

Visualizing multivariate data using lattice and direct labels

<http://directlabels.r-forge.r-project.org>

Toby Dylan Hocking
toby.hocking@inria.fr

15 October 2009

Outline

The lattice system

Adding direct labels using the lattice`l` package

Brief history of lattice

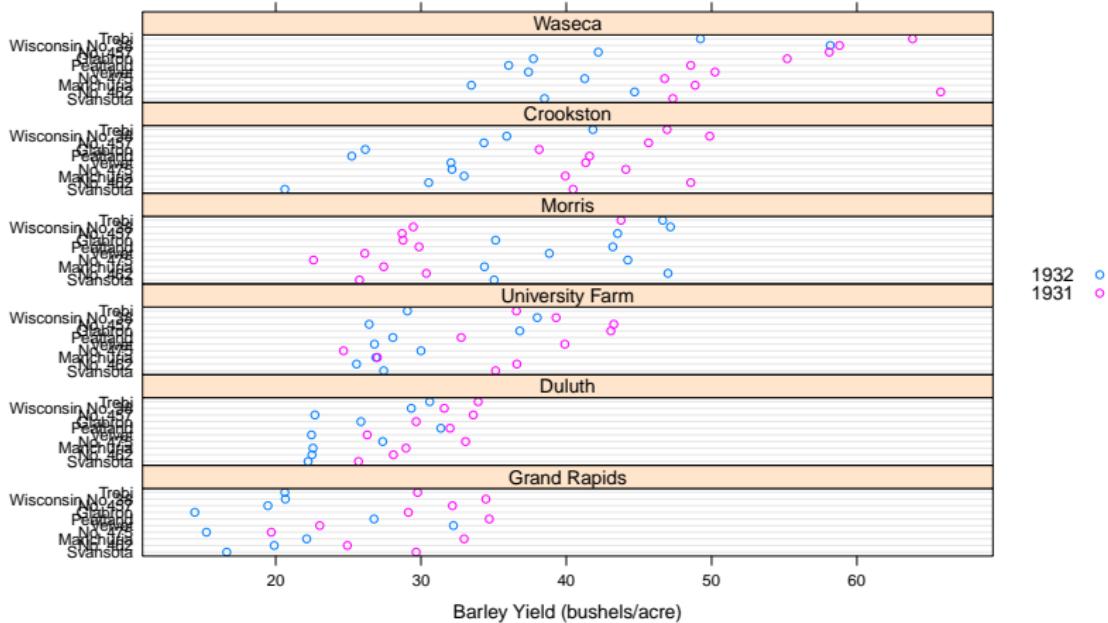
- ▶ Bill Cleveland, *Visualizing Data* (1993).
- ▶ Bill Cleveland, Rick Becker, Bell Labs, 1990s: trellis graphics system for S:
<http://cm.bell-labs.com/cm/ms/departments/sia/project/trellis/>
- ▶ Deepayan Sarkar, 2000s: the lattice package for R.
- ▶ Deepayan Sarkar (2008) Lattice: Multivariate Data Visualization with R, Springer.

Installing the required packages

- ▶ I used R 2.9.2 for the following examples.
- ▶ lattice is preinstalled with R.
- ▶ `library(lattice)`
- ▶ `install.packages(c("latticeExtra","latticedl"))`
- ▶ `library(latticeExtra)`
- ▶ `library(latticedl)`

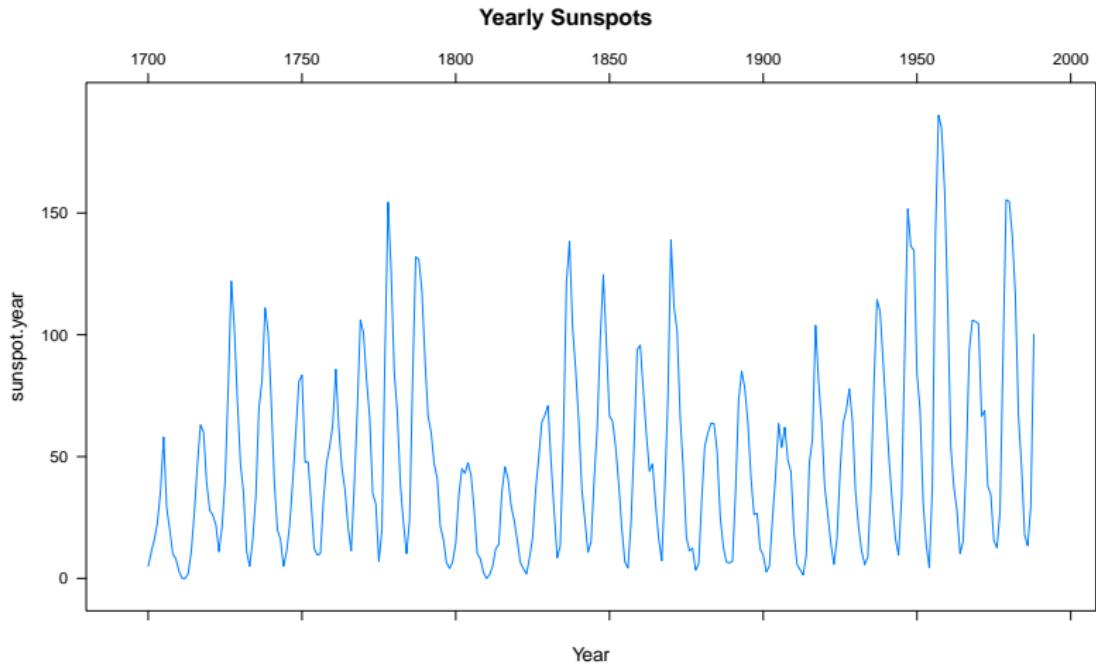
Lattice allows easy visualization of many variables

```
> options(width = 55)
> library(lattice)
> dotplot(variety ~ yield | site, data = barley,
+           groups = year, auto.key = list(space = "right"),
+           layout = c(1, 6), xlab = "Barley Yield (bushels/acre)")
```



