Estadística climatológica con R

Evaluación y visualización de datos: gráficos

GRAFICOS EN R

R ofrece una enorme cantidad de gráficos. Además posee una gran flexibilidad para generar gráficos a medida con un gran número de opciones para cada una de sus funciones de dibujo. El sistema permite desde gráficos muy simples a figuras de gran calidad para incluir en artículos o libros.

Introducción a los gráficos

Los resultados de una función gráfica son enviados a un dispositivo de representación que puede ser una ventana o un fichero de imagen en distintos formatos.

Existen *dos tipos de funciones gráficas*: ***de alto nivel*** que crean una nueva gráfica y las funciones ***de bajo nivel*** que agregan elementos a una grafica ya existente. Las gráficas se producen con respecto a parámetros gráficos que están definidos por defecto y pueden ser modificados con la función ***par***.

Primero veremos como manejar gráficos y dispositivos gráficos; después veremos en detalle algunas funciones graficas y sus parámetros.

Manejo de gráficos

Cuando se ejecuta por primera vez una función grafica *R* abre una ventana para mostrar el grafico. El dispositivo grafico por defecto son las ventanas bajo *Windows* o *X11* en sistemas operativos *Linux*.

El último dispositivo abierto se convierte en el dispositivo activo sobre el que irán todas nuestras acciones.

La función ***dev.list()*** muestra los dispositivos abiertos. Para saber el dispositivo activo: ***dev.cur().*** Si se quiere cerrar un dispositivo: ***dev.off (num)*** donde ***num*** es el número de dispositivo. Si no se pone nada se cierra el activo: ***dev.off().*** Para abrir un dispositivo nuevo: ***dev.new()***

Además de visualizar los gráficos en el ordenador también se pueden convertir, si se desea, en archivos gráficos en multitud de formatos como ***pdf***, ***png***, ***jpeg***, etc.

Funciones gráficas de alto nivel

Vamos a describir unas cuantas funciones graficas de alto nivel aunque existen muchas más.

- ***plot(x)*** representa los valores de vector *x* en el eje y ordenados en el eje x.
- ***plot(x,y)*** grafico divariado de *x* (en el eje x) frente a *y* (en el eje y). Tienen que tener la misma longitud *x* e *y*.
- ***pie(x)*** grafico de tarta o de pastel.
- ***boxplot(x)*** grafico de cajas
- ***hist(x)*** histograma de frecuencias de x
- ***barplot(x)*** histograma de los valores de x

Las opciones y argumentos para cada una de estas opciones se pueden encontrar en la ayuda incorporada en R. Por ejemplo:

> ?plot o help(plot)

Para dibujar una función cualquiera primero se crea un vector con los puntos del eje de abscisas (x) y luego se representa la función. Por ejemplo para pintar el ***cos(x),*** creamos un vector entre ***–pi*** y ***pi*** con muchos puntos (50 o más)

> x <- seq(-pi,pi,len=200)

Y después ploteamos la función ***cos(x)*** frente a ***x***. El type ***'l'*** es para que una los puntos con una línea continua:

> plot(x, cos(x), type='l')

Existe otra función en R, ***curve()***, que nos pinta directamente cualquier función en un intervalo, sin necesidad de crear primero un vector para dibujar el eje x, simplemente dando la función y el intervalo del eje x. La función ***curve*** admite los mismos parámetros que ***plot***, como títulos, colores, etc. El caso anterior lo plotearíamos así:
> curve (cos, -pi, pi)

Algunas de estas opciones son idénticas para varias funciones graficas; estas son las principales:

type="p" especifica el tipo de línea al pintar el grafico***;***
- ***"p":*** puntos (por defecto)
- ***"l":*** líneas
- ***"b":*** puntos conectados por líneas
- ***"o":*** igual al anterior, pero las líneas están sobre los puntos
- ***"h":*** líneas verticales
- ***"s":*** escaleras, los datos se representan como la parte superior de las líneas verticales
- ***"S":*** igual al anterior pero los datos se representan como la parte inferior de las líneas verticales

xlim=, ylim= especifica los lımites inferiores y superiores de los ejes; por ejemplo con:
xlim=c(1, 10) o ***xlim=range(x)***
xlab=, ylab= títulos (etiquetas, labels) en los ejes; deben ser variables de tipo carácter
main= título principal; debe ser de tipo carácter
sub= subtitulo (escrito en una letra más pequeña y en la parte inferior)

