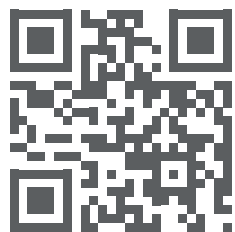




Universitat
de les Illes Balears

#SOM
UIB



AprendeR: Introducción al tratamiento de datos con R y RStudio



Módulo 4

Lección 6
Gráficos I

Edició: abril 2016

Edita: Campus Extens – UIB Virtual

Disseny portada: Direcció de l'Estratègia de Comunicació i Promoció Institucional (dircom.uib.cat)



Aquesta obra està subjecta a una llicència CC

[Reconeixement-NoComercial-SenseObraDerivada 4.0 Internacional](https://creativecommons.org/licenses/by-nc-nd/4.0/)

Lección 6

Gráficos I

El objetivo de esta lección es introducir los aspectos básicos de los gráficos que se obtienen por medio de la función `plot`. Muchos de los parámetros y funciones auxiliares que explicamos en esta lección se podrán usar más adelante en otras funciones que producen tipos específicos de gráficos en estadística descriptiva.

6.1. La función `plot`

Dada una familia de puntos

$$(x_1, y_1), \dots, (x_n, y_n),$$

podemos usar la función `plot` para dibujar su gráfico. La construcción básica para hacerlo es

$$\text{plot}(x, y),$$

donde $x = (x_1, \dots, x_n)$ es el vector de sus primeras coordenadas e (y_1, \dots, y_n) el vector de sus segundas coordenadas. Por ejemplo, para dibujar el gráfico de los puntos

$$(2, 1), (5, 7), (6, 3), (3, 2), (4, 1),$$

basta entrar

```
> x=c(2,5,6,3,4)
> y=c(1,7,3,2,1)
> plot(x,y)
```

y obtenemos la Figura 6.1.

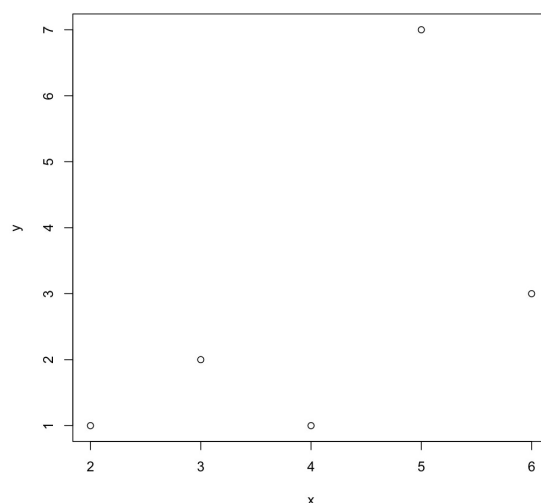


Figura 6.1. Gráfico básico de los puntos $(2, 1)$, $(5, 7)$, $(6, 3)$, $(3, 2)$, $(4, 1)$.

Cuando aplicamos `plot` a un solo vector (x_1, \dots, x_n) , R produce el gráfico de los puntos

$$(1, x_1), \dots, (n, x_n).$$

Es decir, si el vector tiene longitud n , `plot(x)` es una abreviatura de `plot(1:n, x)`. Así, la Figura 6.2 se obtiene con la siguiente instrucción:

```
> plot(2^(1:5))
```

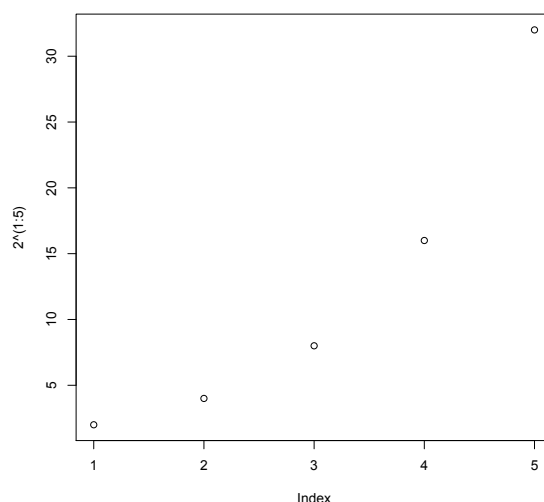


Figura 6.2. Gráfico básico de los valores $(2^n)_{n=1,\dots,5}$.

La función `plot` también sirve para dibujar el gráfico de una función definida mediante `function`. Por ejemplo, la gráfica de la función $y = x^2$ entre $x = 0$ y $x = 1$ de la Figura 6.3 se obtiene con el código siguiente:

```
> f=function(x){x^2}  
> plot(f)
```

6.2. Parámetros de plot

El aspecto de los gráficos que produce `plot` se puede modificar especificando parámetros en su argumento. Por ejemplo, ya vimos en la Lección 2 el parámetro `log`, que sirve para indicar si queremos algún eje en escala logarítmica.

Un primer grupo de parámetros permiten modificar el aspecto «exterior» del gráfico:

- Para poner un título al gráfico, tenemos que emplear `main="título"`.
- Para modificar las etiquetas de los ejes de coordenadas, tenemos que usar `xlab="etiqueta"` e `ylab="etiqueta"`.

Los valores de estos parámetros se tienen que entrar entre comillas o, si son fórmulas matemáticas, aplicarles la función `expression()`, para que aparezcan en un formato matemático

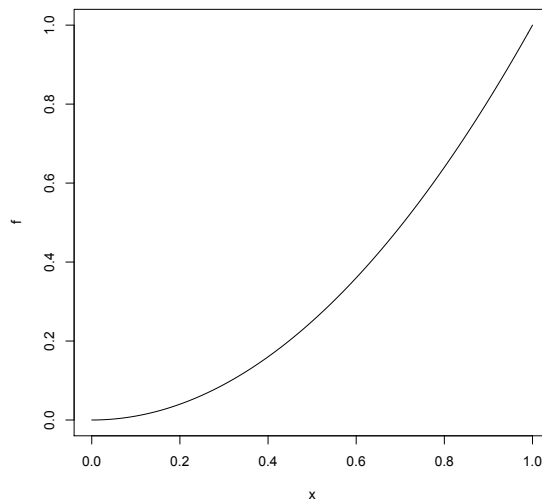


Figura 6.3. Gráfico básico de la función $y = x^2$.

más adecuado. Por ejemplo, suponemos que conocéis la sucesión $(F_n)_{n \geq 0}$ de los números de Fibonacci

$$1, 1, 2, 3, 5, 8, 13, 21, \dots,$$

que empieza con $F_0 = F_1 = 1$ y a partir de aquí cada término es la suma de los dos anteriores. Esta sucesión está definida por la fórmula

$$F_n = \frac{1}{\sqrt{5}} \cdot \left(\frac{1 + \sqrt{5}}{2} \right)^{n+1} - \frac{1}{\sqrt{5}} \cdot \left(\frac{1 - \sqrt{5}}{2} \right)^{n+1} \quad \text{para todo } n \geq 0.$$

