

Introducción al R

Procedimientos Gráficos

true

17 de marzo de 2018

Contents

Procedimientos Gráficos	1
Funciones Gráficas	1
Funciones gráficas de alto nivel	1
Funciones gráficas de bajo nivel	6
Práctica	10

Procedimientos Gráficos

- Una de las grandes potencialidades en R es la calidad gráfica. Su potencialidad es del mismo nivel e incluso superior al de muchos software comerciales.
- Resultaría un curso exclusivo de graficación en R para poder dar a conocer todas las posibilidades de graficación en la herramienta. Para ilustrar algunas posibilidades:

```
demo(graphics)
```

Funciones Gráficas

- El resultado de una función gráfica no puede ser asignado a un objeto sino que es enviado a un dispositivo gráfico. Un dispositivo gráfico es una ventana gráfica o un archivo.
- Existen dos tipos de funciones gráficas:
 - Funciones de *gráficas de alto nivel*: Crean una nueva gráfica
 - Funciones de *gráficas de bajo nivel*: agregan elementos a una gráfica existente
- Las gráficas se producen con respecto a *parámetros gráficos* que están definidos por defecto y pueden ser modificados con la función **par**.

Funciones gráficas de alto nivel

Función	Descripción
<code>plot(x)</code>	graficar los valores de x (en el eje y) ordenados en el eje x
<code>plot(x,y)</code>	gráfico bivariado de x (en el eje x) y y (en el eje y)
<code>pie(x)</code>	gráfico circular tipo <i>pie</i>
<code>boxplot(x)</code>	Gráfico de caja y bigotes
<code>hist(x)</code>	histograma de las frecuencias de x
<code>barplot(x)</code>	histograma de los valores de x

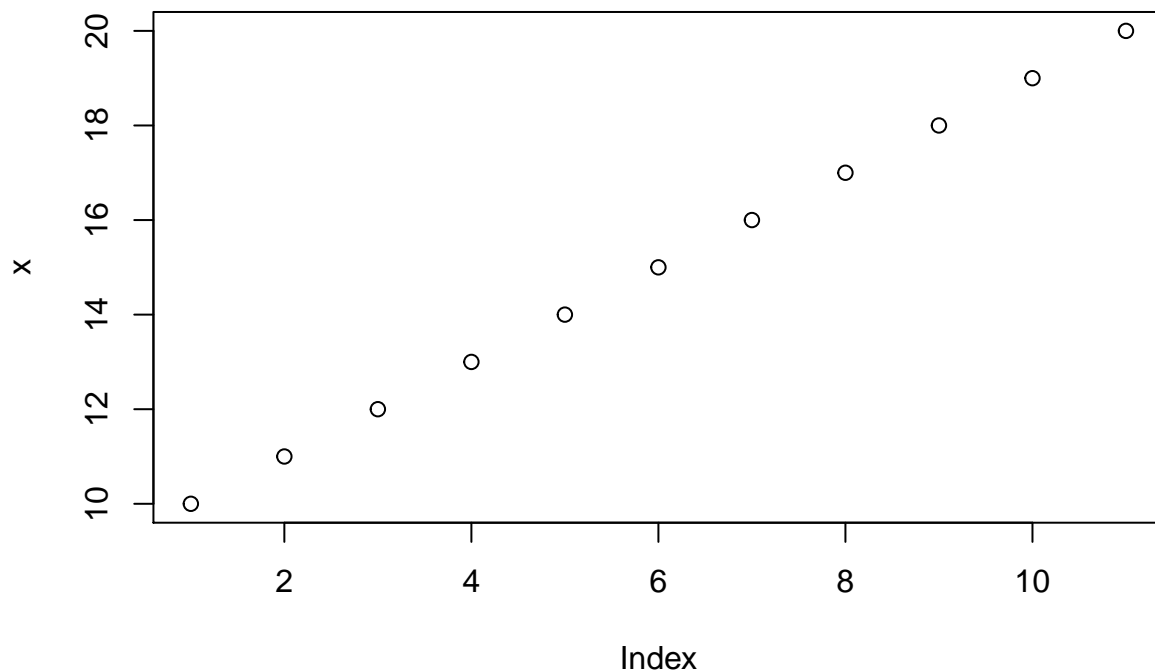
- Las opciones de cada unas de las funciones se pueden ver en la ayuda de R.