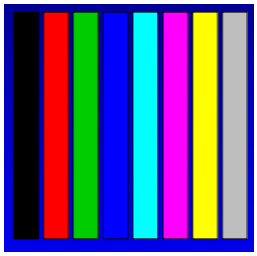
Nota: A todas estas opciones se le pueden añadir las de los *parámetros gráficos* que se verán más adelante, como ***col='red'*** (color de los símbolos), ***cex=0.9*** (tamaño símbolos), ***lwd=1.1*** (anchura de las líneas), etc.

axes=TRUE si es FALSE no dibuja los ejes ni la caja del grafico
ann=TRUE si es FALSE no dibuja anotaciones en los ejes

Colores en R

Los colores que se utilizan en los gráficos pueden designarse en R de diversas maneras. La forma más sencilla es la ***paleta básica***.

> barplot (rep(1,8), yaxt="n", col=1:8) # pintamos un diagrama de barras con los numeros del 1 al 8 y con los colores de la paleta básica del 1 al 8. (yaxt='n' es para que no pinte el eje y)



Para ver los colores asignados a la ***paleta básica***:
> palette()

También se pueden definir los colores por su nombre:
colores <- c('blue','red','yellow')
> barplot (rep(1,3), yaxt="n", col=colores)

Para ver la lista de colores con sus nombres:
> colors()

(Para saber más sobre colores ver: ***Notas.Sueltas.sobre.R***)

Ejercicios 1.1

- Representar la función ***sin(x)*** entre ***-π*** y ***π***
- Crear un vector ***y*** de 10 elementos aleatorios entre 1 y 50 con la función ***runif (n, a1, a2)*** que genera *n* números aleatorios *reales*, entre *a1* y *a2*, reales (***?runif***) o con la función ***sample (a1:a2, n)*** que es igual que la anterior pero con números aleatorios *enteros*.
- Plotear el vector ***y*** con ***plot(y)***
- Plotear ***plot(y)*** variando el tipo de gráficos: puntos, líneas, etc., con la opción ***type=***
- Igual pero que solo represente los valores entre x=2 y x=8
- Mostrar en el ploteo unos títulos para cada eje, un tipo principal y un subtítulo.
- Poner las líneas o puntos en colores, y variar el tamaño de los puntos o símbolos y la anchura de las líneas.
- Dibujar un gráfico de tarta, un gráfico de cajas (*boxplot*), histograma de frecuencias y de valores con el vector ***y.***

Salvar un gráfico a fichero

R, además de visualizarlos en pantalla, permite salvar los gráficos en ficheros de diversos formatos: ***png***, ***jpg***, ***pdf***, ***bmp***, ***tiff***, etc.

El método consiste en escribir, antes del ploteo, la orden con el tipo de formato de salida que queremos para el grafico, por ejemplo para obtener un fichero en formato ***png***:

png ('nombre_fichero.png')

después se pinta el grafico:
plot()

y terminar cerrando el dispositivo gráfico:
dev.off()

En el nombre del fichero se puede incluir la ruta completa si lo queremos dejar en otro lugar distinto del directorio de trabajo.

Si queremos un fichero ***jpg*** usaríamos ***jpeg ('nombre_fichero.jpg')***

Las órdenes admiten diversas opciones como la resolución, la anchura o la altura de la imagen. Estas opciones dependen del formato. Hay que mirar en la ayuda de cada una. ***help(png)***

Para el formato ***png*** se pueden usar:
png('nombre_fichero.png',res=130, width = 910, height = 690,units="px") donde se obtendrá una imagen de *910x690* pixeles con una resolución de *130*.