Para comprobarlo para algunos valores, vamos a generar la parte inicial $(F_n)_{n=0,\dots,30}$ de la sucesión definida por esta fórmula.

```
> n=0:30
> Fib=(1/sqrt(5))*((1+sqrt(5))/2)^(n+1) -
      (1/sqrt(5))*((1-sqrt(5))/2)^(n+1)
> Fib
[1]      1      1      2      3      5      8     13
[8]     21     34     55     89    144    233    377
[15]    610    987   1597   2584   4181   6765  10946
[22]   17711  28657  46368  75025  121393 196418 317811
[29]  514229  832040 1346269
```

Para dibujar los pares $(n, F_n)_{n=0,\dots,30}$ en un gráfico titulado «Números de Fibonacci», con el eje de abscisas etiquetado con n y el eje de ordenadas etiquetado con F_n igualado a su fórmula explícita, podemos entrar la instrucción siguiente:

```
> plot(n, Fib, xlab="n", main="Números de Fibonacci",
      ylab=expression(F[n]==(1/sqrt(5))*((1+sqrt(5))/2)^(n+1)
        -(1/sqrt(5))*((1-sqrt(5))/2)^(n+1)))
```

El resultado es la Figura 6.4. Si os interesa información sobre cómo escribir las fórmulas dentro de `expression`, podéis consultar la Ayuda de `plotmath`.

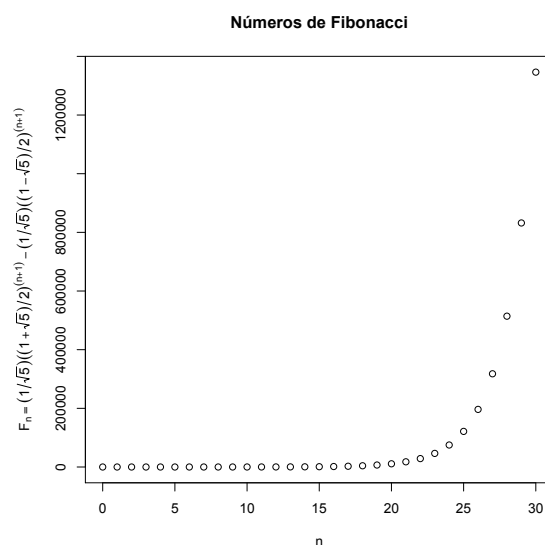


Figura 6.4. Gráfico de los 31 valores iniciales F_n de la sucesión de Fibonacci.

Vale, hemos hecho trampa. Si habéis ejecutado la instrucción anterior tal cual, la etiqueta del eje de ordenadas os habrá quedado demasiado cerca del borde, y, por ejemplo, no se verá la línea horizontal superior de las raíces cuadradas. Para solucionarlo, hemos modificado los márgenes de la figura. Estos márgenes, especificados en números de líneas, se controlan con el parámetro `mar`, pero este parámetro no se puede especificar dentro de `plot`, sino antes, como argumento de la función `par`, que sirve para modificar el aspecto general de los gráficos. Así que en realidad, las instrucciones que han producido el gráfico han sido:

```
> viejo.par=par() #Guardamos los valores antiguos de par
> par(mar=c(5,4.5,4,2)+0.1) #Modificamos los márgenes
> plot(n, Fib, xlab="n", main="Números de Fibonacci",
      ylab=expression(F[n]==(1/sqrt(5))*((1+sqrt(5))/2)^(n+1)
        -(1/sqrt(5))*((1-sqrt(5))/2)^(n+1)))
> par=viejo.par #Volvemos a los valores anteriores de par
```

Consultad la Ayuda de `par` para conocer cómo modificar otros parámetros gráficos.

Cómo podéis ver, por defecto `plot` dibuja los puntos como círculos vacíos. Esto se puede cambiar especificando el signo con el parámetro `pch`, que puede tomar como valor cualquier número natural entre 0 y 25. Con `pch=0` obtenemos cuadrados, `pch=1` produce los círculos que ya habéis visto (es el valor por defecto), `pch=2` produce triángulos, etc. La Figura 6.5 muestra los signos correspondientes a los valores de `pch`. También se pueden usar letras para representar los puntos: hay que especificarlo igualando el parámetro `pch` a la letra entre comillas.



Figura 6.5. Signos correspondientes a los diferentes valores de `pch`.

El tamaño de estos signos se puede modificar mediante el parámetro `cex` igualado al factor de escalado: `cex=2` produce signos el doble de grandes que los que obtenemos por defecto, `cex=0.5` produce signos de la mitad de tamaño, etc. Por ejemplo,

```
> x=c(2,5,6,3,4); y=c(1,7,3,2,1)
> plot(x, y, pch=20, cex=3)
```

produce la Figura 6.6.

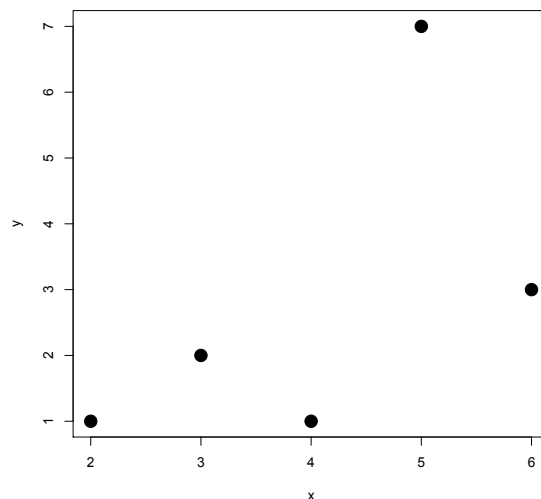


Figura 6.6. Gráfico de los puntos (2,1), (5,7), (6,3), (3,2), (4,1) con puntos de tamaño triple.

El color de los puntos se puede especificar mediante el parámetro `col` igualado al nombre del color en inglés. La paleta de colores de R consta de 502 colores diferentes. Podéis encontrar una presentación muy clara de esta paleta en el documento `Rcolor.pdf` que encontraréis en el repositorio del curso o en su *url* original

<http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf>

Así, por ejemplo,

```
> #Con los últimos valores de x e y
> plot(x, y, col="red", pch=15)
```

produce el gráfico de la izquierda de la Figura 6.7.

Los puntos que se obtienen con `pch` de 21 a 25 admiten un color para el borde (que se especifica con `col`) y uno de diferente para el relleno, que se especifica con `bg`. A modo de ejemplo, el gráfico de la derecha de la Figura 6.7 se obtiene con el código siguiente:

```
> #Con los últimos valores de x e y
> plot(x, y, pch=21, col="blue", bg="red", cex=2)
```

El parámetro `type` permite indicar el tipo de gráfico que queremos producir. El valor del parámetro se tiene que entrar entre comillas y puede ser:

- "p", para dibujar los puntos como simples puntos, como hasta ahora; es el valor por defecto.
- "l", para dibujar los puntos unidos por líneas rectas sin que se vean los puntos.

