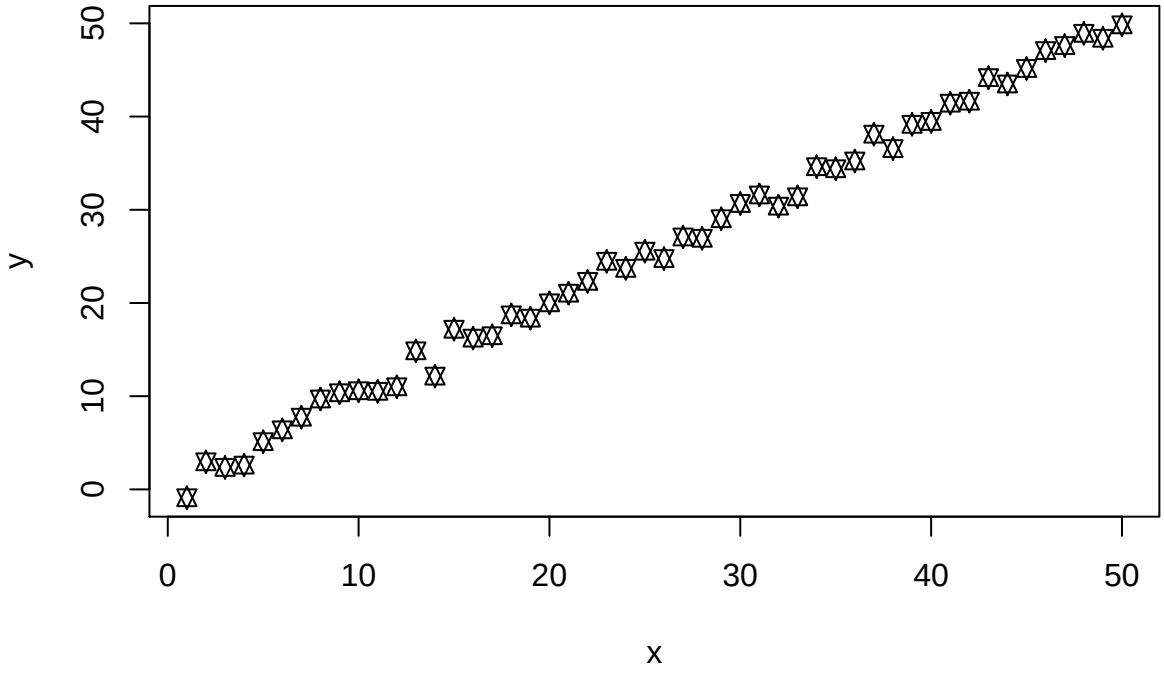

```
plot(x, y, pch=9)  # rombos con x
```



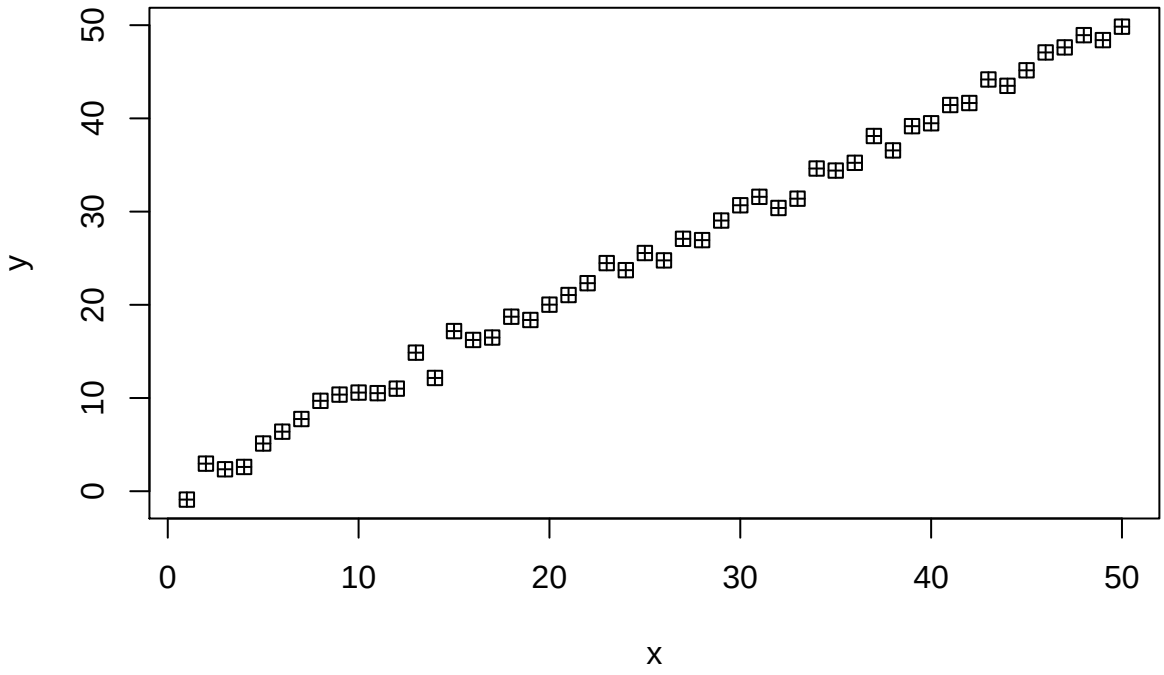

```
plot(x, y, pch=10)  # círculos con mirilla
```



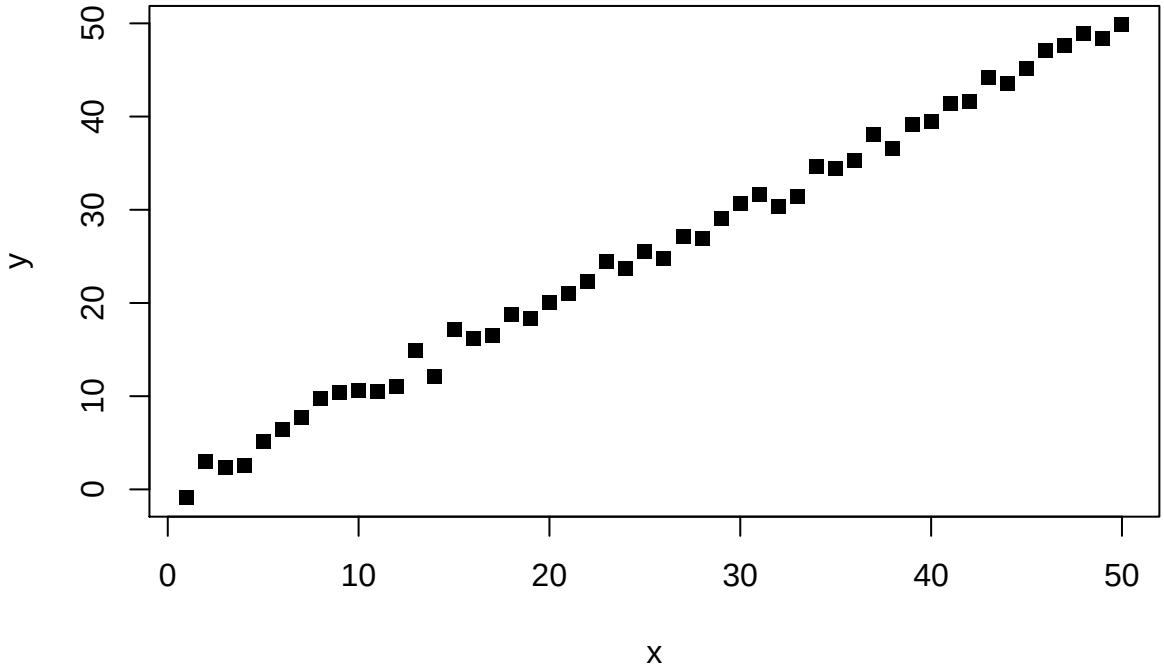

```
plot(x, y, pch=11)  # estrellas
```



