Introducción a graficación en R

Instructor: Felipe González, ITAM. Septiembre, 2008

Resumen. Gráficos de datos en R: Cómo producirlos, refinarlos, y exportarlos (tradicional y lattice). GGobi para exploración de datos multivariados.

Maneras de hacer gráficos en R. Veremos ejemplos del paquete tradicional (para producir gráficas estadísticas estándar), de lattice (más enfocado a datos multivariados) y de ggplot2 (una propuesta reciente). En cuanto a los principios de graficación, todas las alternativas se basan principalmente en el trabajo de Cleveland ([2],[3]). Otra referencia importante es [7].

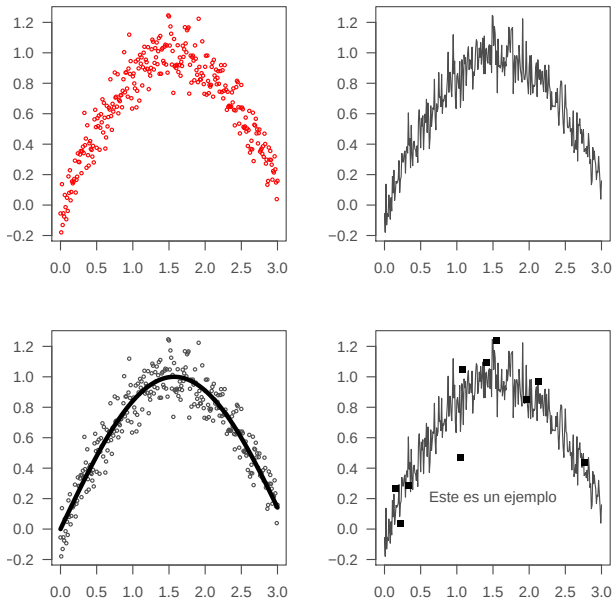
Los gráficos en R se producen en *dispositivos*, que pueden ser ventanas de nuestro sistema operativo o archivos (svg, pdf, png, etc.). El orden que se sigue para hacer una gráfica es: abrir un dispositivo, establecer los parámetros de graficación, y luego hacer uno o varios pasos donde graficamos nuestros datos. Cuando trabajamos interactivamente (enviando comandos desde la consola de R), si no abrimos explícitamente un dispositivo, R abre una ventana de gráficos del sistema.

Gráficos simples. Para producir las gráficas más tradicionales y simples en R (ver [5]) usamos la función `plot`. Los parámetros de graficación que usa `plot` se pueden ver con la función `par`:

```
> head(names(par()),10)
 [1] "xlog"     "ylog"     "adj"      "ann"      "ask"      "bg"
 [7] "bty"      "cex"      "cex.axis" "cex.lab"
```

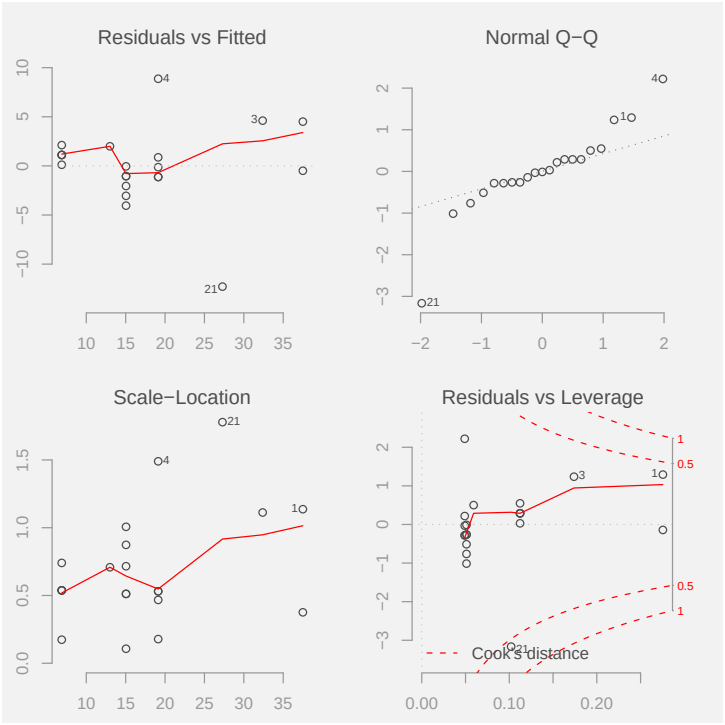
Usando `par()` podemos cambiar estos parámetros. Además de la función general `plot`, podemos agregar otros objetos con funciones como `points`, `line`, y otras.

```
> #región de 2x2 gráficas, definir márgenes
> par(mfrow=c(2,2),mar=c(2.5,2.5,2.5,2.5))
> #Colores
> par(col.axis="gray30",fg="gray30",col="gray30")
>  #Arreglar etiquetas de eje vertical
> par(las=1)
> #Crear algunos datos
> x<-seq(from=0,to=3,by=0.01)
> z<-sin(x)
> y<-z+0.1*rnorm(length(x))
> #Graficar
> plot(x,y,cex=0.5,col="red")
> plot(x,y,type="l")
> plot(x,y,type="p",cex=0.5)
> lines(x,z,col="black",lwd=4)
> plot(x,y,type="l")
> points(x<-runif(10,0,3),y<-sin(x)+0.2*rnorm(10),pch=15,col="black")
> text(1.5,0.2,"Este es un ejemplo")
```