- Las principales son:

Opción	Descripción
<code>add=FALSE</code>	Es TRUE superpone el gráfico en el ya existente (si existe)
<code>axes=TRUE</code>	Es FALSE no dibuja los ejes ni la caja del gráfico
<code>type="p"</code>	especifica el tipo de gráfico; "p": puntos, "l": líneas, "h": líneas verticales, "s": escaleras, los datos se representan como la parte superior de las líneas verticales, entre otros
<code>xlim=</code>	especifica los límites inferiores y superiores de los ejes; por ejemplo con <code>xlim=c(1, 10)</code> o <code>xlim=range(x)</code>
<code>main=</code>	Título principal; debe ser de tipo carácter
<code>sub=</code>	sub-título (escrito en una letra más pequeña)

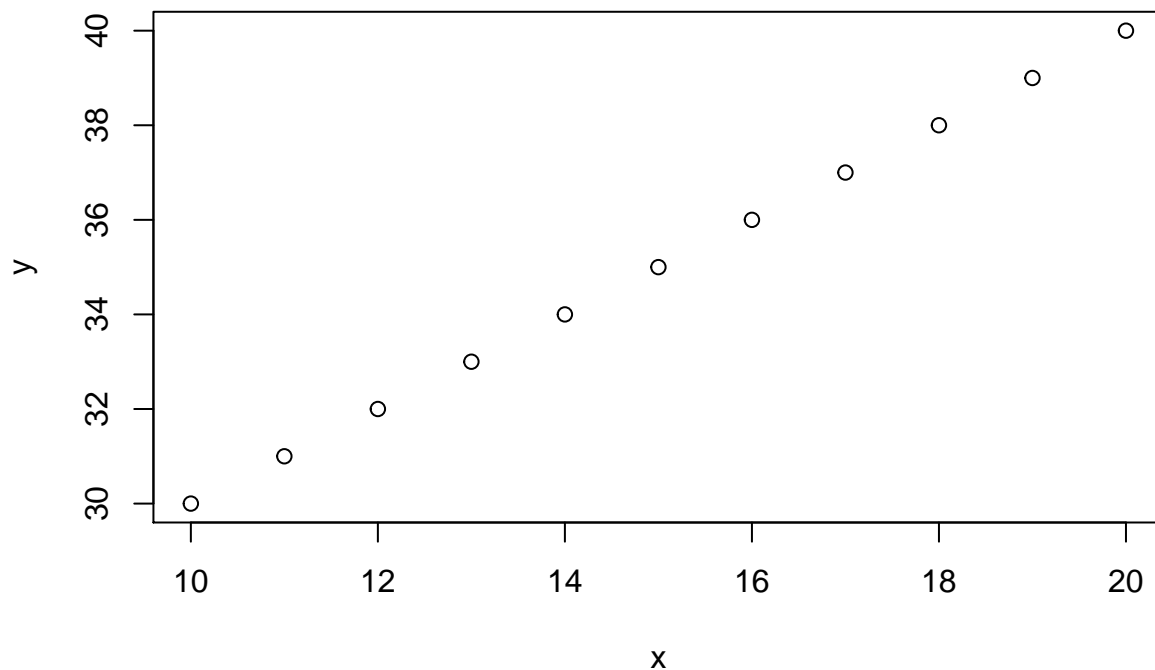
La función `plot(x)`

```
x <- seq(10,20,1)
plot(x)
```



La función `plot(x,y)`

```
y <- seq(30,40,1)
plot(x,y)
```



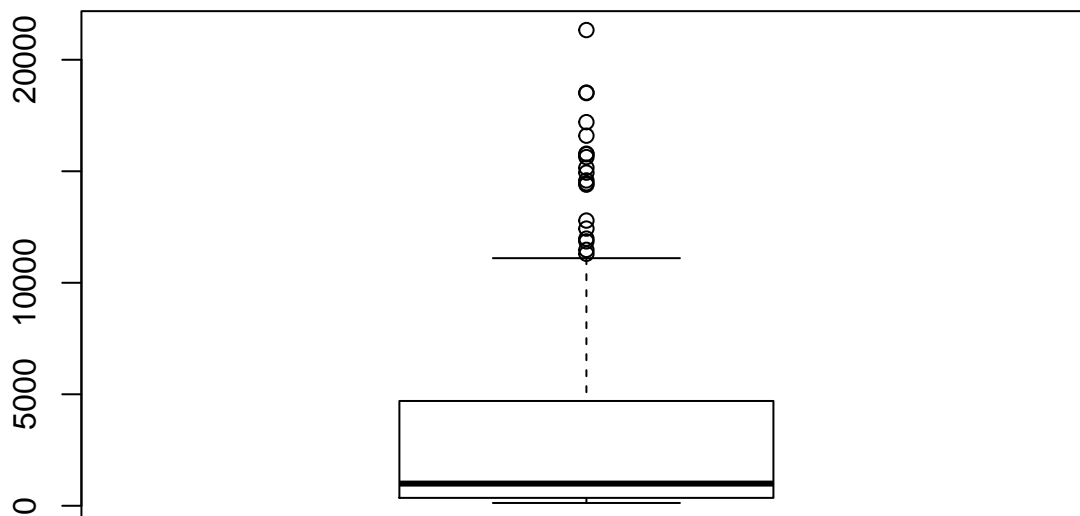
Boxplot

- También se le llama gráfico de caja y bigotes, refleja gráficamente el resumen de estadísticas principales (Min, Q1, Mediana, Q3, Max, Outlayers). Importa los datos `Mundo.csv` y...

```
boxplot(PNB_PC)
boxplot(log(PNB_PC))
```