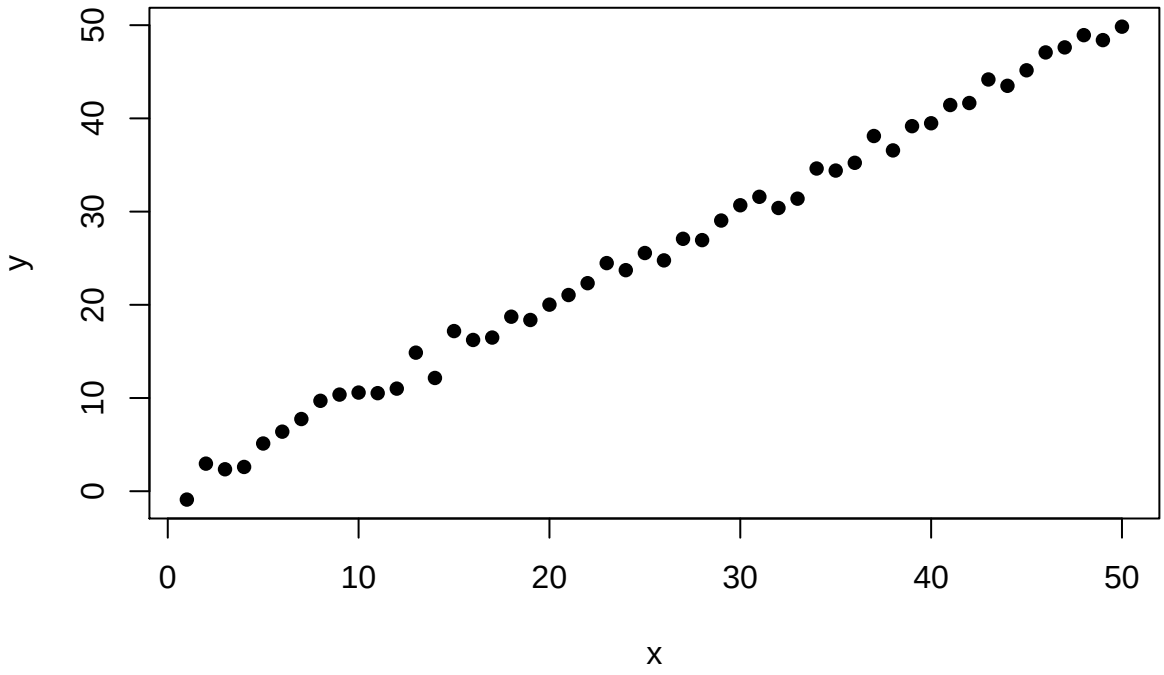

```
plot(x, y, pch=12)  # cuadrados con +
```



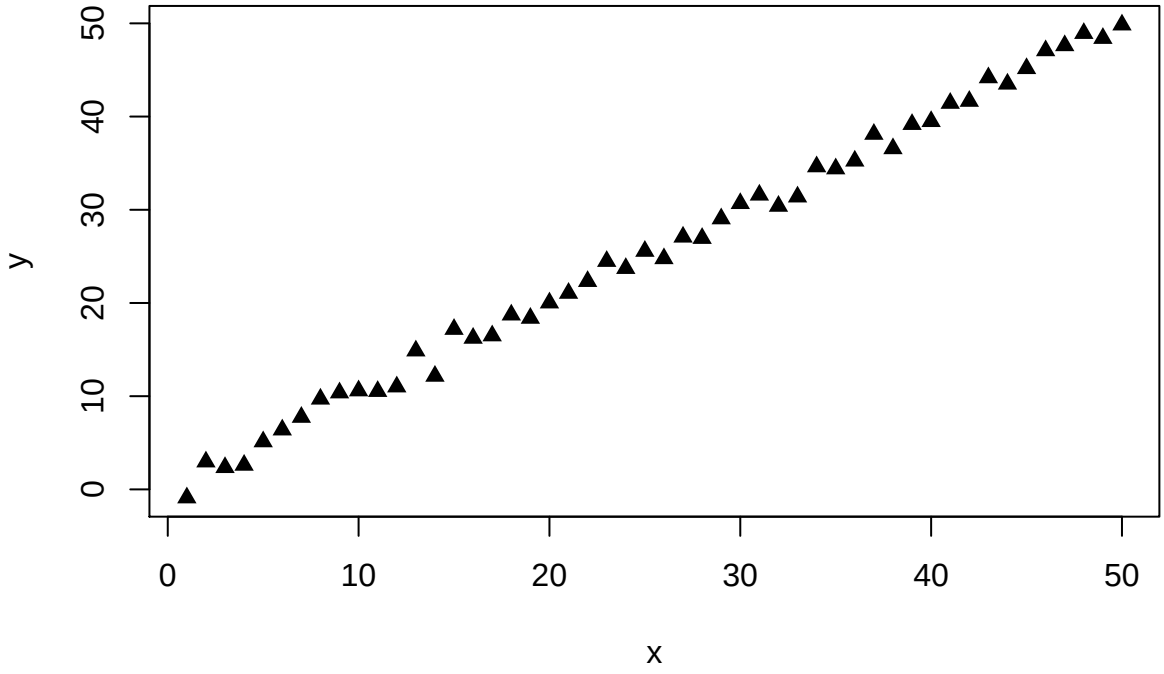

```
plot(x, y, pch=15)  # cuadrados rellenos
```



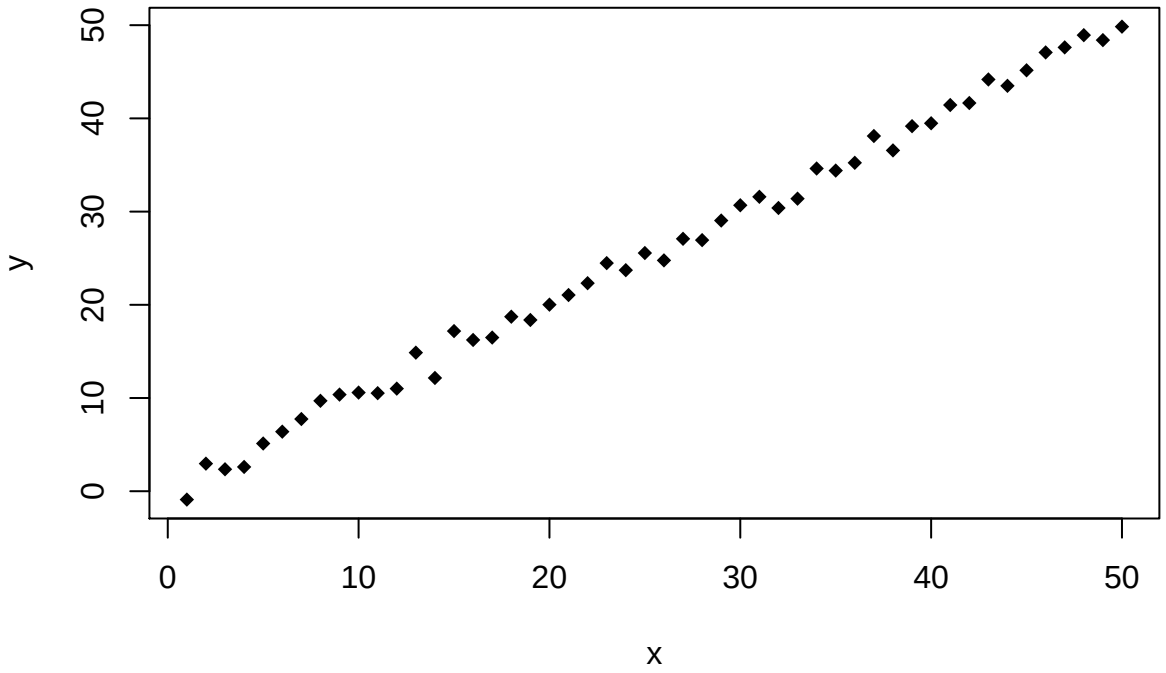

```
plot(x, y, pch=16)  # círculos rellenos
```



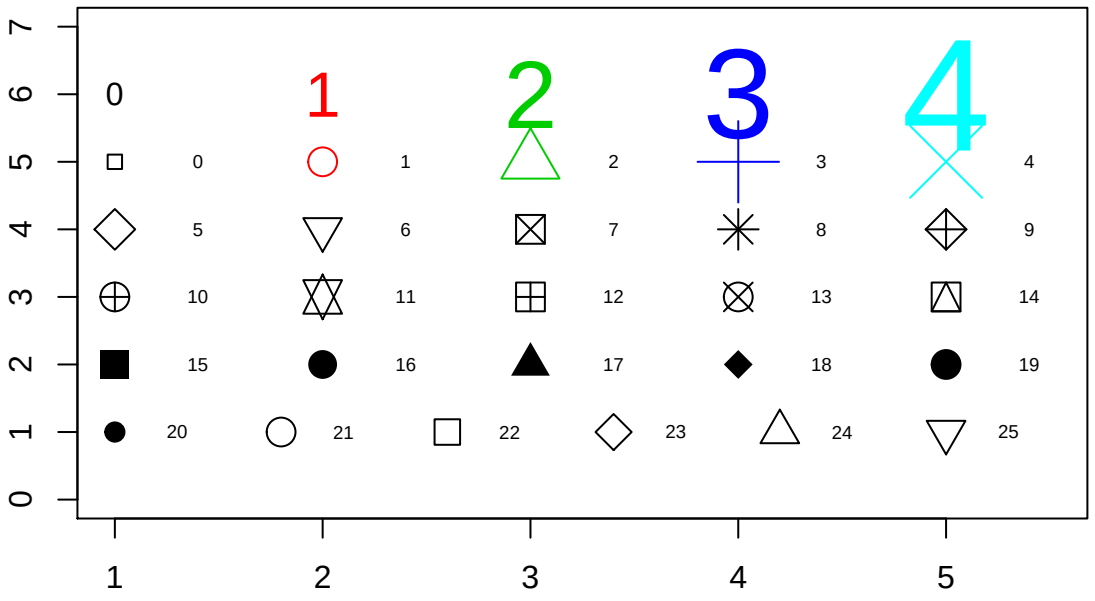

```
plot(x, y, pch=17)  # triángulos rellenos
```




```
plot(x, y, pch=18)  # rombos rellenos
```