Para ficheros ***jpeg*** admite también *width* y *height* pero en vez de resolución se usa *quality* para elegir la compresión del ***jpg*** entre 0 y 100.

Ejemplo:

```
x <- sample(1:10)  # creamos un vector x con números aleatorios de 1 al 10
png('prueba.png')  # ponemos el formato de salida a png
plot(x)   # hacemos un gráfico sencillo con el vector x (no se abre la pantalla ni se crea el
            grafico todavía.
dev.off() # se crea en el directorio de trabajo un fichero prueba.png con el ploteo
```

Funciones gráficas de bajo nivel

Son funciones para modificar los gráficos hechos con alguna función de alto nivel, para mejorar el gráfico con otros elementos más específicos u otros detalles mas finos, e incluso hacer un grafico desde cero (habría que empezar por abrir una ventana nueva de gráfico con ***plot.new()***). Si no hay ningún grafico abierto, antes hay que hacer un ploteo. Si se quiere un ploteo vacío, poner ***type="n"***, por ejemplo: ***plot(1:10,1:10,type="n").***

Las funciones de bajo nivel más usadas son las que se muestran a continuación. En algunos el par (*x*,*y*) puede ser un solo punto o a veces vectores de la misma longitud.

points(x,y)	Agrega puntos, se puede usar la opción (***type=)***
lines(x,y)	Igual que la anterior pero con líneas.
text(x, y, 'texto')	Agrega texto en la posición *x*,*y*
mtext(text, side=3, line=n, ...)	Agrega texto dado por *text* en la posición *side* y en la línea que se especifique.
segments(x0,y0,x1,y1)	Agrega una línea desde el punto (*x0*,*y0*) hasta el (*x1*,*y1*)
arrows(x0, y0,x1, y1, angle=30)	Igual que el anterior pero con flechas y un ángulo.
abline(h=y)	Dibuja una línea horizontal en la posición del eje y = *y*
abline(v=x)	Dibuja una línea vertical en la posición del eje x = *x*
rect(x1, y1, x2, y2)	Rectángulo en la posición indicada
polygon(x, y)	Polígono uniendo los puntos dados en *x*,*y*
legend(x, y, legend)	Dibuja una leyenda en (*x*,*y*) con los símbolos dados en legend ('*','.','P', etc)
title()	Agrega un titulo y si se quiere un subtitulo
axis(side,vect)	Agrega un eje en la parte inferior (*side=1*), izquierda (2), superior (3), o derecha (4); vect (opcional) da la abscisa (u ordenada) donde se deben dibujar los marcadores ('*tick marks*') del eje

La función ***text*** sirve también para incluir como un dibujo en el grafico fórmulas matemáticas si se escribe la función expresión. Por ejemplo:

```
text(x, y, expression(p == over(1, 1+e^-(beta*x+alpha))))
```

se vería en la grafica como la siguiente ecuación en el punto de coordenadas (x,y):

$$p = \frac{1}{1 + e^{-(\beta x + \alpha)}}$$

Ejercicios 1.2:

- Crear un ploteo vacío con 10 puntos en el eje x y 10 en el eje y
- Pintar puntos, líneas, textos, segmentos, flechas, líneas horizontales y verticales, rectángulos, etc.
- Opcional: escribir una fórmula de ejemplo en un punto del gráfico.

Parámetros gráficos

Además de la utilización de comandos de bajo nivel, la presentación de gráficos se puede mejorar con parámetros gráficos adicionales.

Estos se pueden utilizar como opciones de funciones graficas (pero no funciona para todas), o usando la función ***par*** para cambiar de manera permanente opciones generales en los gráficos. La lista completa de parámetros gráficos se puede ver con ***?par***.

Cualquier cambio que hagamos en un parámetro grafico general, *se aplicara a todos los gráficos siguientes* por lo es conveniente antes de cambiar algún parámetro grafico hacer una copia de los valores actuales y cuando terminemos restaurar la copia inicial. Esto se hace con:
old.par <- par(no.readonly = TRUE) o simplemente ***old.par <- par()*** *para salvarlos*.