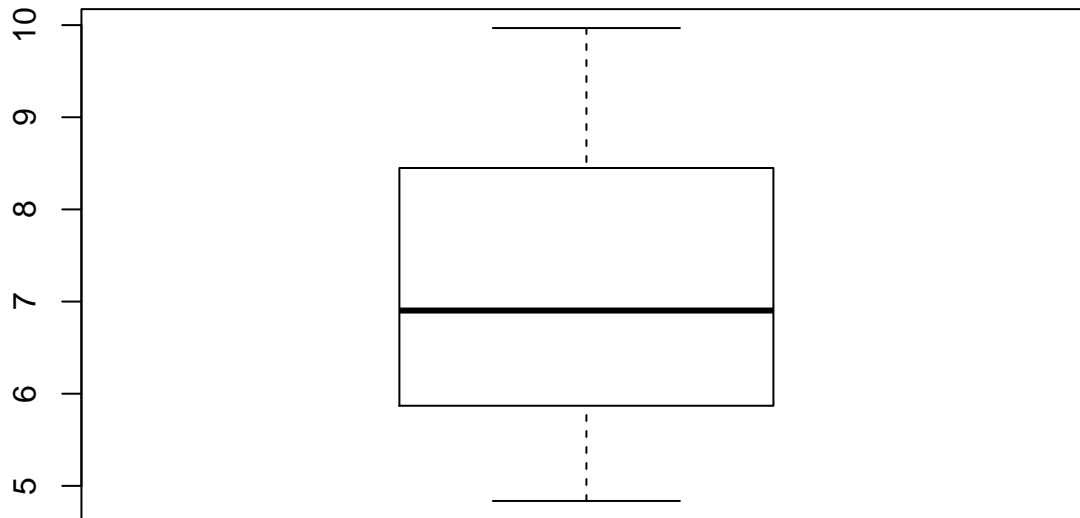
- Exácto! Es muy probable que hayas obtenido un error (**¿Por qué?**). Hay dos formas de solucionarlo. Una de las formas sería:

```
boxplot(datos$PNB_PC)
```



- Mmmm, algo anda mal, la variable no se puede apreciar muy bien. Tratemos visualizarla haciendo una transformación logarítmica:

```
boxplot(log(datos$PNB_PC))
```



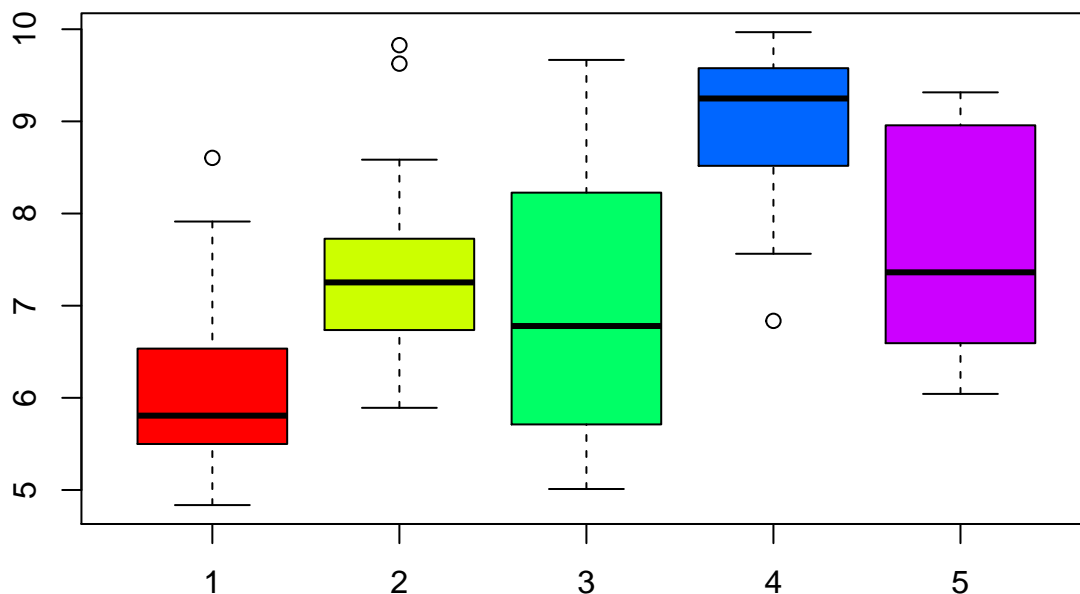
En general, si al aplicar el boxplot sobre una variable se tiene problemas para visualizarla, se debe usar el logaritmo de la variable para una mejor lectura de la variable.