`plot` es polimórfica. Por ejemplo, si hacemos

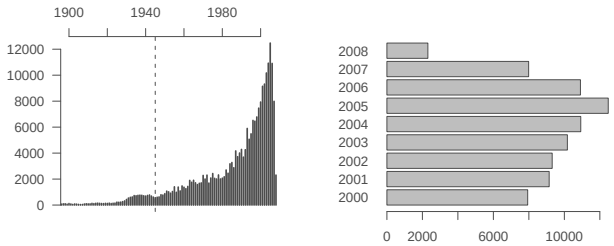
```
> par(bg = "gray95", mfrow = c(2, 2), mar = c(2.5, 2.5,
+     2.5, 2.5), bty = "n")
> par(col.axis = "gray60", fg = "gray60", col = "gray30")
> modelo_lineal <- lm(stack.loss ~ Air.Flow, data = stackloss)
> class(modelo_lineal)
[1] "lm"
> plot(modelo_lineal)
```



En la gráfica de la izquierda cambiamos la presentación del eje horizontal para graficar un objeto `table`:

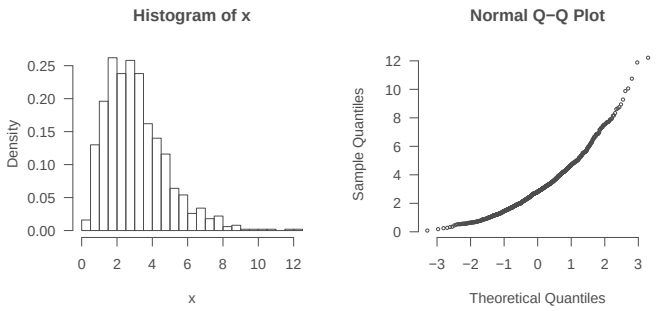
```
> par_orig <- par(mfrow = c(1, 2), col.axis = "gray30",
+     col.lab = "gray30", fg = "gray30", col.main = "gray30",
+     las = 1, bty = "n")
> mis_par <- par()
```

```
> par(mis_par)
> tabla1 <- table(ratings2$año)
> tail(tabla1, 10)

 1999  2000  2001  2002  2003  2004  2005  2006  2007  2008
 7463  7933  9144  9314 10170 10924 12473 10906  7987  2309

> class(tabla1)

[1] "table"

> plot(tabla1, xlim = c(1900, 2008), ylab = "", xaxt = "n")
> abline(v = 1945, lty = 2)
> axis(side = 3)
> par(xaxt = "s")
> barplot(tabla1[121:129], horiz = TRUE)
> par(par_orig)
```



donde adicionalmente mostramos una gráfica de barras. Otras gráficas estadísticas usuales son fáciles de producir, por ejemplo:

```
> par(mis_par)
> par(mfrow = c(1, 2))
> x <- rgamma(1000, shape = 3)
> hist(x, breaks = 20, freq = FALSE)
> qqnorm(x, cex = 0.5)
```



Ver más en [5].

El paquete *lattice*. El paquete *lattice* de R está hecho producir gráficas multivariadas, y está implementado según ideas de Cleveland ([2]). La referencia [1] es excelente para aprender de este paquete.

En lattice tenemos diversas funciones estándar de graficación: 1) univariadas: barchart, bwplot, densityplot, dotplot, histogram, qqmath, stripplot, 2) bivariadas: qq, xyplot, 3) trivariadas: levelplot, countourplot, cloud, wireframe, y 4) multivariadas: splom, parallel.