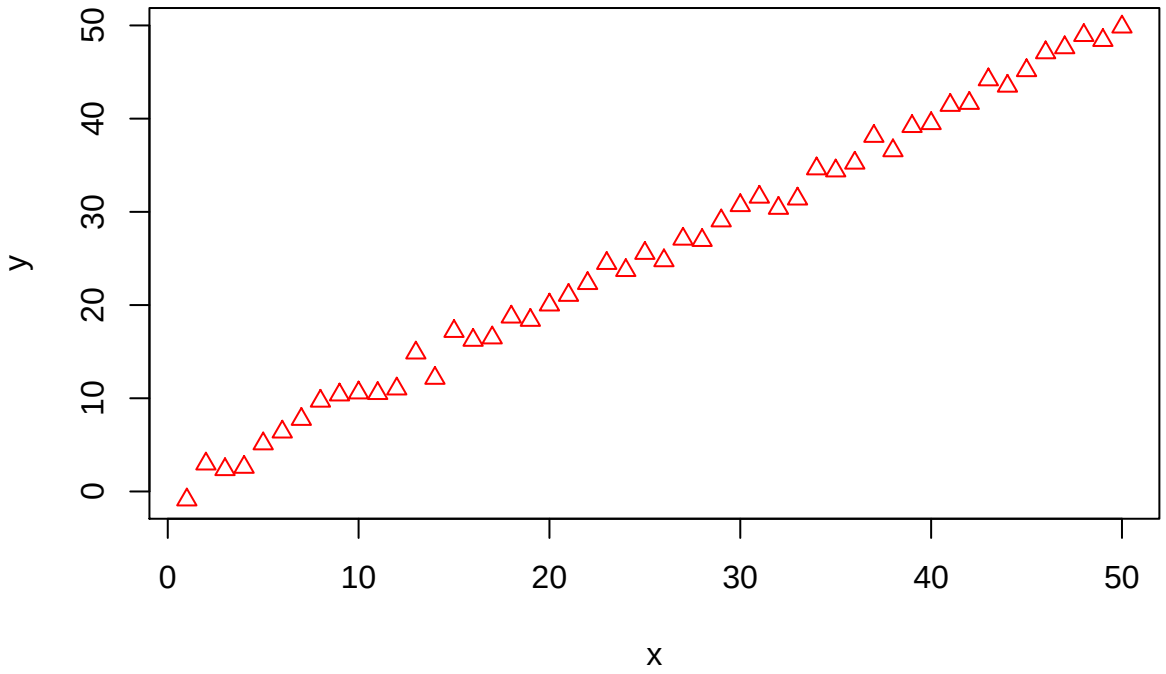



```
# Preparamos el gráfico
plot(1, 1, xlim=c(1,5.5), ylim=c(0,7), type="n", ann=FALSE)

# Dibujamos los primeros 4 incrementando tamaño y color
text(1:5, rep(6,5), labels=c(0:4), cex=1:5, col=1:5)

# Pintamos símbolos de esos primeros 4
points(1:5, rep(5,5), cex=1:5, col=1:5, pch=0:4)
text((1:5)+0.4, rep(5,5), cex=0.6, (0:4))

# Pintamos los símbolos del 5-9 con sus etiquetas
points(1:5, rep(4,5), cex=2, pch=(5:9))
text((1:5)+0.4, rep(4,5), cex=0.6, (5:9))

# Pintamos los símbolos del 10-14 con sus etiquetas
points(1:5, rep(3,5), cex=2, pch=(10:14))
text((1:5)+0.4, rep(3,5), cex=0.6, (10:14))

# Pintamos los símbolos del 15-19 con sus etiquetas
points(1:5, rep(2,5), cex=2, pch=(15:19))
text((1:5)+0.4, rep(2,5), cex=0.6, (15:19))

# Pintamos los símbolos del 20-24 con sus etiquetas
points((1:6)*0.8+0.2, rep(1,6), cex=2, pch=(20:25))
text((1:6)*0.8+0.5, rep(1,6), cex=0.6, (20:25))
```