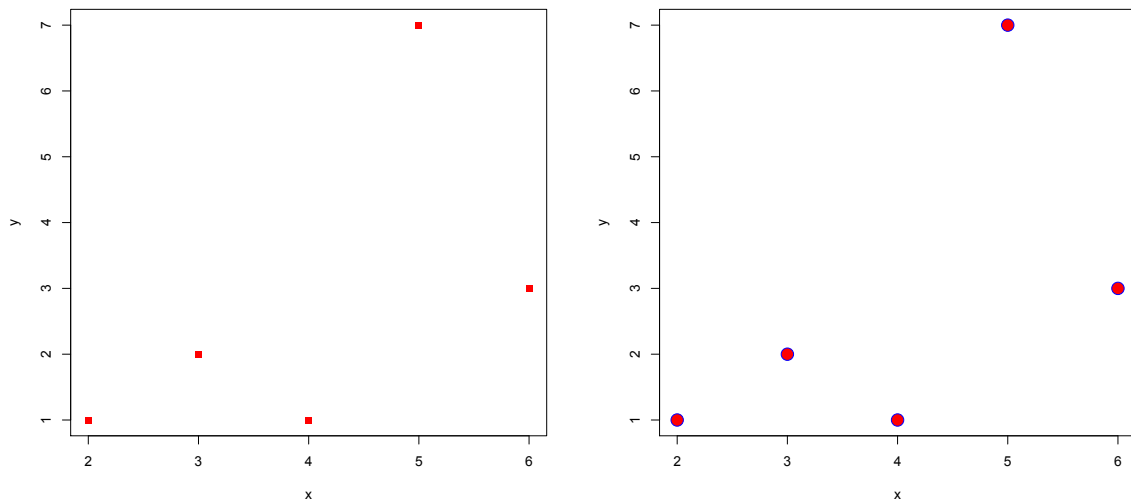


Figura 6.7. Dos gráficos de los puntos $(2, 1)$, $(5, 7)$, $(6, 3)$, $(3, 2)$, $(4, 1)$.

- "b", para dibujar los puntos unidos por líneas rectas de manera que se vean los puntos (los dibuja ambos, *both*: rectas y puntos), pero sin que las rectas entren dentro de los signos que representan los puntos.
- "o", que es como "b", pero ahora las rectas sí que entran dentro de los puntos.
- "h", para dibujar líneas verticales desde el eje de abscisas a cada punto (un *histograma de líneas*).
- "s", para dibujar un diagrama de escalones.
- "n", para no dibujar los puntos, sólo el exterior del gráfico (ejes, título, etc.).

Veamos un ejemplo sencillo. Con el código

```
> x=c(1,3,5,7,11)
> y=c(2,8,4,6,3)
> plot(x, y, main="con p")
> plot(x, y, type="l", main="con l")
> plot(x, y, type="b", main="con b")
> plot(x, y, type="o", main="con o")
> plot(x, y, type="h", main="con h")
> plot(x, y, type="s", main="con s")
> plot(x, y, type="n", main="con n")
```

se obtienen los gráficos de la Figura 6.8.

La función `plot` aplicada a una función f en realidad produce el gráfico de tipo "l" de una familia de puntos $(x, f(x))$ (por defecto, 101 puntos distribuidos uniformemente a lo largo del dominio; este valor se puede modificar con el parámetro `n`).

El estilo de las líneas usadas por `plot` se puede modificar con los parámetros siguientes:

- El parámetro `col` especifica el color tanto de los puntos como de las líneas.

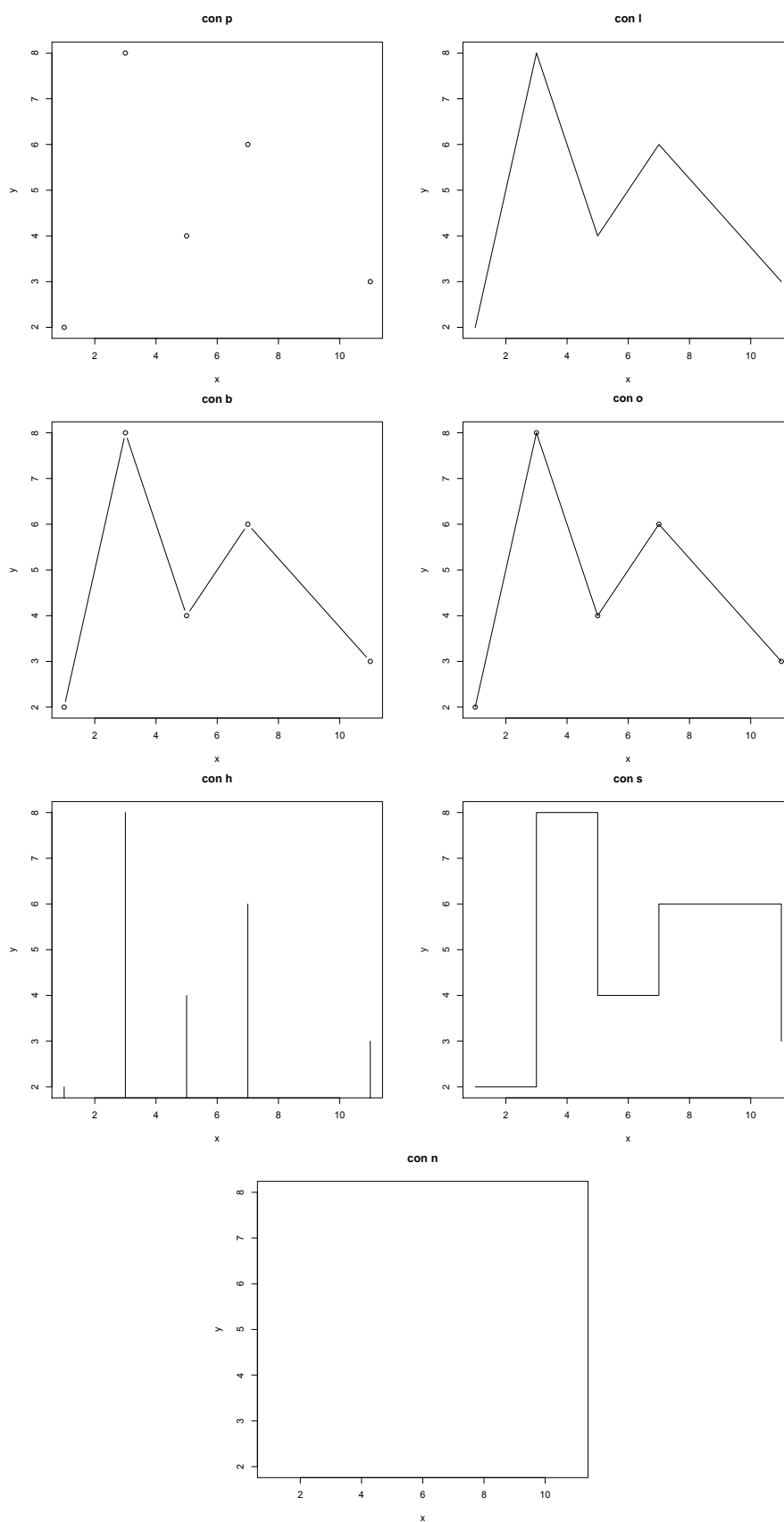


Figura 6.8. Gráficos de los puntos $(1, 2)$, $(3, 8)$, $(5, 4)$, $(7, 6)$, $(11, 3)$ con los distintos valores del parámetro `type`.