Cuando queramos *recuperar* otra vez los que teníamos al principio: ***par(old.par)***

De cualquier forma si salimos y volvemos a entrar de R se pondrán de nuevo los gráficos por defecto.

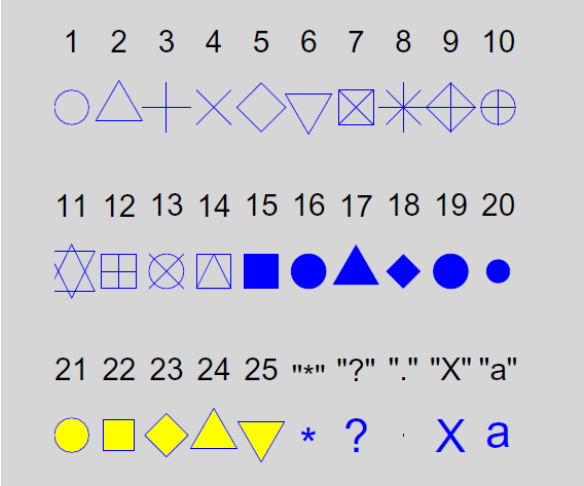
Por ejemplo, el siguiente comando: ***par(bg="yellow")*** dará como resultado que todos los gráficos siguientes tengan el fondo de color amarillo.

Mostraremos algunos de ellos

adj	*controla la justificación del texto (0 justificado a la izquierda, 0.5 centrado, 1 justificado a la derecha)*
bg	*especifica el color del fondo (ej. **: bg='red'**) La lista de colores disponibles se puede ver con **colors()***
bty	*controla el tipo de caja que se dibuja alrededor del gráfico: ej: ¨o", "l" (la caja se parece a su respectivo carácter); si **bty="n"** no se dibuja la caja*
cex	*un valor que controla el tamaño del texto y símbolos con respecto al valor por defecto; los siguientes parámetros tienen el mismo control para números en los ejes: **cex.axis**, títulos en los ejes: **cex.lab**, el título principal: **cex.main**, y el subtítulo: **cex.sub***
col	*color de los símbolos ploteados; para controlar el color en otros elementos: **col.axis, col.lab, col.main** y **col.sub***
font	*un entero para el estilo del texto (**1: normal, 2: cursiva, 3: negrilla, 4: negrilla cursiva**); como en cex existen**: font.axis, font.lab, font.main y font.sub***
las	*un entero que controla la orientación de los caracteres en los ejes **(0: paralelo a los ejes, 1: horizontal,2: perpendicular a los ejes, 3: vertical)***
lty	*un entero o carácter que controla el tipo de las líneas; **(1: solida, 2: quebrada, 3: punteada, 4: punto-línea, 5: línea larga-corta, 6: dos líneas cortas)***
lwd	*un numero que controla la anchura de las líneas (def=1)*
mar	*un vector con 4 valores numéricos que controla el espacio entre los ejes y el borde de la grafica en la forma **c(inferior, izquierda, superior, derecha)**; los valores por defecto son c(5.1, 4.1, 4.1, 2.1)*

mgp	*un vector con 3 valores numéricos que controla el espacio entre los ejes y las etiquetas (labels) de los ejes (primer numero), o entre los ejes y los números en cada eje (segundo numero).*
pch	*controla el tipo de símbolo, ya sea un entero entre 1 y 25, o un carácter (Ver figura abajo)*
ps	*un entero que controla el tamaño (en puntos) de textos y símbolos*
xaxt	*si **xaxt="n"** el eje x se coloca pero no se muestra (útil en conjunción con **axis**)*
yaxt	*si **yaxt="n"** el eje y se coloca pero no se muestra (útil en conjunción con **axis**)*

Tipos de símbolos con los que pintar los gráficos:



- *Figura 2: Los símbolos gráficos en R (pch=1:25). Los colores se obtuvieron con las opciones col="blue", bg=" yellow"; la segunda opción tiene efecto solo sobre los símbolos 21–25. Se puede usar cualquier carácter (pch="*", "?", ".", . . .).*