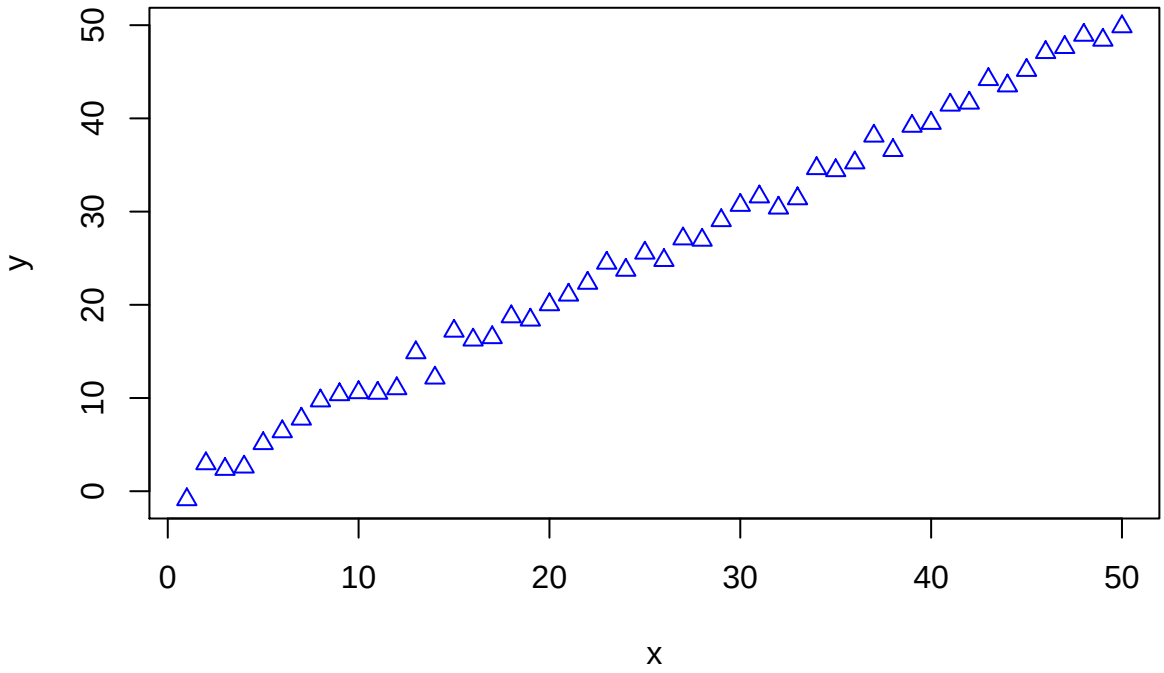


Colores en los gráficos

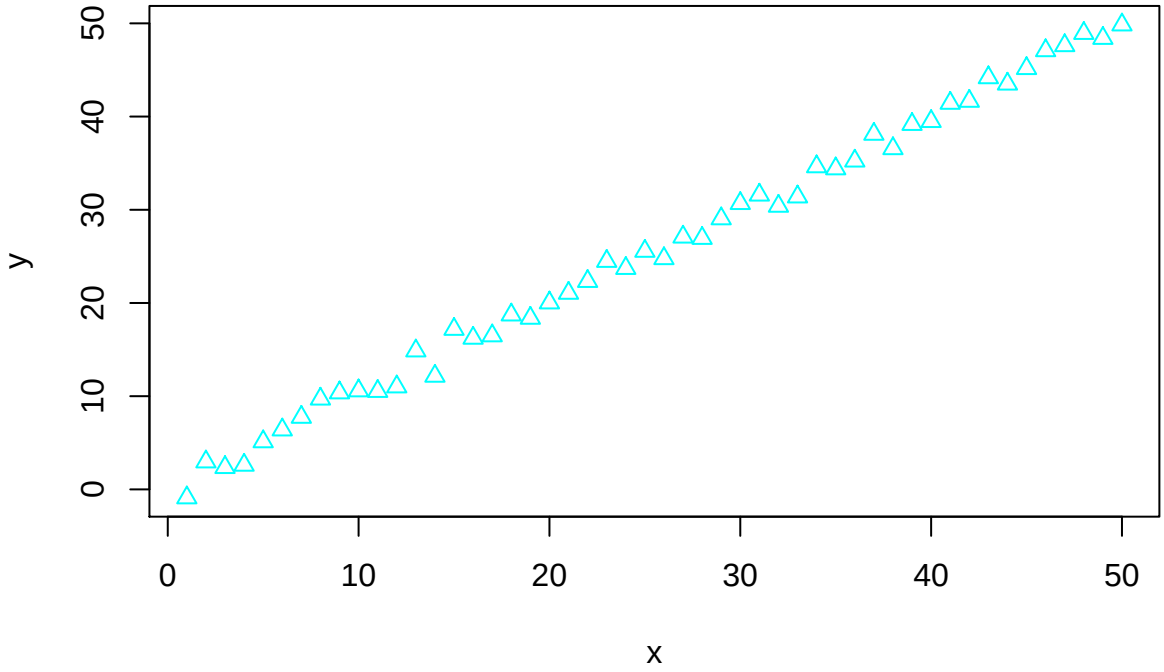
```
plot(x, y, pch=2, col="red")  # rojo
```



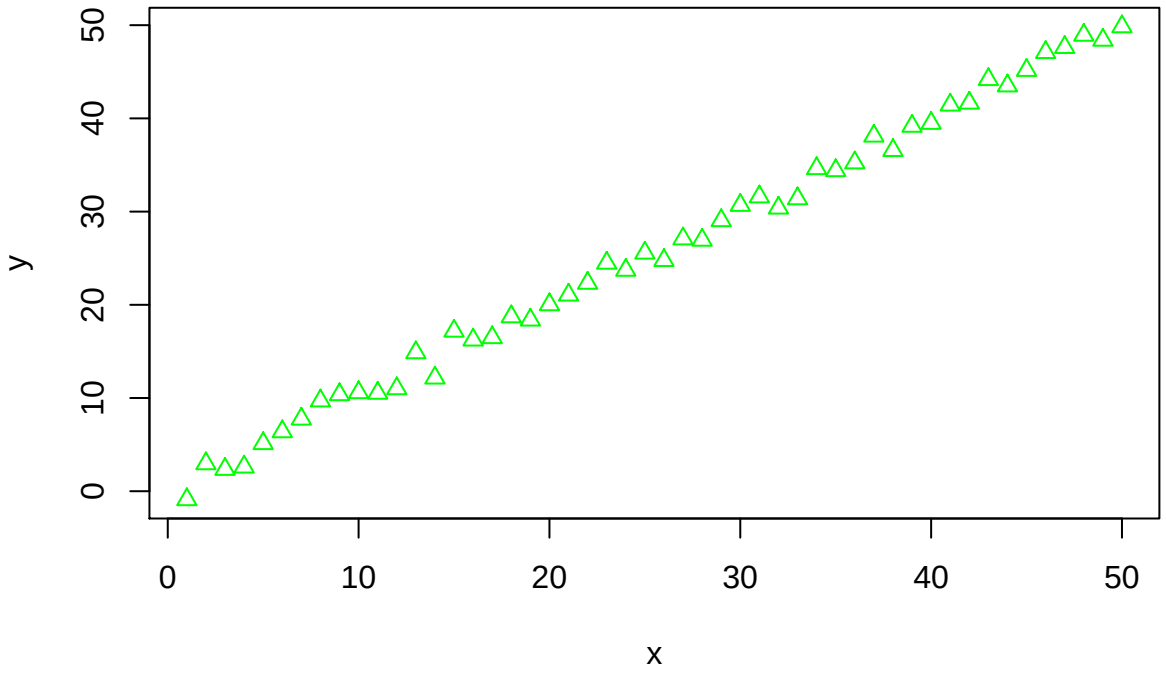

```
plot(x, y, pch=2, col="blue")  # azul
```



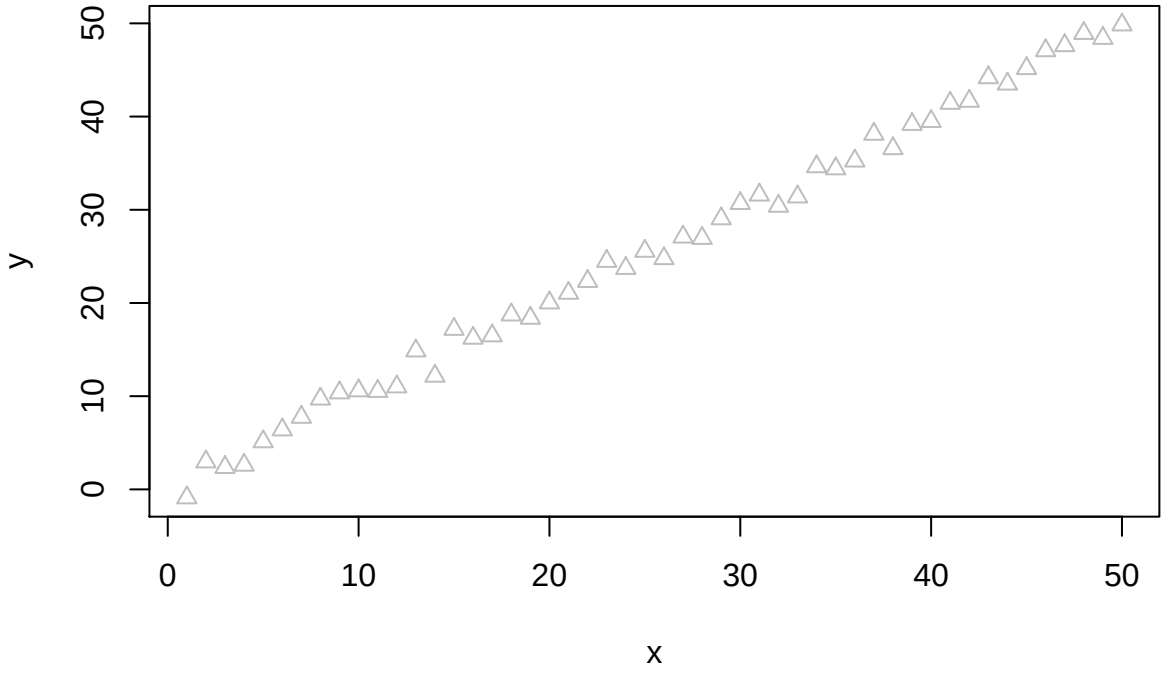

```r
plot(x, y, pch=2, col="cyan")  # celeste
```



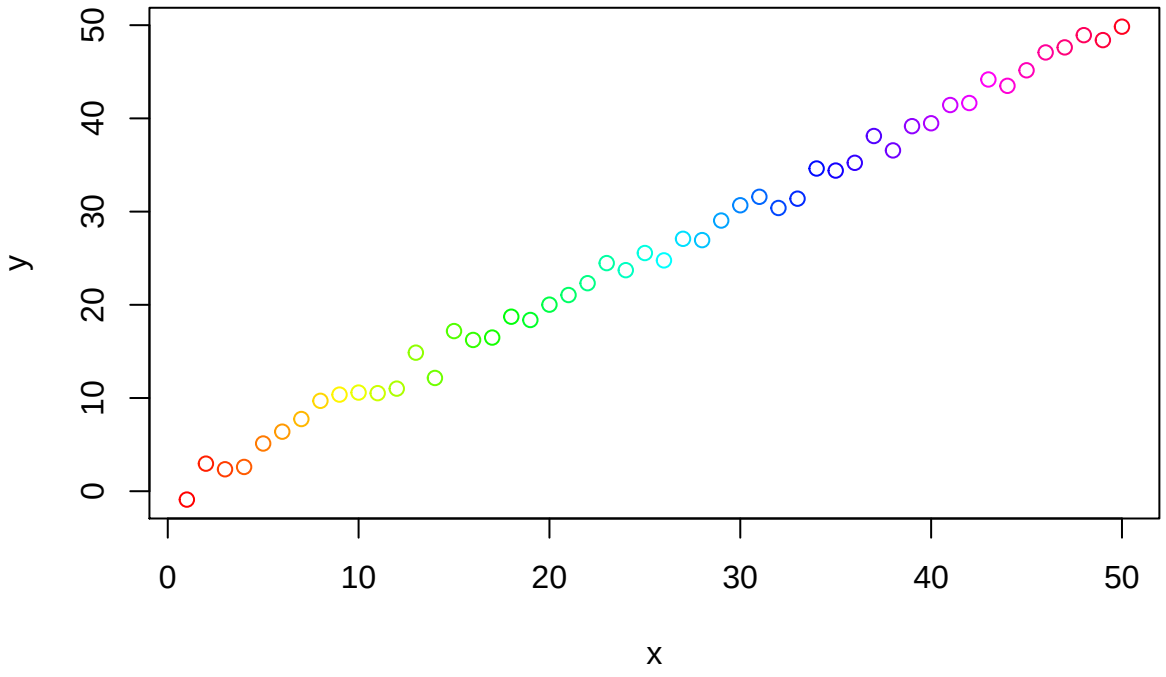

```r
plot(x, y, pch=2, col="green")  # verde
```




```
plot(x, y, pch=2, col="grey")  # gris
```