Comenzamos con un ejemplo simple de [2]:

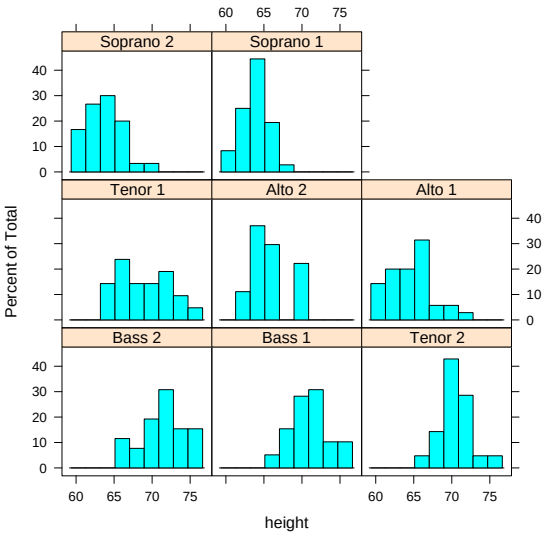
```
> library(lattice)
> names(singer)
```

```
[1] "height"     "voice.part"
```

```
> table(singer$voice.part)
```

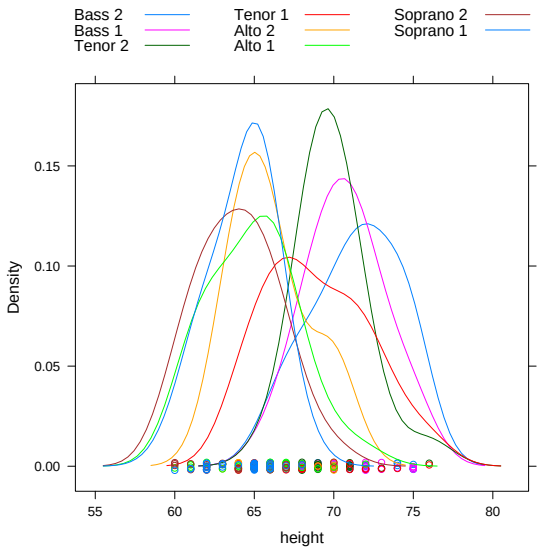
```
   Bass 2    Bass 1   Tenor 2   Tenor 1    Alto 2    Alto 1 Soprano 2
       26        39        21        21        27        35        30
Soprano 1
       36
```

```
> print(histogram(~height | voice.part, data = singer))
```



que podríamos también graficar con densidades, pero agrupando en lugar de condicionando:

```
> print(densityplot(~height, data = singer, groups = voice.part,
+     bw = 1.5, auto.key = list(columns = 3)))
```



Nótese que la primer gráfica se especifica con una fórmula: `height | voice.part`, que se lee: hacer las gráficas de `height` condicionadas a cada valor de *voice.part*. En el segundo caso, simplemente pedimos graficar `height`, pero agregamos el parámetro `group`. Más en general, en la siguiente fórmula