- Otra forma de solucionar el problema anterior es usando el comando `attach`. Esta función es utilizada para poder *ingresar* a las variables de un `data frame` sin el operador `$`. Así

```
attach(datos)
```

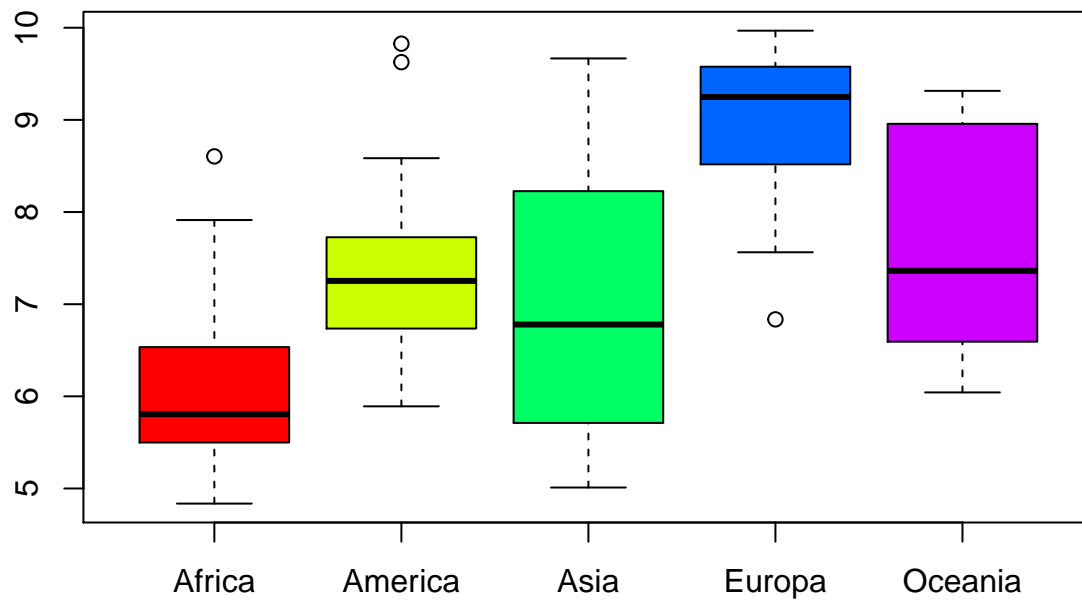
- Realicemos varios boxplot a la vez.

```
boxplot(log(PNB_PC)~REGION, col=rainbow(5))
```



- Agreguemos etiquetas:

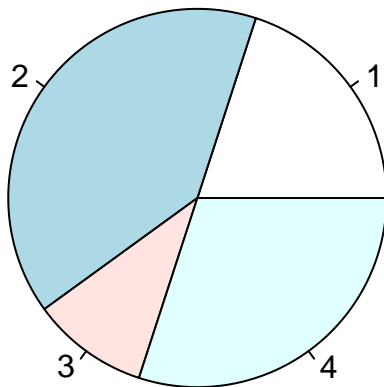
```
fregion <- factor(REGION, labels=c("Africa", "America", "Asia", "Europa", "Oceania"))  
boxplot(log(PNB_PC)~fregion, col=rainbow(5))
```



Pie

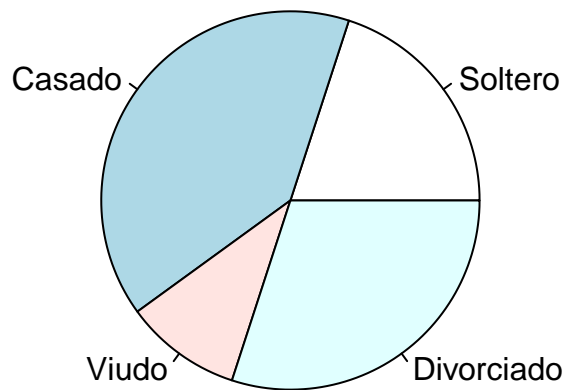
- Realiza un gráfico de *PIE* de los datos (tiene sentido cuando son datos de suma 100)

```
z.pie <- c(20,40,10,30)
pie(z.pie)
```