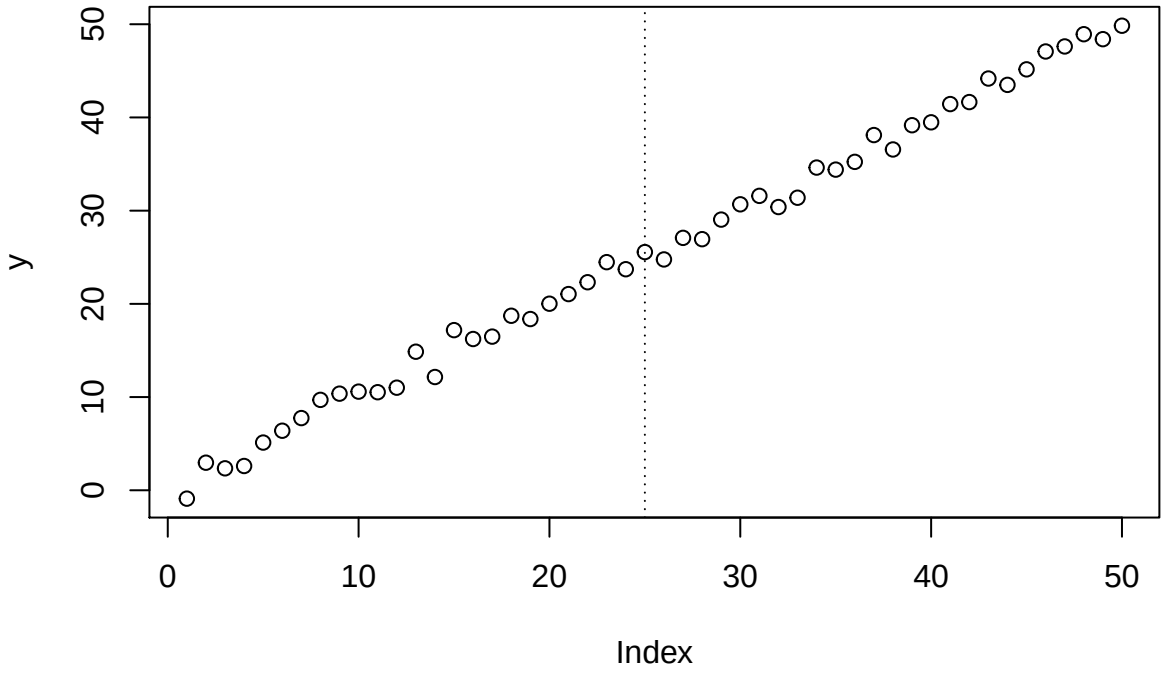

```
mis_colores <- rainbow(length(x))
plot(x, y, col=mis_colores)
```