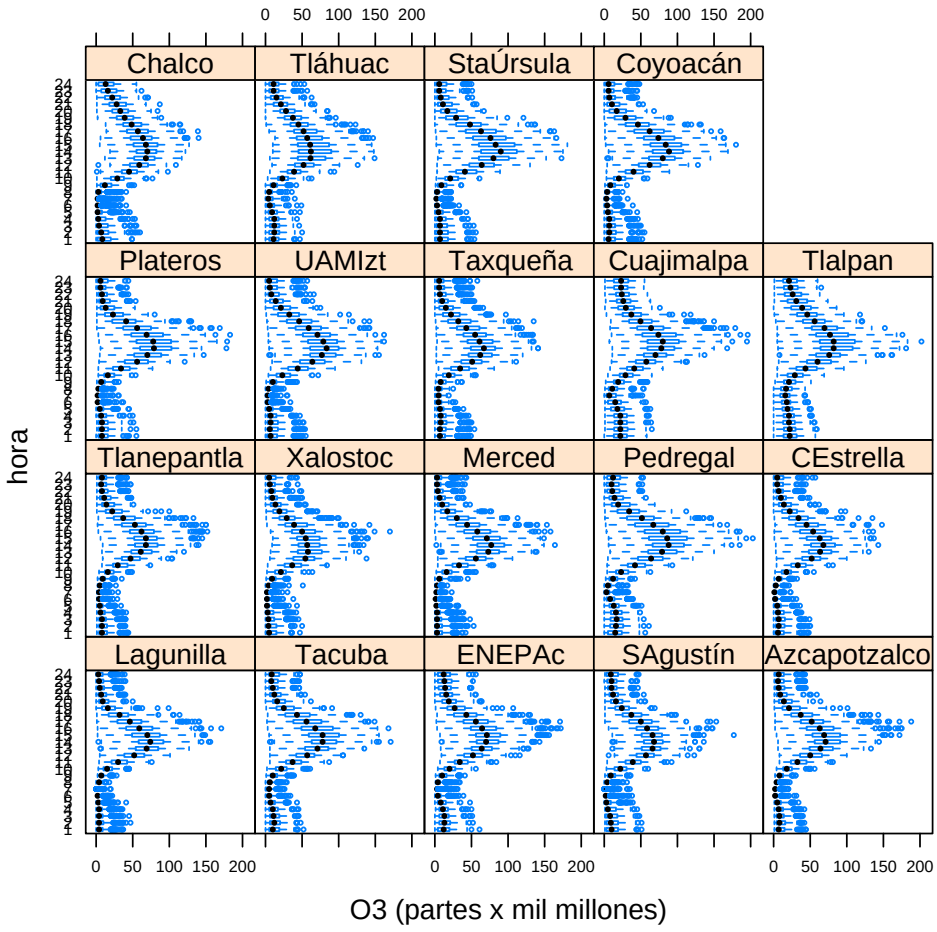
$$a + b + c \sim d + e | f + g + h$$

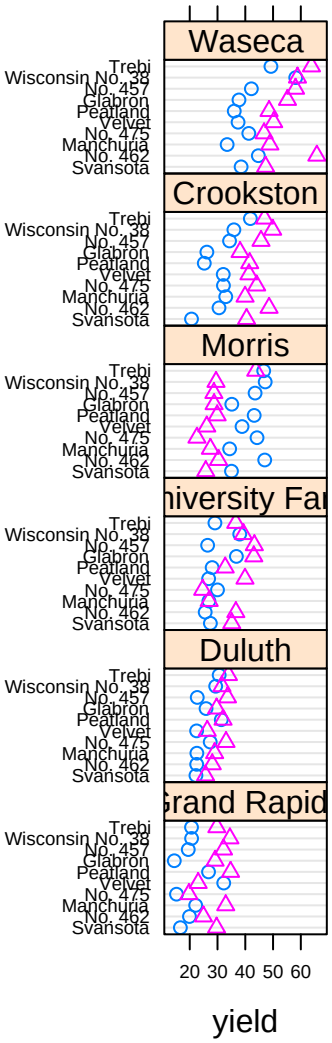
queremos decir: graficar las variables a, b y c contra d y e, condicionado a todos los posibles cruces de valores de f,g y h, posiblemente separando por grupos de una variable adicional. Otro ejemplo, usando diagramas de caja y brazos (en donde también utilizamos `trellis.par.set()`, el equivalente de `par`:

```
> library(reshape)
> load("./datos/ozono_2007.rdata")
> names(ozono)<-c("fecha","hora","Lagunilla","Tacuba","ENEPAc","SAgustín","Azcapotz
> ozono[1:4,1:4]
```

```
     fecha hora Lagunilla Tacuba
1 01/01/07    1      0.008  0.006
2 01/01/07    2      0.002  0.007
3 01/01/07    3      0.002  0.007
4 01/01/07    4      0.002  0.009
```

```
> ozono_m<-melt(ozono,id=c(1,2))
> trellis.par.set(list(   axis.text=list(cex=0.6) , plot.symbol=list(cex=0.3) ))
> print(bwplot(hora~1000*value|variable,data=ozono_m,cex=0.3,xlab="O3 (partes x mi
```





Los datos son de la red automática de monitoreo atmosférico del DF (SIMAT). Los *dotplots* son también útiles para gráficar tablas: ([2],[1]):

```
> trellis.par.set(list(axis.text = list(cex = 0.6)))
> print(dotplot(variety ~ yield | site, data = barley,
+     layout = c(1, 6), aspect = c(0.7), groups = year,
+     pch = 1:2))
```

Estos gráficos son muy flexibles. Las funciones que se encargan de dibujar cada panel se pueden modificar (o definir de cero) para agregar otros elementos ([1]).

Exportando gráficas. Si queremos incluir nuestras gráficas en otros documentos, hay que abrir los dispositivos correspondientes. El formato más conveniente en general el *svg*.

```
> library(RSvgDevice)
> devSVG(file = "salida.svg")
> print(plot(rnorm(100)))
NULL
> dev.off()
pdf
  2
```

El gráfico resultante es vectorial y puede cualquiera de sus aspectos puede modificarse con programas como *Illustrator* o *Inkscape*. Se puede exportar también a png, por ejemplo:

```
> png(file = "salida.png")
> print(plot(rnorm(100)))
NULL
> dev.off()
pdf
  2
```

Referencias

[1] D. Sarkar, Lattice: Multivariate Visualization with R, Springer, 2008.

[2] W. S. Cleveland, Visualizing Data, Hobart Press, 1993.

[3] W. S. Cleveland, The Elements of Graphing Data, Hobart Press, 1994.

[4] H. Wickham, ggplot2, por aparecer en Springer, 2009, `http://had.co.nz/ggplot2`.

[5] P. Murrell, R Graphics, Chapman & Hall, 2005.

[6] W.N. Venables, and B.D. Ripley, Modern Applied Statistics with S, 4th. ed, Springer, 2003.

[7] E. R. Tufte, The Visual Display of Quantitative Information, Graphic Press, 2nd. ed, 2001.