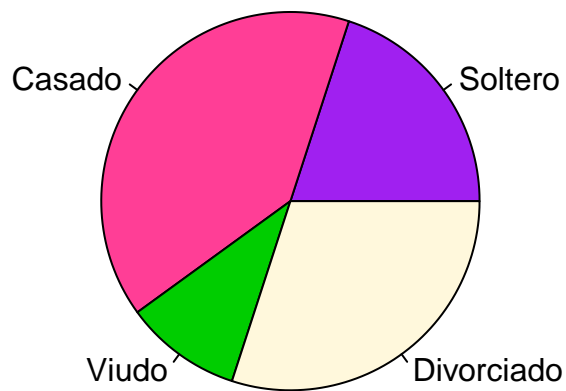
- Con etiquetas

```
names(z.pie) <- c("Soltero", "Casado", "Viudo", "Divorciado")
pie(z.pie)
```



- Con colores elegidos (más opciones de colores puedes encontrarlos aquí)

```
pie(z.pie, col = c("purple", "violetred1", "green3", "cornsilk"))
```



- Juguemos un poco con las opciones:
 - Realizar un `plot` de x e y (ya creados) con las siguientes opciones:
 - Título: Gráficas en R
 - Subtítulo: Centro de Estudios Fiscales
 - Etiquetas en ejes: "Eje X", "Eje Y" Respectivamente
 - Gráfico tipo escalera
 - Color Azul

Funciones gráficas de bajo nivel

Función	Descripción
<code>points(x, y)</code>	Agrega puntos
<code>lines(x,y)</code>	Mismo que <code>points</code> pero con líneas
<code>segments(x0, y0, x1, y1)</code>	Agrega una línea desde el punto (x_0, y_0) hasta el punto (x_1, y_1)
<code>abline(a,b)</code>	Agrega una línea con pendiente b e intercepto a

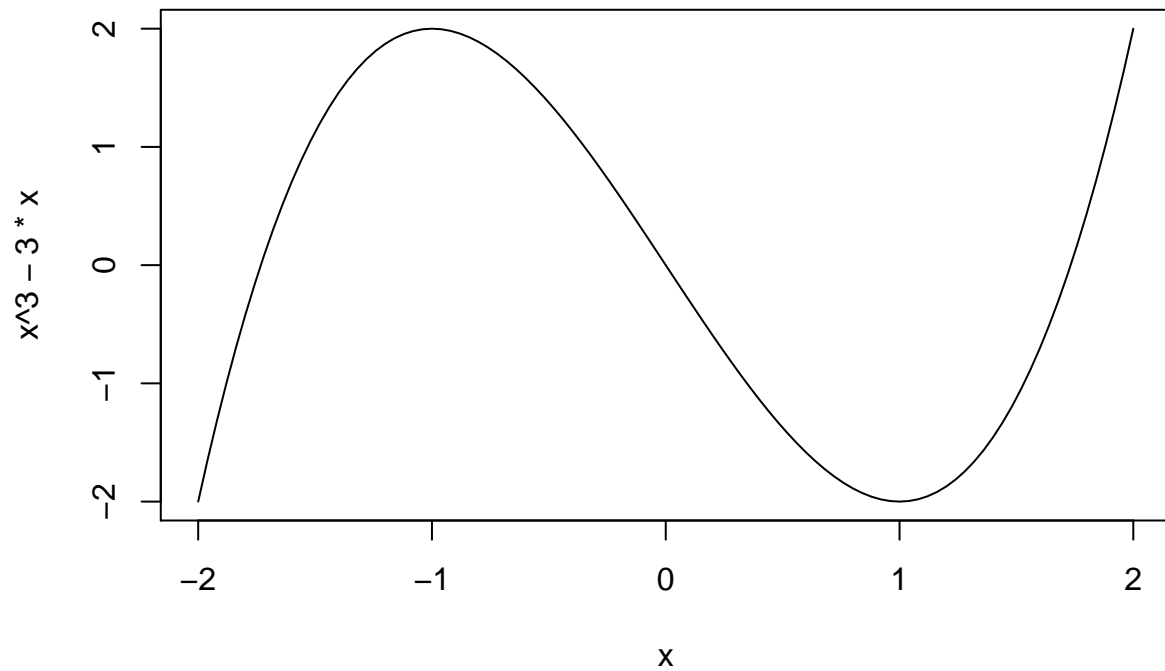
Curve

Sirve para graficar funciones, el formato es

```
curve(expr, from, to, add=FALSE,...)
```

Por ejemplo:

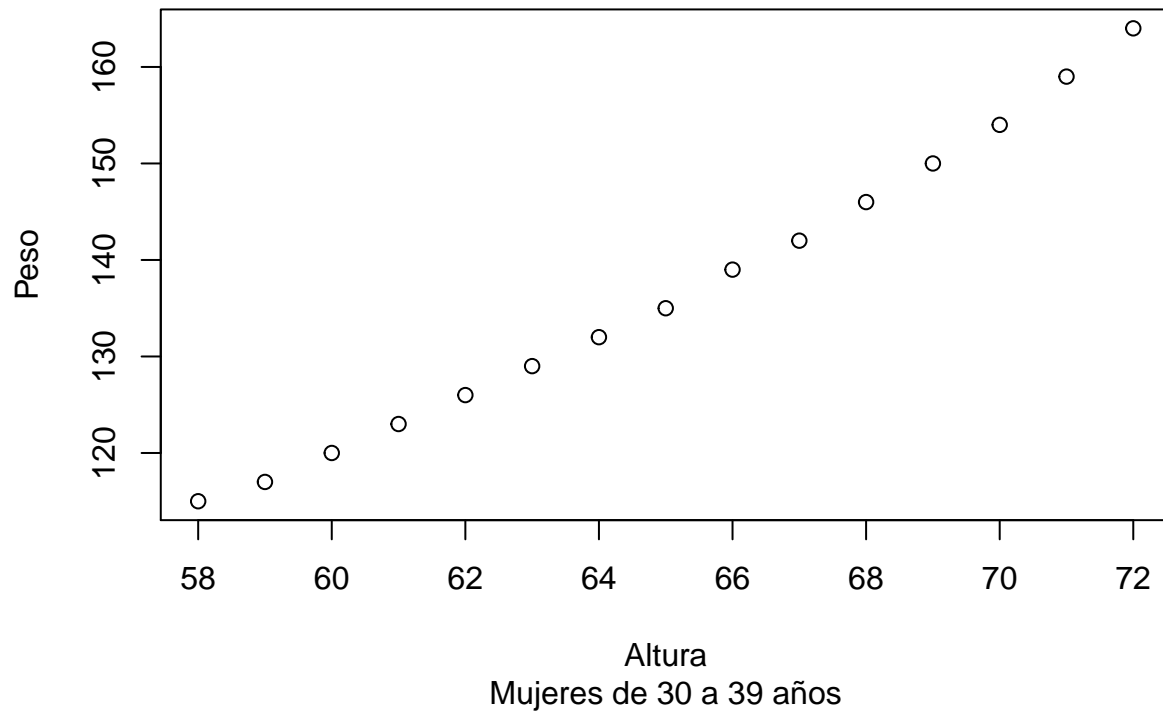
```
curve(x^3-3*x,-2,2)
```



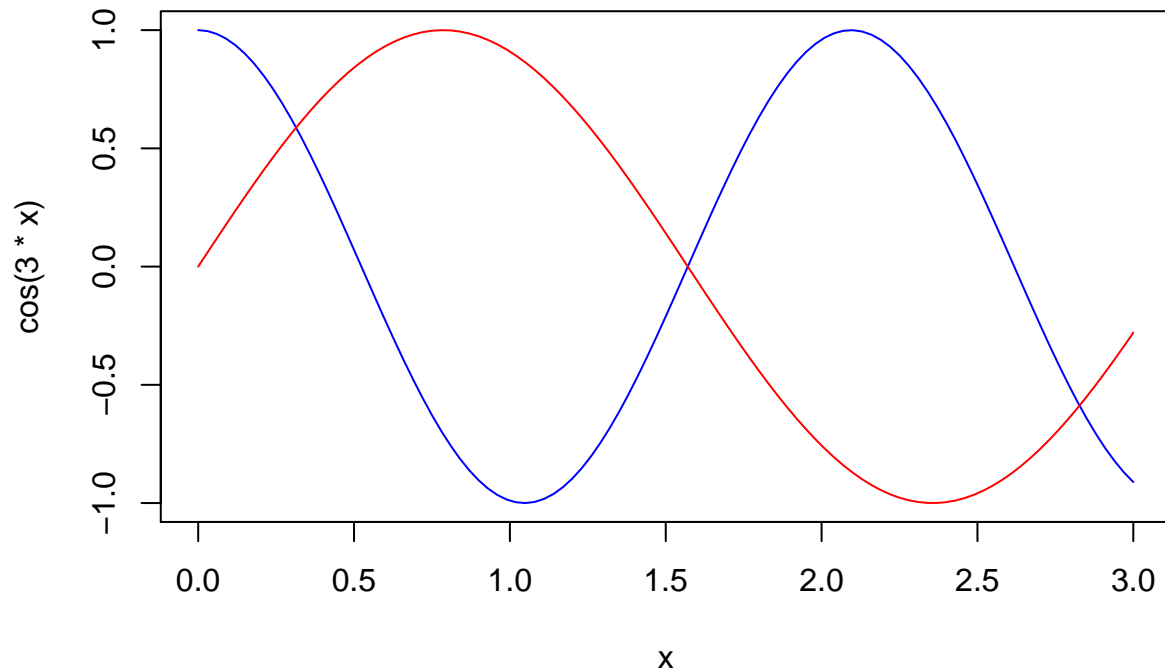
Micro práctica

- Realiza una gráfica del conjunto de datos `women` (`women` es una base de datos *precargada* en R, se accede a ella con `data(women)`). Coloque "Altura" como etiqueta en el eje x y "Peso" en el y. Como título ponga "Valores promedio de altura y peso", subtítulo: "Mujeres de 30 a 39 años". El gráfico debe lucir así:

Valores promedio de altura y peso



- Haga una gráfica de la función $\cos(3x)$ de 0 a 3, de color azul. Superponga la gráfica de $\sin(2x)$ de color rojo. La gráfica debe lucir así:



- Cerremos esta primera parte gráfica con un ejemplo en 3D:

```
library(rgl)
r <- 1
a <- runif(1000,0,2*pi)
```



```
u <- runif(1000,-r,r)

x <- cos(a)*sqrt(1-u^2)
y <- sin(a)*sqrt(1-u^2)
z <- u
plot3d(x,y,z,col="blue")
```

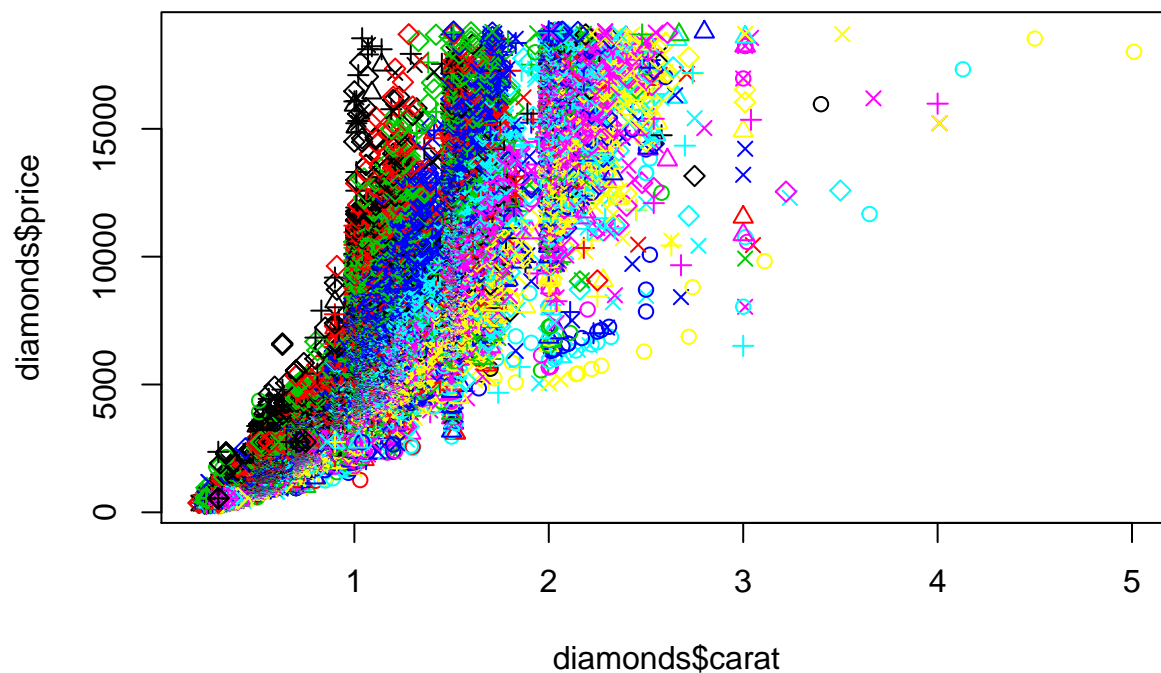
- Hasta el momento hemos cubierto formas básicas de hacer gráficos. Existen opciones más avanzadas como lattice, ggplot2 y ggvis. Hagamos un breve paseo por **ggplot** y luego hacemos una práctica:

ggplot2

Un gráfico básico (usando lo aprendido hasta ahora)