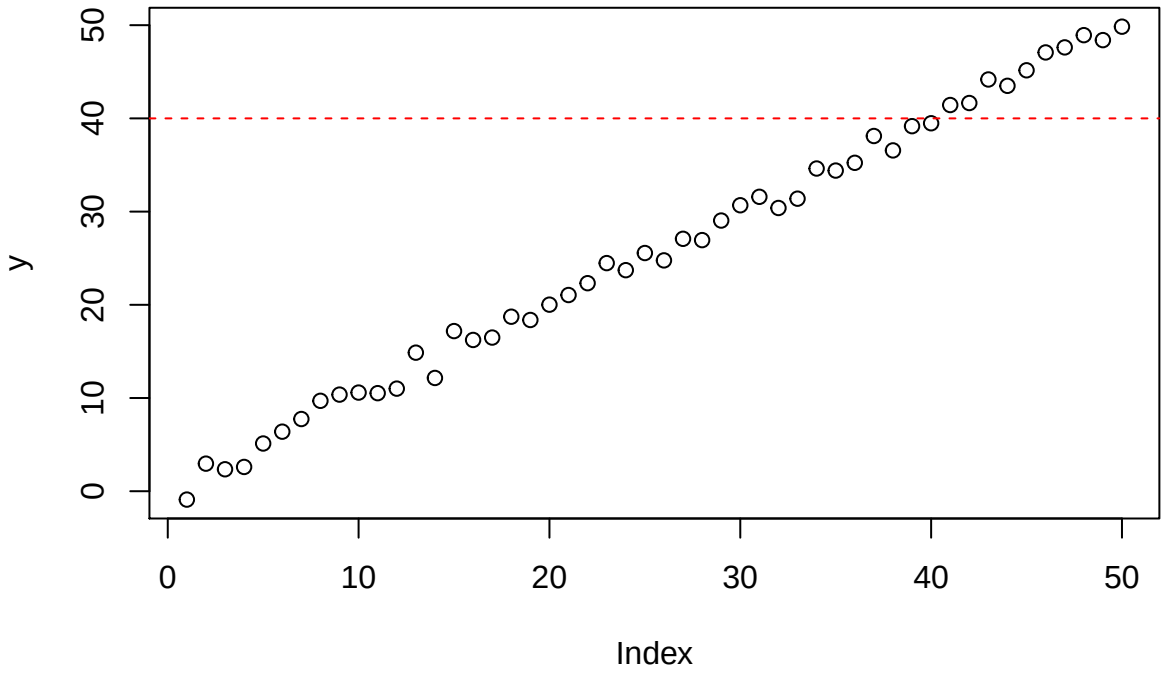
##

Agregar líneas a un gáfico

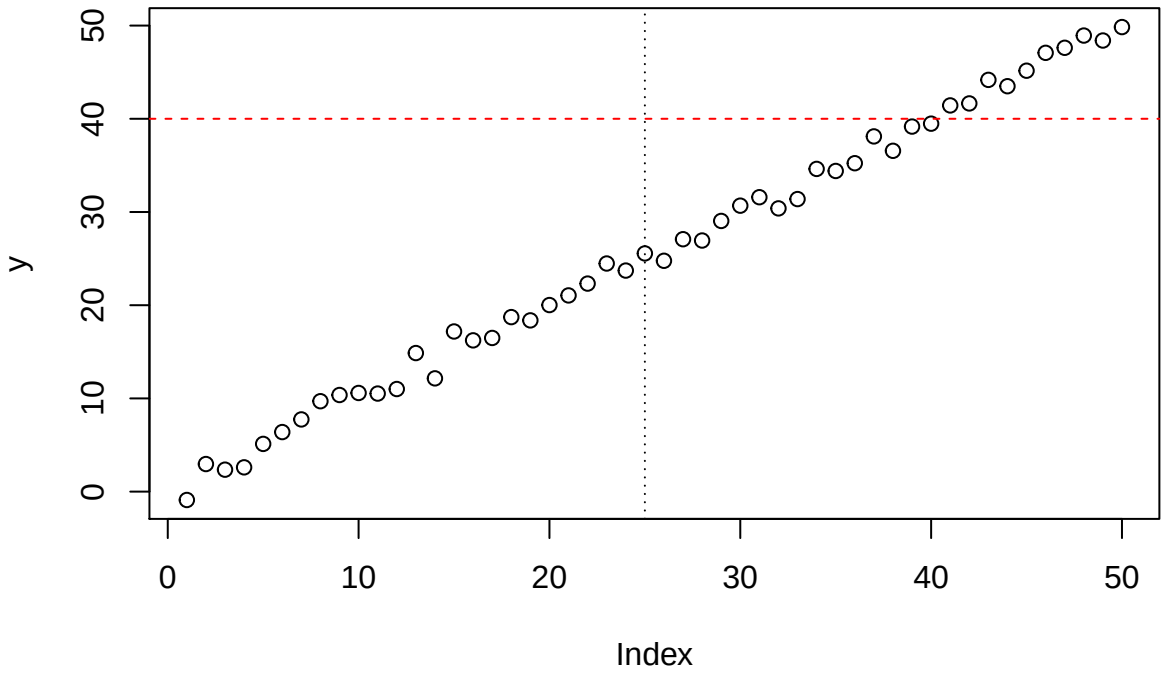
```
plot(y) ; abline(v=25, lty=3)
```



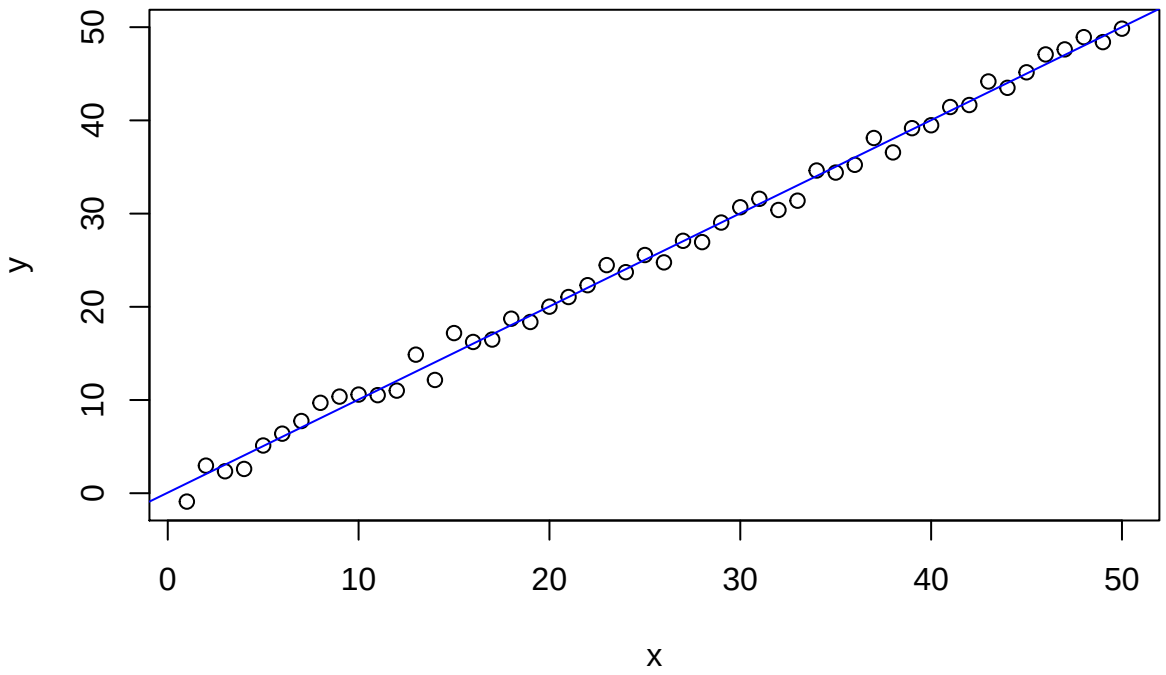

```
# Añade línea de puntos vertical

plot(y) ; abline(h=40, lty=2, col="red")
```




```
# Añade línea de rayas horizontal roja a la altura 40

plot(y) ; abline(h=40, lty=2, col="red"); abline(v=25, lty=3)
```




```
#todo junto

plot(y~x) ; abline(lm(y~x), lty=1,col="blue")
```

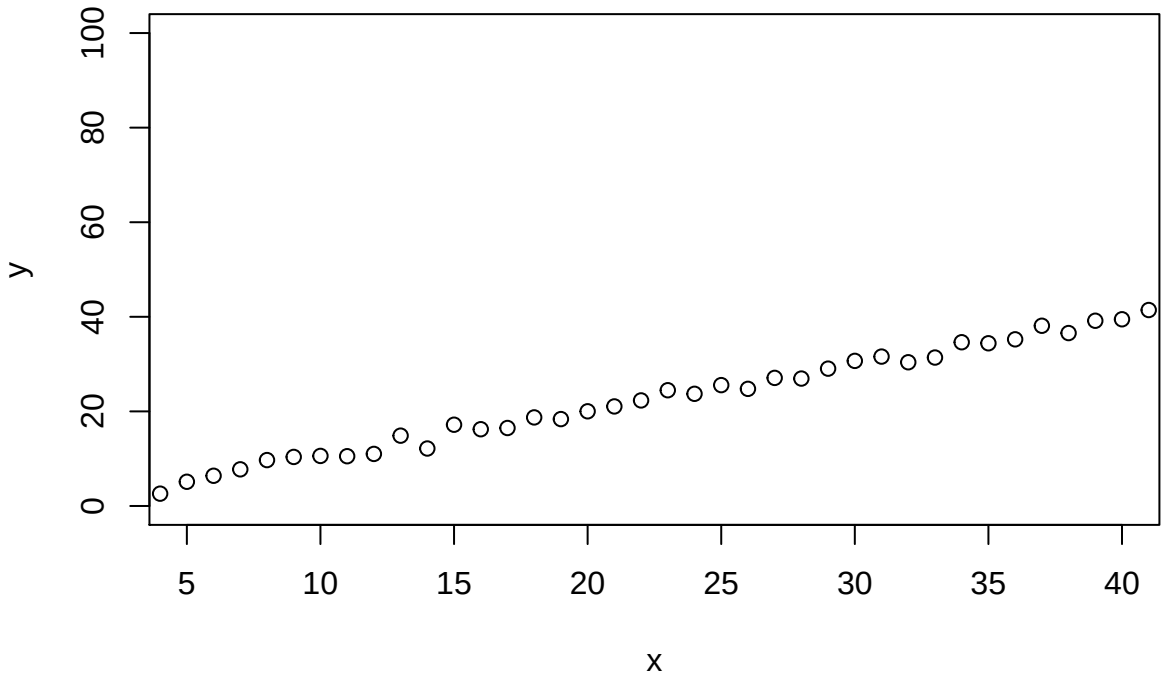



```
# línea de la recta de regresión lineal y~x
```

Controlar límites en los ejes

Podemos establecer los límites para los ejes usando los parámetros "xlim=c(Xmin,Xmax)" e "ylim=c(Ymin,Ymax)". Por ejemplo, usaremos límites distintos para x e y con el siguiente comando:

```
plot(x, y, xlim=c(5, 40), ylim=c(0, 100))
```

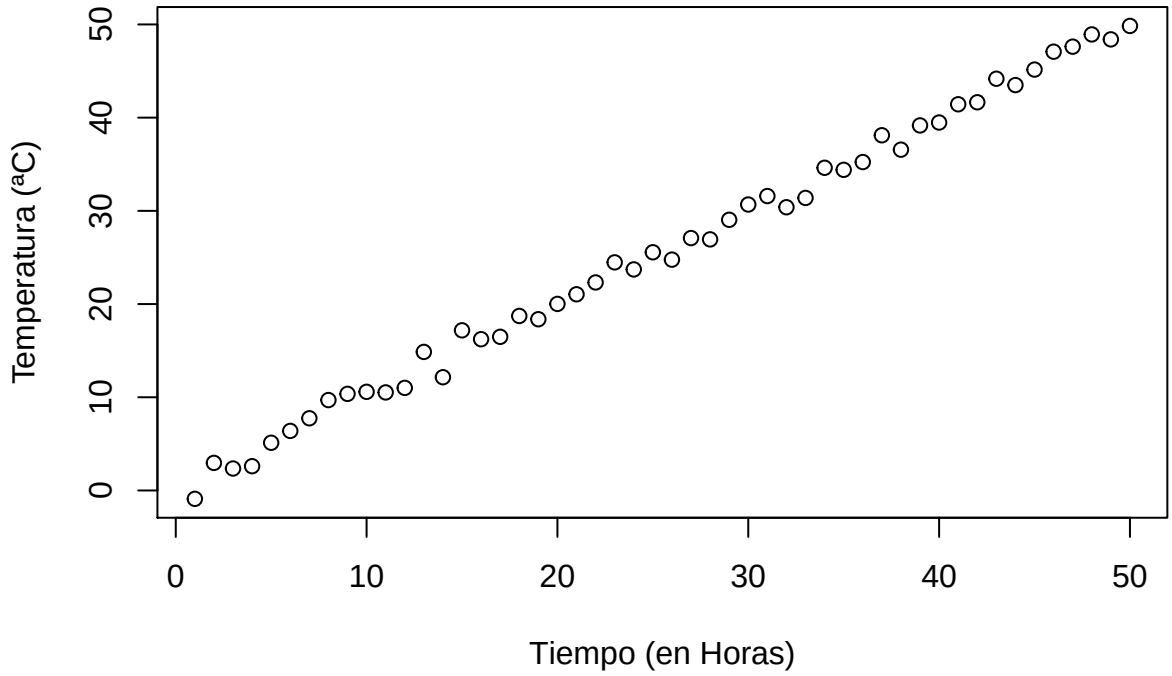



```
#límites para los ejes x e y
```

Agregar etiquetas a los ejes

Es fundamental etiquetar correctamente los ejes en los gráficos. Además es muy fácil agregar etiquetas en R. Usaremos los parámetros "xlab=" e "ylab=" de los comandos "plot" en R.

```
plot(y, xlab = "Tiempo (en Horas)", ylab = "Temperatura (ªC)" )
```