- El tipo de línea se puede especificar con el parámetro `lty` igualado a uno de los valores siguientes: `"solid"`, o 1, (el valor por defecto, que produce una línea continua); `"dashed"`, o 2, (que produce una línea discontinua); `"dotted"`, o 3, (que produce una línea de puntos); `"dotdash"`, o 4, (que produce una línea que alterna puntos y rayas).
- El grosor se puede especificar con el parámetro `lwd`. Con `lwd=x` imponemos que el grosor de las líneas sea x veces su valor por defecto.

Así, los dos gráficos de la Figura 6.9 se obtienen con el código siguiente:

```
> x=c(1,3,5,7,11)
> y=c(2,8,4,6,3)
> plot(x, y, type="o", pch=19, lty="dotted")
> plot(x, y, type="o", pch=19, lty="dashed", lwd=2)
```

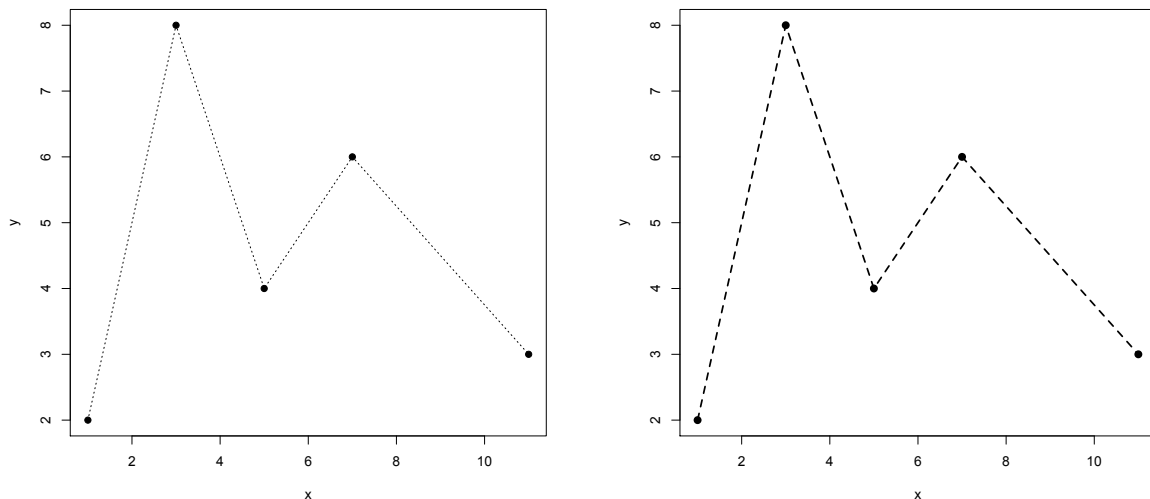


Figura 6.9. Gráficos de los puntos (1,2), (3,8), (5,4), (7,6), (11,3) con distintos valores de los parámetros `lty` y `lwd`.

Si el argumento de `plot` son dos vectores, por norma general los rangos de los ejes de coordenadas van, por defecto, del mínimo al máximo de los vectores correspondientes. Si su argumento es una función f , por defecto el rango del eje de abscisas es el intervalo $[0, 1]$ y el rango del eje de ordenadas va del valor mínimo de f sobre el rango de las abscisas al máximo. Si queremos modificar estos rangos, tenemos que usar los parámetros `xlim` e `ylim`, igualados cada uno a un vector de entradas los extremos del rango.

Veamos un ejemplo de uso de estos parámetros:

```
> f=function(x){x*log(x)}
> plot(f)      #Rangos por defecto
> plot(f, xlim=c(1,10))    #Fijamos el rango de x
> plot(f, xlim=c(1,10), ylim=c(0,50)) #Fijamos los rangos de x e y
```

Estas instrucciones producen, respectivamente, los tres gráficos de la Figura 6.10.

Para modificar las posiciones de las marcas en los ejes de abscisas y ordenadas podemos usar los parámetros `xaxp` e `yaxp`, respectivamente. Mediante la expresión `xaxp=c(n_1, n_2, m)`,

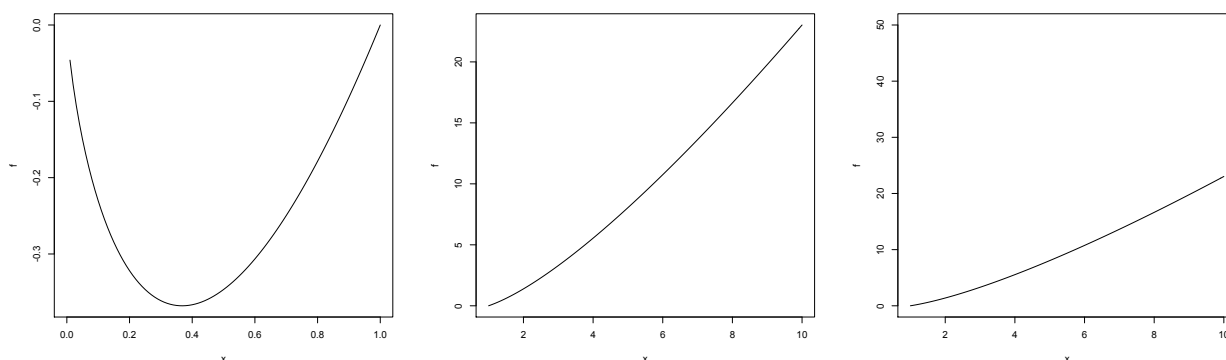


Figura 6.10. Gráficas de la función $f(x) = x \log x$ con diferentes rangos de coordenadas de los ejes.

imponemos que R dibuje $m + 1$ marcas igualmente espaciadas entre los puntos n_1 y n_2 del eje de abscisas. La sintaxis para `yaxp` es la misma. Estas instrucciones no definen los rangos de los ejes de coordenadas, que se han de especificar con `xlim` e `ylim` si se quieren modificar. Por ejemplo,

```
> f=function(x){x*log(x)}
> plot(f, xlim=c(1,10))
> plot(f, xlim=c(1,10), yaxp=c(1,10,9))
> plot(f, xlim=c(1,10), yaxp=c(1,10,9), ylim=c(0,25),
yaxp=c(0,25,10))
```

producen, respectivamente, los tres gráficos de la Figura 6.11.