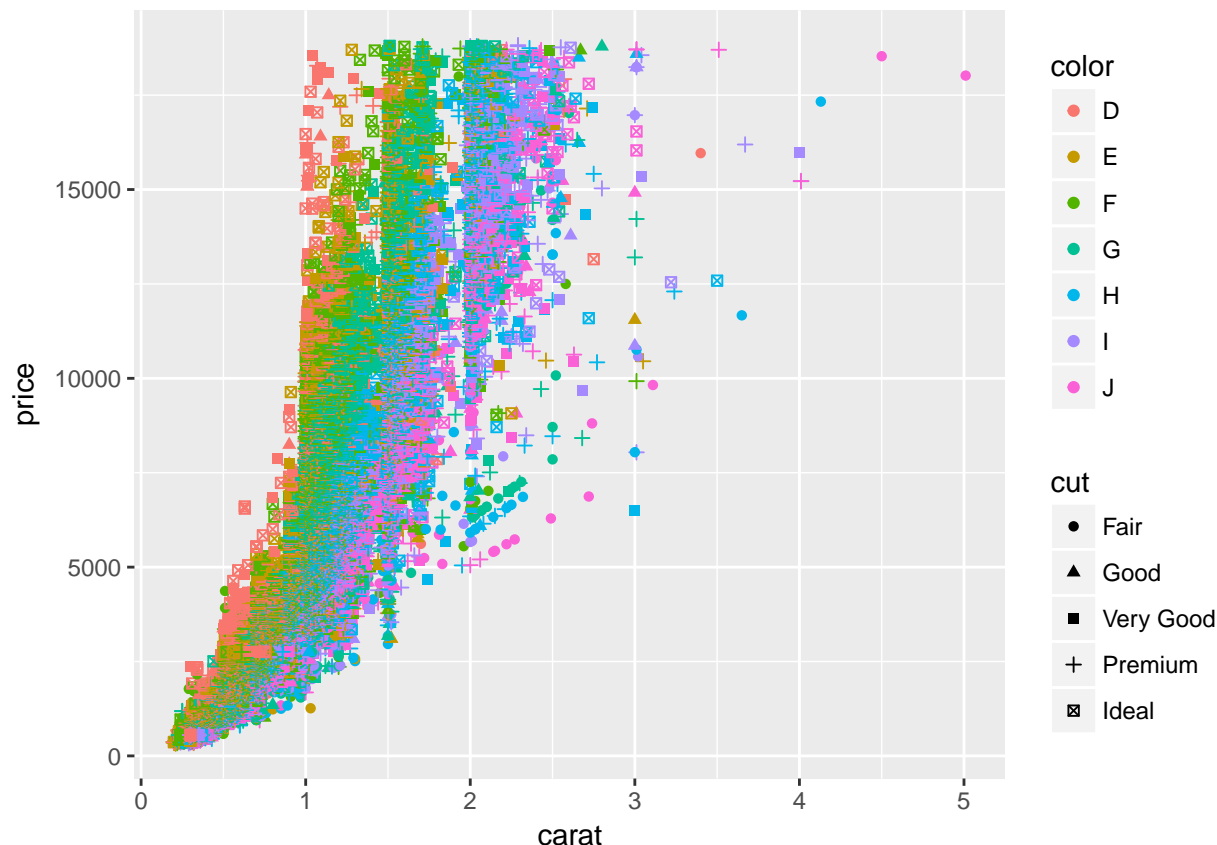
```
# Obtenemos los datos
library(ggplot2)
data(diamonds)

# basic plotting
plot(diamonds$carat, diamonds$price, col = diamonds$color,
     pch = as.numeric(diamonds$cut))
```



Usando ggplot:

```
ggplot(diamonds, aes(carat, price, col = color, shape = cut)) +
  geom_point()
```



Práctica

- Abra los datos de R `cars`. Es una buena práctica el conocer los nuestros datos. Mira la ayuda de `cars`.
- Ejecuta `head(cars)`. Para tener una mejor idea de nuestros datos, podemos usar funciones como: `dim()`, `names()`, `head()`, `tail()` y `summary()`.
- Hagamos un gráfico básico `plot(cars)`
 - R nota que el data frame tiene sólo dos columnas, por lo que asume que deseas graficar una columna vs la otra.
- Además, ya que no se proporcionan etiquetas para los ejes, R utiliza los nombres de las columnas.
- Ahora, ejecuta `plot(x = cars$speed, y = cars$dist)`. ¿Notas alguna diferencia con el gráfico anterior?
- Ten en cuenta que hay otras maneras de usar el comando `plot`, esto es, utilizando el interfaz de "fórmula". Por ejemplo, tenemos gráfico similar al anterior con `plot(dist ~ speed, cars)`. Sin embargo, vamos a usar más tarde en la práctica antes de utilizar la interfaz de fórmula.
- Hagamos un gráfico de cars con `dist` en el eje x y `speed` en el eje y .
- En el gráfico anterior, agrega la etiqueta "Speed" en el eje x .
- En el gráfico anterior, agrega la etiqueta "Stopping Distance" en el eje y .
- Ahora, agrega el título "My Plot".
- Agrega el subtítulo "My Plot Subtitle".

- Agrega la opción `col=2`
- Limita los ejes usando la opción `xlim = c(10,15)`
- Podemos cambiar la forma de los puntos, trata con la opción `pch = 2`.
- Carga los datos `mtcars`. Usa `boxplot()` con la fórmula `= mpg ~ cyl` y `data = mtcars`.
- Finalmente, realiza un histograma de `mtcars$mpg` con la función `hist`.