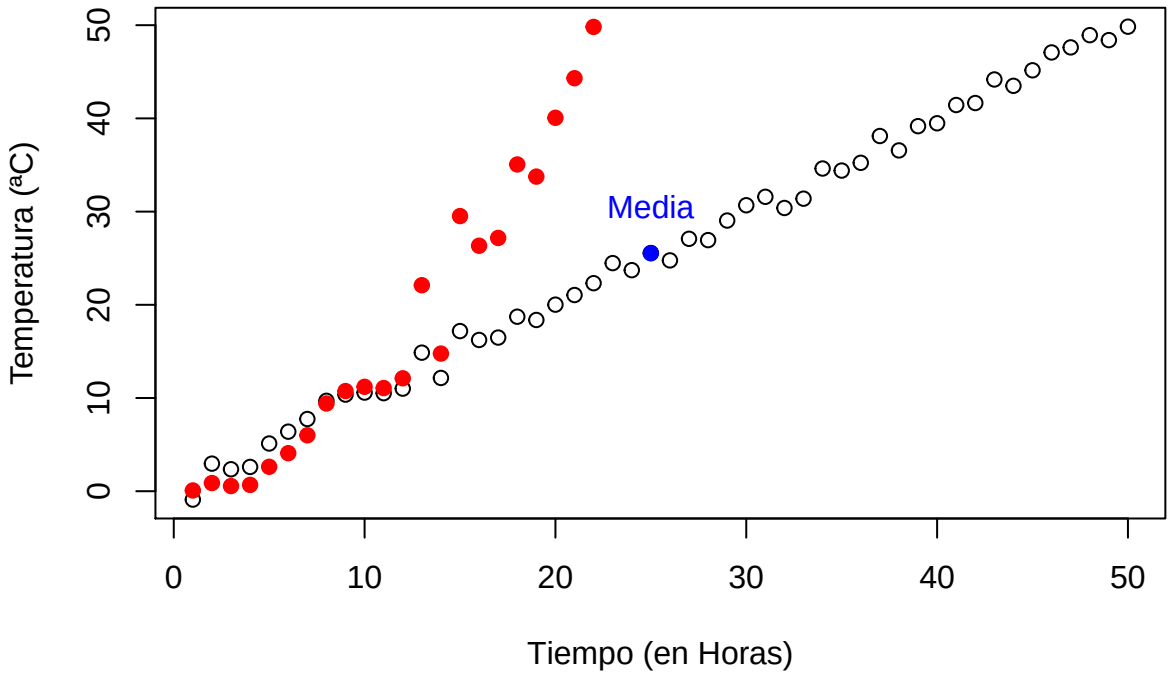

```
#etiquetas para ejes
```

Añadir puntos a un gráfico

Podemos añadir puntos a un gráfico, ya sea individualmente algún punto que se quiera resaltar en otro color, o bien una serie de puntos en otro color o formato a destacar.

Por ejemplo:

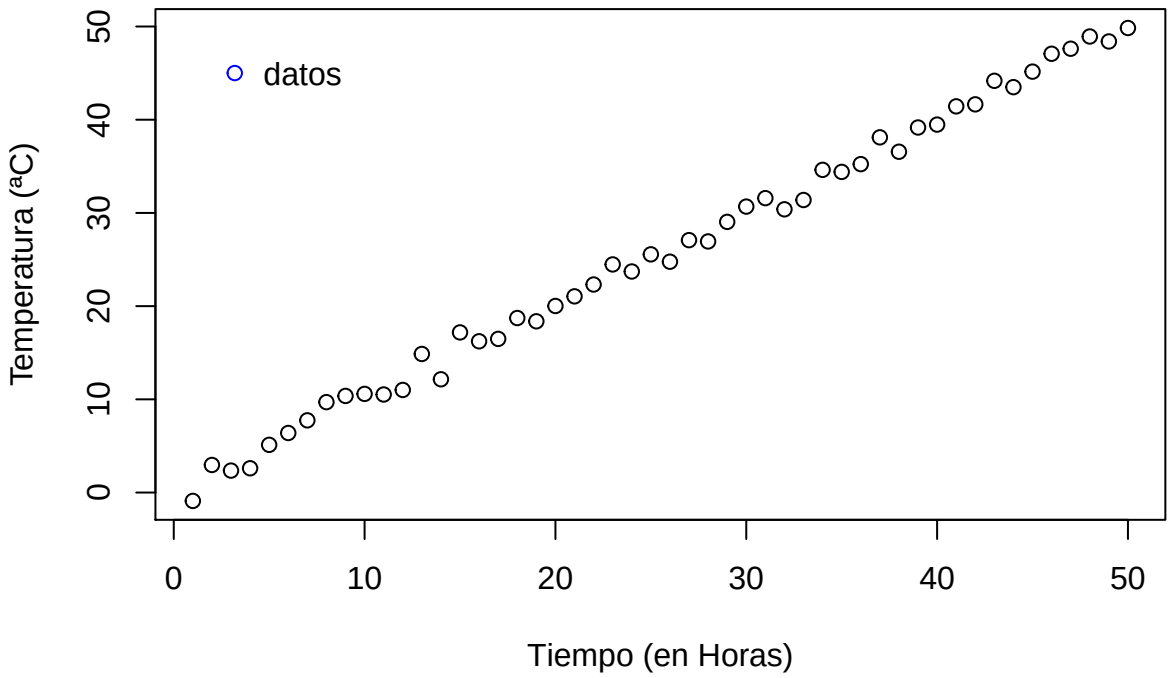
```
plot(y, xlab = "Tiempo (en Horas)", ylab = "Temperatura (ªC)" )

#añadimos puntos de nuevos datos en círculos de color rojo
z<-y*y/10
points(z, col="red", pch=19)

#Ahora añadimos un punto para resaltar algo
points(25,mean(y),col="blue",pch=19)

#Agregamos un texto
text(25,mean(y)+5,"Media",col="blue")
```



Agregar leyenda a un gráfico

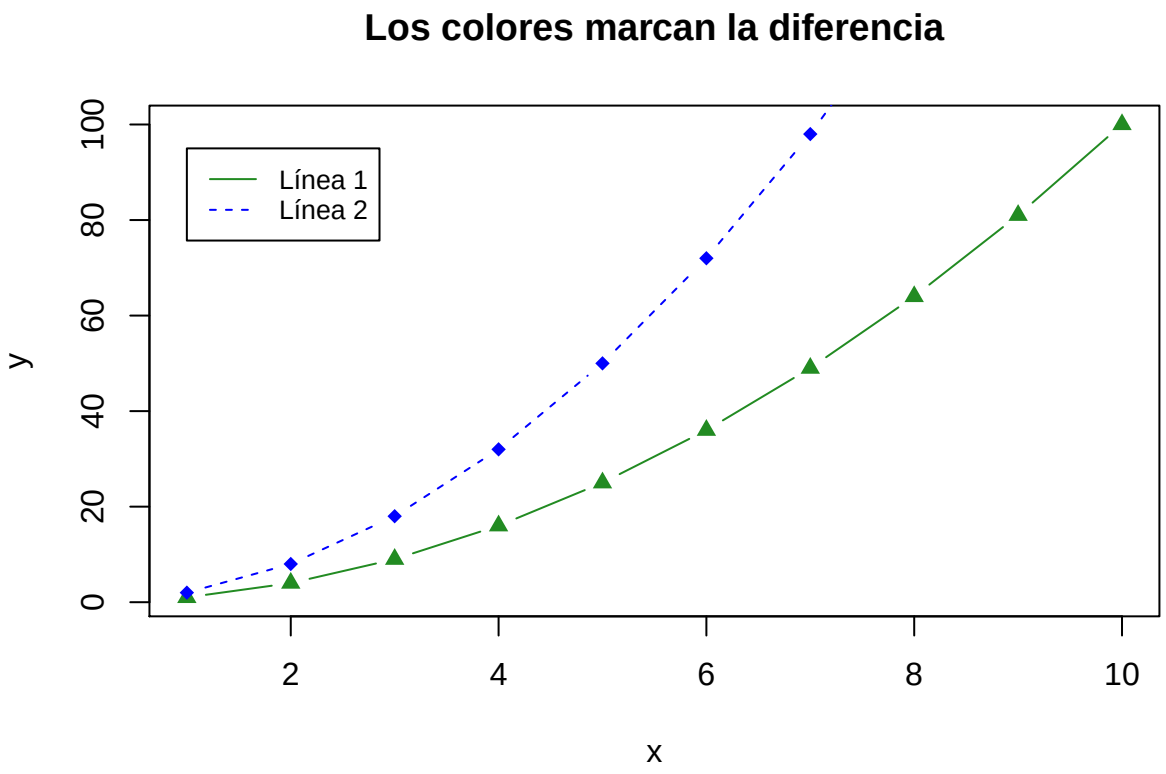
Puede resultar útil en ocasiones agregar una leyenda a un gráfico. Para ello R tiene una función específica denominada "legend()".

```
plot(y, xlab = "Tiempo (en Horas)", ylab = "Temperatura (ªC)" )  ; legend("topleft", inset=0.05, "datos
```




```
# Añade una leyenda

# Vamos a hacer un ejemplo mas complejo
```