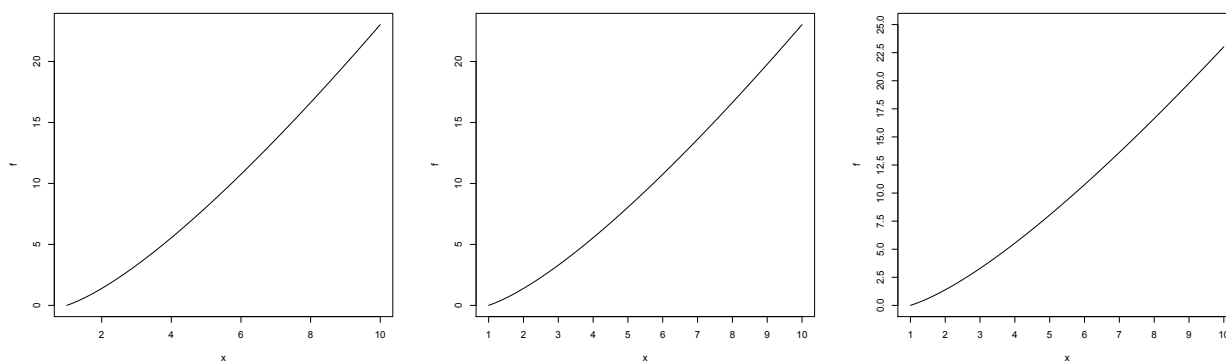


Figura 6.11. Gráficas de la función $f(x) = x \log x$ con diferentes posiciones de las marcas en los ejes.

La función `plot` dispone de muchos otros parámetros, que podéis consultar en la Ayuda de `plot` y las de las funciones que se citan en su sección **See Also**. Son especialmente útiles los parámetros que se explican en la Ayuda de `par`.

6.3. Cómo añadir elementos a un gráfico

La función `plot` permite dibujar una sola cosa (una familia de puntos o una función) con un único estilo. Si queremos dibujar varios objetos en un gráfico, los tenemos que añadir uno a

uno, usando las funciones adecuadas. En esta sección veremos algunas funciones que permiten añadir elementos a un gráfico.

Antes de empezar, queremos avisaros de algo muy importante. Cuando añadimos objetos a un gráfico, ya no podemos modificar su diseño general. Por ejemplo, los rangos de coordenadas de los ejes del gráfico final o sus etiquetas serán los del primer gráfico, aunque especifiquemos valores nuevos en el argumento de las funciones que usemos para añadir objetos.

La instrucción `points(x,y)` añade un punto de coordenadas (x,y) al gráfico activo. Podemos declarar el color, el signo etc., de este punto mediante los parámetros usuales. (**¡Atención!** La instrucción es `points`, no `point`.) Por ejemplo,

```
> f=function(x){x^2}
> plot(f, xlim=c(-3,3))
> points(0,0, pch=19)
```

produce la Figura 6.12, en la que hemos dibujado la parábola $y = x^2$ y hemos marcado su vértice con un punto.

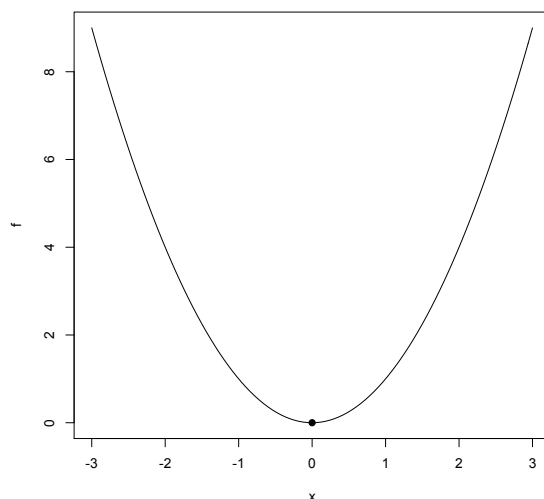


Figura 6.12. Gráfica de la función $f(x) = x^2$ con su vértice marcado.

La función `points` también sirve para añadir una familia de puntos. En este caso, hay que entrar en el argumento de `points(x,y)` la lista x de sus primeras coordenadas y la lista y de sus segundas coordenadas, como lo haríamos en `plot`. Veamos algunos ejemplos:

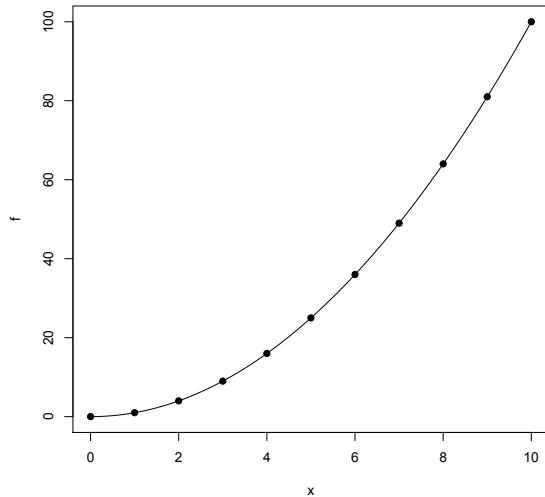
```
> f=function(x){x^2}
> plot(f, xlim=c(0,10))
> points(0:10,(0:10)^2, pch=19)
```

dibuja un trozo de la parábola $y = x^2$ y añade los puntos $(n, n^2)_{n=0,\dots,10}$, produciendo la Figura 6.13.(a), y el código

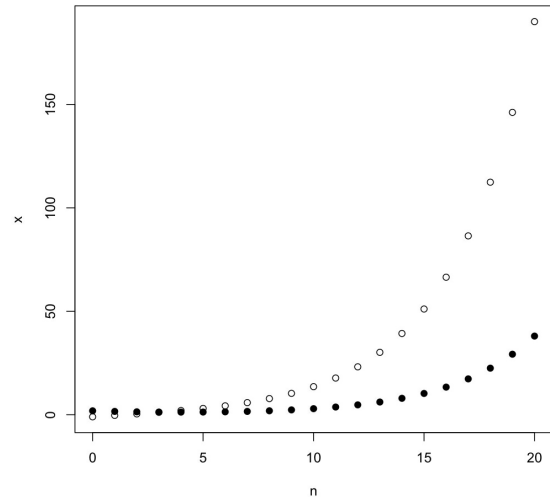
```
> n=0:20
> x=1.3^n-2*0.8^n
> y=0.2*1.3^n+1.7*0.8^n
> plot(n,x, col="blue")
```

```
> points(n,y, pch=19, col="red")
```

primero dibuja los puntos $(n, 1.3^n - 2 \cdot 0.8^n)_{n=0,\dots,20}$ como circulitos azules vacíos, y después añade los puntos $(n, 0.2 \cdot 1.3^n + 1.7 \cdot 0.8^n)_{n=0,\dots,20}$ como circulitos rojos llenos, produciendo la Figura 6.13.(b).



(a)



(b)

Figura 6.13. (a) Gráfico de la función $y = x^2$ y los puntos $(n, n^2)_{n=0,\dots,10}$; (b) Gráfico de las familias de puntos $(n, 1.3^n - 2 \cdot 0.8^n)_{n=0,\dots,20}$ y $(n, 0.2 \cdot 1.3^n + 1.7 \cdot 0.8^n)_{n=0,\dots,20}$ usando diferentes signos para cada familia.

La función `abline` sirve para añadir una recta; ya la usamos en la Lección 2 para añadir a un gráfico una recta de regresión calculada con `lm`. Esta función tiene tres variantes:

- `abline(a,b)` añade la recta $y = a + bx$.
- `abline(v=x0)` añade la recta vertical $x = x_0$.
- `abline(h=y0)` añade la recta horizontal $y = y_0$.

Podemos especificar las características de estas rectas, como su grosor, su estilo o su color, mediante los parámetros pertinentes. Por ejemplo,

```
> f=function(x){x^2}
> plot(f, xlim=c(-3,3), col="red")
> points(0,0, pch=19, col="blue")
> points(1,1, pch=19, col="blue")
> abline(v=1, lty="dashed")
> abline(h=0, lty="dotted")
> abline(-1,2, lty="dotdash")
```

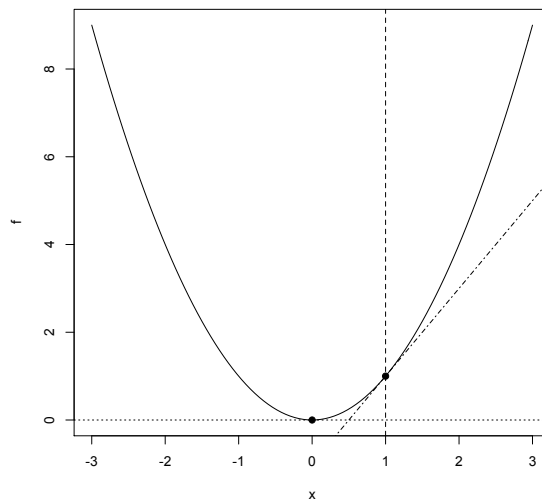


Figura 6.14. Gráfica de la función $f(x) = x^2$, los puntos $(0,0)$ y $(1,1)$ y las rectas $x = 1$, $y = 0$ e $y = 2x - 1$.

produce la Figura 6.14. En este caso, hemos añadido a la gráfica de la parábola $y = x^2$, dos puntos en $(0,0)$ y $(1,1)$, la recta horizontal $y = 0$ de puntos, la recta vertical $x = 1$ discontinua y la recta $y = -1 + 2x$ de puntos y rayas.

Los parámetros `v` y `h` de `abline` se pueden igualar a vectores numéricos, en cuyo caso la instrucción añade en un solo paso todas las rectas verticales u horizontales correspondientes, todas del mismo estilo. Incluso se pueden combinar los parámetros `v` y `h` en una misma función:

```
> f=function(x){x^2}
> plot(f, xlim=c(-3,3), col="red")
> abline(h=0:9, v=-3:3, lty="dotted")
```

produce la Figura 6.15.

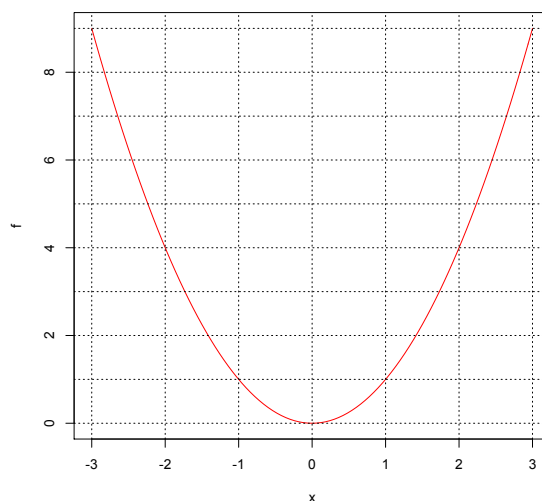


Figura 6.15. Gráfica de la función $f(x) = x^2$ y una rejilla de rectas horizontales y verticales.

La instrucción

```
text(x,y, labels=...)
```

añade en el punto de coordenadas (x,y) el texto especificado como argumento de `labels`. El texto se puede entrar entre comillas o en una `expression`. Esta instrucción admite un parámetro opcional, `pos`, que puede tomar valores 1, 2, 3 o 4 y permite indicar la posición del texto alrededor de las coordenadas (x,y) : 1 significa abajo, 2 a la izquierda, 3 arriba y 4 a la derecha. Sin especificar `pos` (o, equivalentemente, especificando `pos=NULL`, que es su valor por defecto), el texto se sitúa centrado en el punto (x,y) . El efecto de `pos` se ilustra en la Figura 6.16, producida con el código siguiente:

```
> plot(0,0, pch=19, xlab="", ylab="")
> text(0,0, labels="pos 1", pos=1)
> text(0,0, labels="pos 2", pos=2)
> text(0,0, labels="pos 3", pos=3)
> text(0,0, labels="pos 4", pos=4)
> points(0,0.5, pch=19)
> text(0,0.5, labels="no pos")
```

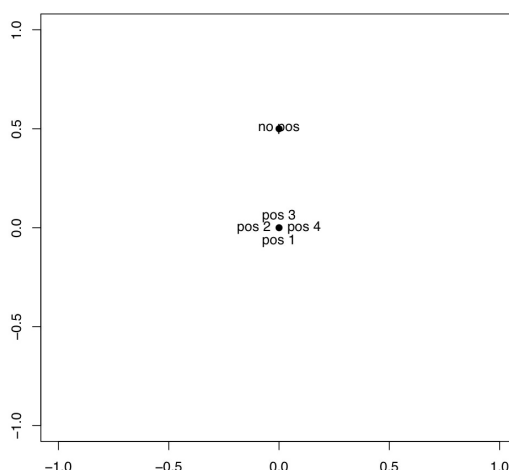


Figura 6.16. Significado del parámetro `pos` de la función `text`.

La función `text` se puede usar para añadir varios textos en un solo paso. En este caso, x e y han de ser los vectores de abscisas y ordenadas de los puntos donde se añadirán los textos, `labels` el vector de textos, y `pos` el vector de sus posiciones; en este vector, los textos que queremos centrados en su posición se han de indicar con `NULL`. Así, la Figura 6.16 también se hubiera podido obtener con el código siguiente, como podréis comprobar si lo ejecutáis:

```
> plot(0,0, pch=19, xlab="", ylab="")
> points(0,0.5, pch=19)
> text(rep(0,5), c(rep(0,4), 0.5), pos=c(1,2,3,4,NULL),
  labels=c("pos 1", "pos 2", "pos 3", "pos 4", "no pos"))
```

La instrucción `lines(x,y)`, donde $x = (x_i)_{i=1,\dots,n}$ e $y = (y_i)_{i=1,\dots,n}$ son dos vectores numéricos de la misma longitud, añade al gráfico una línea poligonal que une los puntos (x_i, y_i) sucesivos. El efecto es como si añadiéramos un `plot(x,y, type=1)`. Naturalmente, la apariencia de las

líneas la podemos modificar con los parámetros usuales de grosor, color, estilo, etc. A modo de ejemplo, la Figura 6.17 se obtiene con el código:

```
> f=function(x){x^2}
> plot(f, xlim=c(0,10))
> lines(3.33*(0:3), (3.33*(0:3))^2, lwd=2, lty="dashed")
```