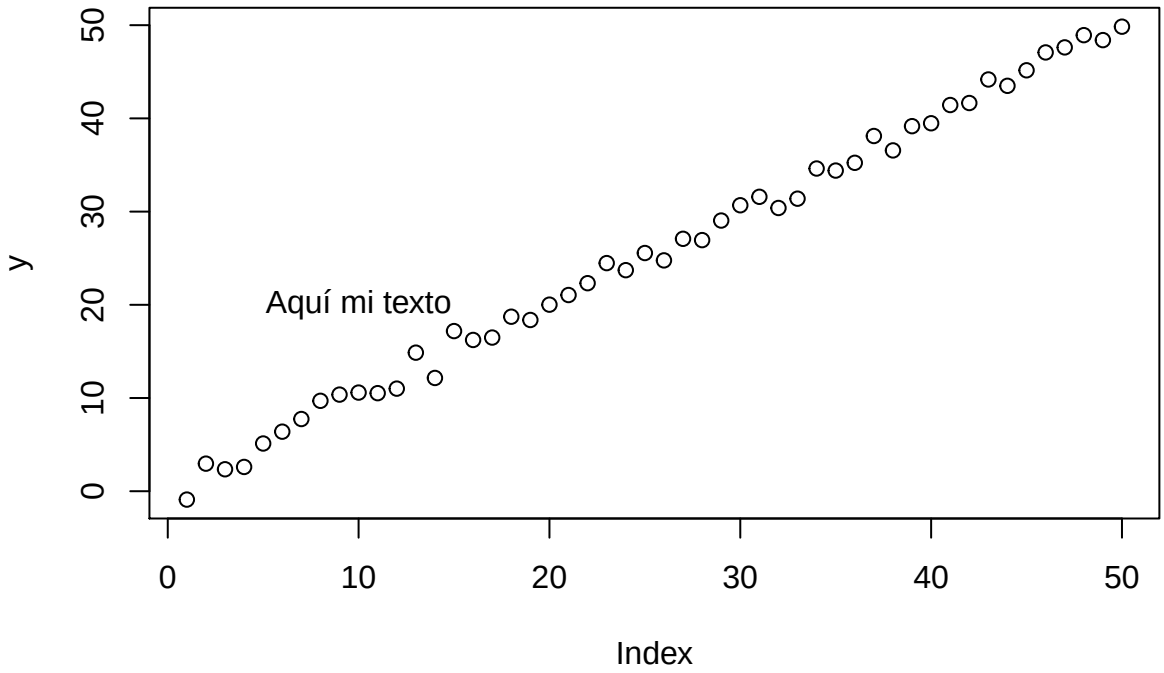
```r
x<-1:10; y1=x*x; y2=2*y1
#Dibujamos línea de puntos en verde oscuro para y1
plot(x, y1, type="b", pch=17, col="forestgreen", xlab="x", ylab="y", main = "Los colores marcan la dife
# Añadimos ahora una línea azul para y2
lines(x, y2, pch=18, col="blue", type="b", lty=2)
# Añadimos una leyenda en la posición x=1, y=95
legend(1, 95, legend=c("Línea 1", "Línea 2"),
       col=c("forestgreen", "blue"), lty=1:2, cex=0.8)
```



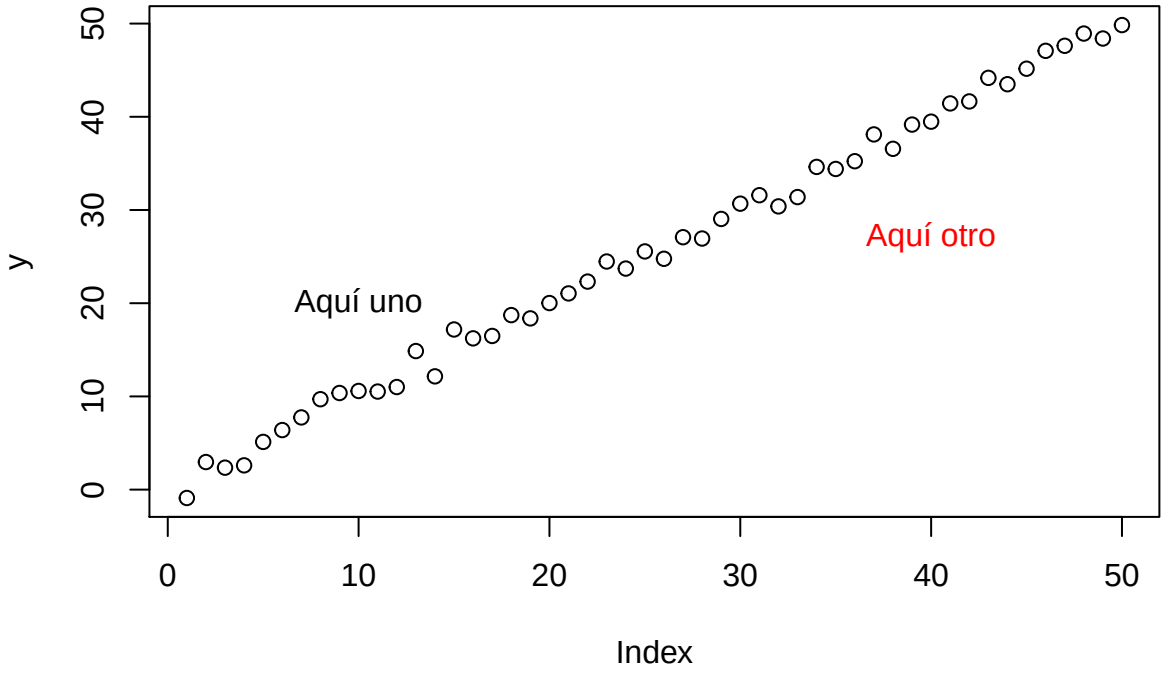
Agregar anotaciones a un gráfico

A veces es necesario agregar alguna anotación a un gráfico. En R existe una función que agrega a un gráfico una anotación, es "text('mi anotación')". Un ejemplo con el gráfico anterior.

```r
plot(y) ; text(10, 20, "Aquí mi texto")
```




```
# Anotación en gráfico

plot(y) ; text(10, 20, "Aquí uno"); text(40, 27, "Aquí otro",col = "red")
```




```
# Anotación en gráfico
```

Guardar un gráfico

Para guardar un gráfico hay varias formas en R y RStudio. Ya hemos explicado como hacerlo desde las opciones de menú y desde el panel de gráficos en RStudio, usando el botón “Export” del panel “Plots”. A

veces queremos automatizar este procedimiento e incluirlo en un guión de comandos para que se guarde de manera automática cada vez que se ejecute el guión.

Para poder guardar un gráfico en un fichero tenemos varios comandos que hacen lo mismo pero guardan en distinto formato: png, jpeg, pdf, etc.

El proceso tiene tres pasos:

1. abrir un contenedor del futuro gráfico
2. hacer el gráfico

3. cerrar el contenedor con el gráfico generado

Por ejemplo para guardar el gráfico en un fichero PNG en nuestra carpeta actual (la que sea el directorio de trabajo actual):

```
png("migrafico.png")   #Preparamos un fichero para guardar
hist(y)                #Dibujamos el histograma de y
dev.off()              #Se cierra/guarda en el fichero elegido
```

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

Ahora guardaremos en formato JPEG:

```
jpeg("migrafico.jpeg")   #Preparamos un fichero para guardar
hist(y)                  #Dibujamos el histograma de y
dev.off()                #Se cierra/guarda en el fichero elegido
```

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

Como en todos los casos en que presentamos funciones o parámetros de funciones, recomendamos buscar en la ayuda de R para más detalles acerca de más características que pueden ser de utilidad para personalizar gráficos.