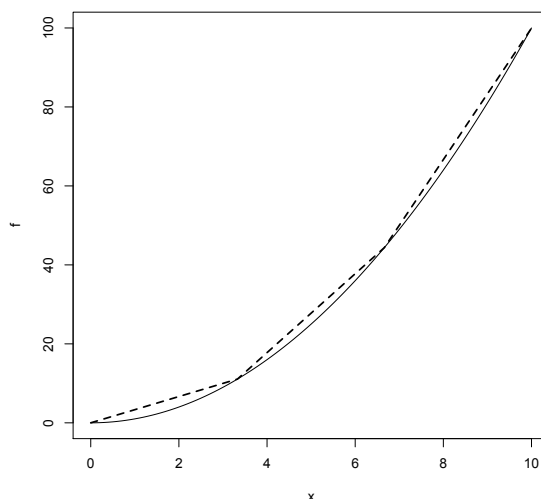


Figura 6.17. Gráfica de la función $f(x) = x^2$ junto a la línea poligonal que une los puntos $(3.33n, (3.33n)^2)_{n=0,\dots,3}$.

La instrucción `curve` con el parámetro `add=TRUE` permite añadir la gráfica de una curva a un gráfico anterior. La curva se puede especificar mediante una expresión algebraica con variable x , o mediante su nombre si la hemos definido antes. Por ejemplo,

```
> f=function(x){x^2}
> plot(f, xlim=c(-10,10), xlab="x", ylab="y", main="monomios")
> curve(x^3, lty="dashed", add=TRUE)
> curve(x^4, lty="dotted", add=TRUE)
```

produce el gráfico de la Figura 6.18.

Las funciones `points`, `abline`, `lines` o `text` sólo sirven para *añadir* elementos a un gráfico. En cambio, la función `curve` también se puede usar para producir la gráfica de una función, como `plot`, con la ventaja sobre esta última que no sólo se puede aplicar a una función definida con `function`, sino también a una expresión algebraica. Además, la función `curve` admite todos los parámetros de `plot`. Así, el gráfico de la Figura 6.18 también se obtiene con

```
> curve(x^2, xlim=c(-10,10), xlab="x", ylab="y", main="monomios")
> curve(x^3, lty="dashed", add=TRUE)
> curve(x^4, lty="dotted", add=TRUE)
```

Cuando dibujamos varias funciones en un gráfico, como el de la Figura 6.18, es conveniente usar una *legenda* para distinguirlas. Para añadirla, se ha de usar la instrucción

```
legend(posición, legend=c(vector de nombres de las curvas), parámetro=c(vector de valores del parámetro), ..., parámetro=c(vector de valores del parámetro)),
```

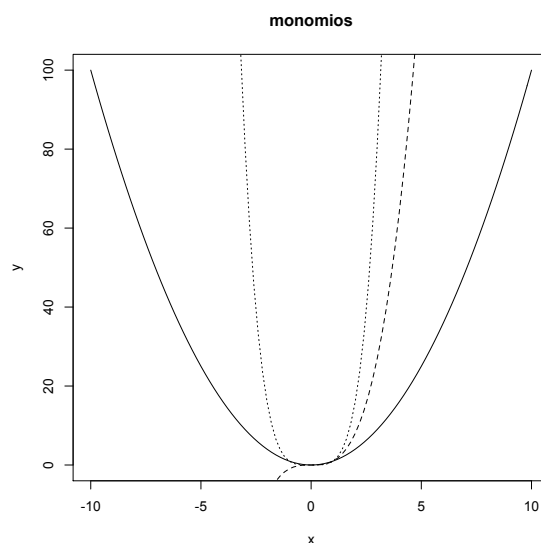



Figura 6.18. Gráficas de diferentes monomios.

donde:

- La *posición* indica donde queremos situar la leyenda, y puede ser o bien dos números para especificar las coordenadas de su esquina superior izquierda, o bien una de las palabras siguientes: "bottomright" (esquina inferior derecha), "bottom" (centrada abajo), "bottomleft" (esquina inferior izquierda), "left" (centrada a la izquierda), "topleft" (esquina superior izquierda), "top" (centrada arriba), "topright" (esquina superior derecha), "right" (centrada a la derecha) o "center" (en el centro del gráfico).
- El vector **legend** contiene los nombres (entre comillas o aplicándoles **expression**) con los que queremos identificar las curvas dentro de la leyenda.
- Cada *parámetro* se usará para especificar el vector de sus valores sobre las diferentes curvas, en el orden en el que aparecen en el vector **legend** e incluyendo los valores por defecto. Se pueden usar varios parámetros. Si se quieren distinguir las curvas por su color, obligatoriamente también se ha de especificar algún otro parámetro (si sólo se distinguen por el color, se ha de añadir **lty** igualado al tipo de línea; por defecto, "solid").

Por ejemplo,

```
> curve(x^2, xlim=c(-10,10), xlab="x", ylab="y", main="monomios")
> curve(x^3, lty="dashed", add=TRUE)
> curve(x^4, lty="dotted", add=TRUE)
> legend("bottomleft", legend=c(expression(x^2), expression(x^3),
  expression(x^4)), lty=c("solid", "dashed", "dotted"))
```

produce el gráfico de la izquierda de la Figura 6.19. Observad que, aunque en el primer **curve** no hemos especificado el parámetro **lty**, y ha tomado su valor por defecto, en el parámetro **lty** del **legend** sí que hemos especificado su valor para la primera función.

Veamos otro ejemplo:

```
> curve(2*x+3, xlim=c(-10,10), ylab="")
```

```
> curve(2*x^2+3, col="red", lwd=2, add=TRUE)
> curve(2*x^3+3, col="blue", lwd=3, add=TRUE)
> legend("topleft", legend=c(expression(2*x+3), expression(2*x^2+3),
  expression(2*x^3+3)), lwd=c(1,2,3), col=c("black","red","blue"))
```

produce el gráfico de la derecha de la Figura 6.19.

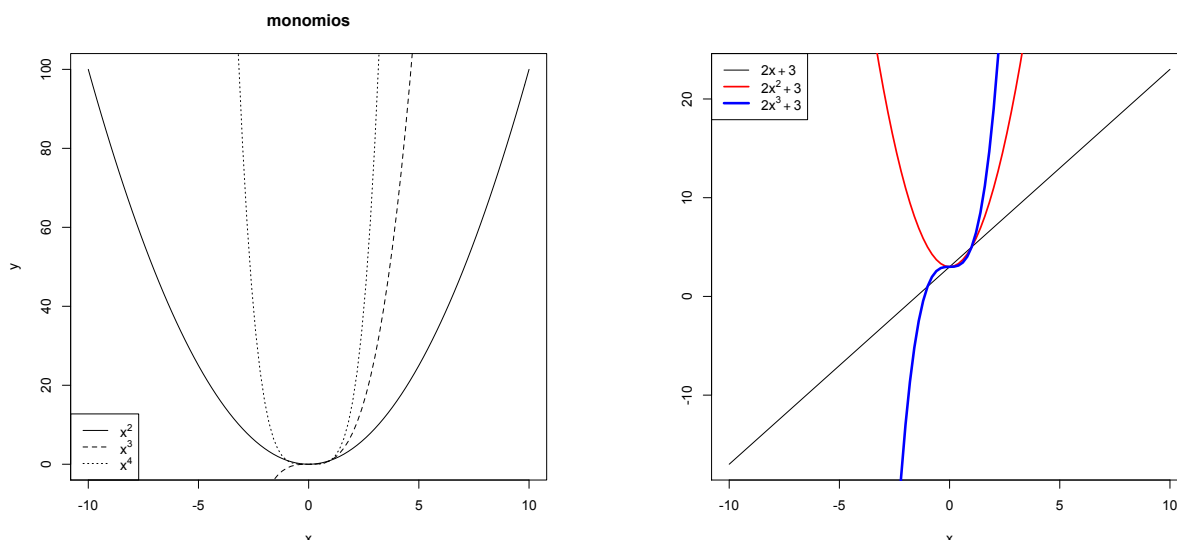


Figura 6.19. Gráficas de diferentes funciones usando como leyenda el tipo de línea (izquierda) o el color y el grosor (derecha).

El código siguiente produce la Figura 6.20; en ella podemos observar que si el único parámetro que especificamos dentro del `legend` es el color, no se ven las líneas dentro de la leyenda.

```
> curve(2*x+3, -10, 10, ylab="")
> curve(2*x^2+3, col="red", add=TRUE)
> curve(2*x^3+3, col="blue", add=TRUE)
> legend("topleft", legend=c(expression(2*x+3), expression(2*x^2+3),
  expression(2*x^3+3)), col=c("black","red","blue"))
```

Si os interesan, también disponéis de las funciones `segments` (para añadir segmentos), `arrows` (para añadir flechas), `symbols` (para añadir signos, como estrellas, termómetros, ...), `polygon` (para añadir polígonos cerrados especificando sus vértices), etc. Consultad las Ayudas de estas instrucciones para los detalles sobre cómo se usan.

6.4. Guía rápida

- `expression` sirve para producir textos matemáticamente bien formateados.
- `par` sirve para modificar el aspecto general de los gráficos. Consultad su Ayuda para conocer todos los parámetros que se pueden especificar en esta función.
- `plot` es la función genérica para producir gráficos. Sus dos usos principales (por el momento) son:

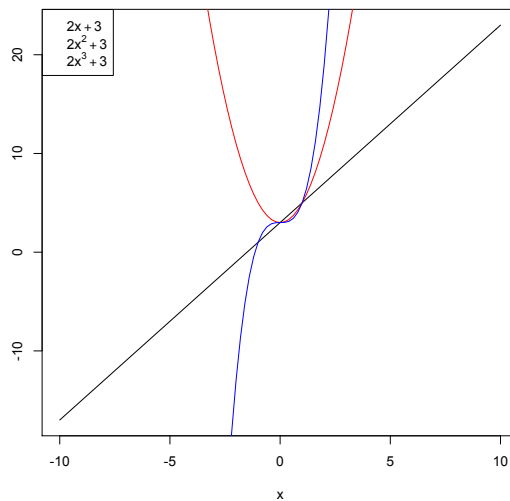


Figura 6.20. Gráficas de diferentes monomios usando como leyenda el color y su expresión, pero no el tipo de línea.

- `plot(x,y)`, que dibuja el gráfico de los puntos (x_n, y_n) .
- `plot(función)`, que dibuja la gráfica de la *función*.

Algunos parámetros importantes:

- `main`: sirve para especificar el título.
- `xlab` e `ylab`: sirven para especificar las etiquetas de los ejes de coordenadas.
- `xlim` e `ylim`: sirven para especificar los rangos de los ejes de coordenadas.
- `xaxp` e `yaxp`: sirven para especificar las marcas en los ejes de coordenadas.
- `log`: sirve para especificar los ejes de coordenadas que estarán en escala logarítmica.
- `type`: sirve para especificar el tipo de gráfico.
- `pch`: sirve para especificar el estilo de los puntos.
- `cex`: sirve para especificar el tamaño de los puntos.
- `col`: sirve para especificar el color del gráfico.
- `bg`: sirve para especificar el color de relleno de los puntos de estilos `pch` de 21 a 25.
- `lty`: sirve para especificar el tipo de las líneas.
- `lwd`: sirve para especificar el grosor de la líneas.

Los parámetros `pch`, `cex`, `col`, `bg`, `type`, `lty` y `lwd` también se pueden usar en las funciones que siguen.

- `points` añade puntos al gráfico activo.
- `abline` añade una recta al gráfico activo. Algunos parámetros específicos importantes:
 - `v`: la abscisa de una recta vertical.
 - `h`: la ordenada de una recta horizontal.

- `lines` añade una línea poligonal al gráfico activo.
- `text` añade un texto al gráfico activo. Algunos parámetros específicos importantes:
 - `pos`: la posición del texto con respecto a sus coordenadas.
 - `labels`: el texto.
- `legend` añade una leyenda al gráfico activo.
- `curve`, con el parámetro `add=TRUE`, sirve para añadir la gráfica de una función al gráfico activo. Esta función admite todos los parámetros de `plot`.

6.5. Ejercicio

La *función de Gompertz* es $G(t) = ae^{-be^{-ct}}$, con $a, b, c \in \mathbb{R}$ estrictamente positivos, y se usa para modelar el crecimiento de tumores o de poblaciones con recursos limitados. Vamos a analizar gráficamente el efecto de los parámetros a, b, c .

- (a) La recta $y = a$ es una asíntota horizontal de $G(t)$. Comprobadlo dibujando en un mismo gráfico dos funciones de Gompertz con los mismos valores de b y c y diferentes valores de a , y las rectas horizontales definidas por estos valores de a . Usad colores, tipos de líneas, textos, leyenda. . . , todo lo que consideréis necesario para ayudar a que el gráfico muestre la información que se desea.
- (b) Para estudiar el efecto de b , dibujad en un mismo gráfico tres funciones de Gompertz con los mismos valores de a y c y diferentes valores de b . De nuevo, usad todo lo que consideréis necesario para mejorar la comprensión del gráfico. A partir de este gráfico, ¿qué efecto diríais que tiene el parámetro b sobre la gráfica de la función?
- (c) Para estudiar el efecto de c , dibujad en un mismo gráfico tres funciones de Gompertz con los mismos valores de a y b y diferentes valores de c . De nuevo, usad todo lo que consideréis necesario para mejorar la comprensión del gráfico. A partir de este gráfico, ¿qué efecto diríais que tiene el parámetro c sobre la gráfica de la función?



Universitat
de les Illes Balears

Campus Extens
UIB Virtual

<http://campusextens.uib.cat>



@campusextensUIB



<http://www.scoop.it/t/recursos-i-eines-per-al-professorat>



<http://campusextensrecursos.uib.es/>