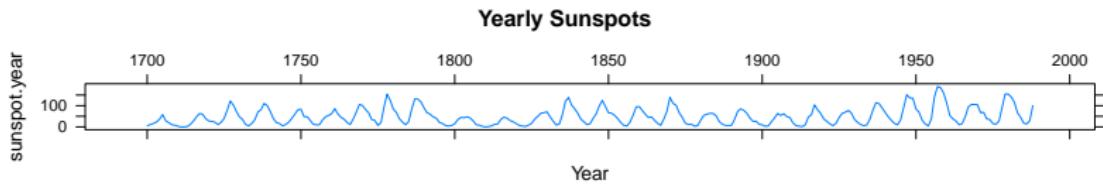
Aspect ratio in scatterplots is important

```
> xyplot(sunspot.year ~ 1700:1988, xlab = "Year",
+         type = "l", scales = list(x = list(alternating = 2)),
+         main = "Yearly Sunspots")
```



Lattice also automatically calculates aspect ratio for optimal decoding

```
> xyplot(sunspot.year ~ 1700:1988, xlab = "Year",
+         type = "l", scales = list(x = list(alternating = 2)),
+         main = "Yearly Sunspots", aspect = "xy")
```



Load a data set

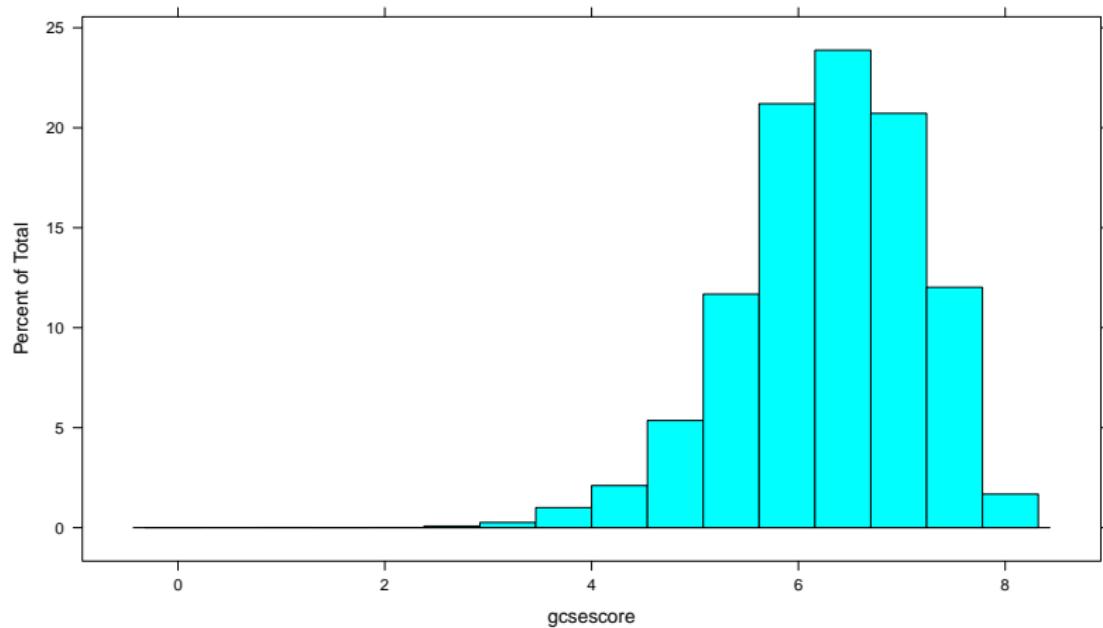
```
> data(Chem97, package = "mlmRev")
> head(Chem97)

  lea school student score gender age gcsescore
  1   1       1       1     4      F   3    6.625
  2   1       1       2    10      F  -3    7.625
  3   1       1       3    10      F  -4    7.250
  4   1       1       4    10      F  -2    7.500
  5   1       1       5     8      F  -1    6.444
  6   1       1       6    10      F   4    7.750

  gcsecnt
  1 0.3393157
  2 1.3393157
  3 0.9643157
  4 1.2143157
  5 0.1583157
  6 1.4643157
```

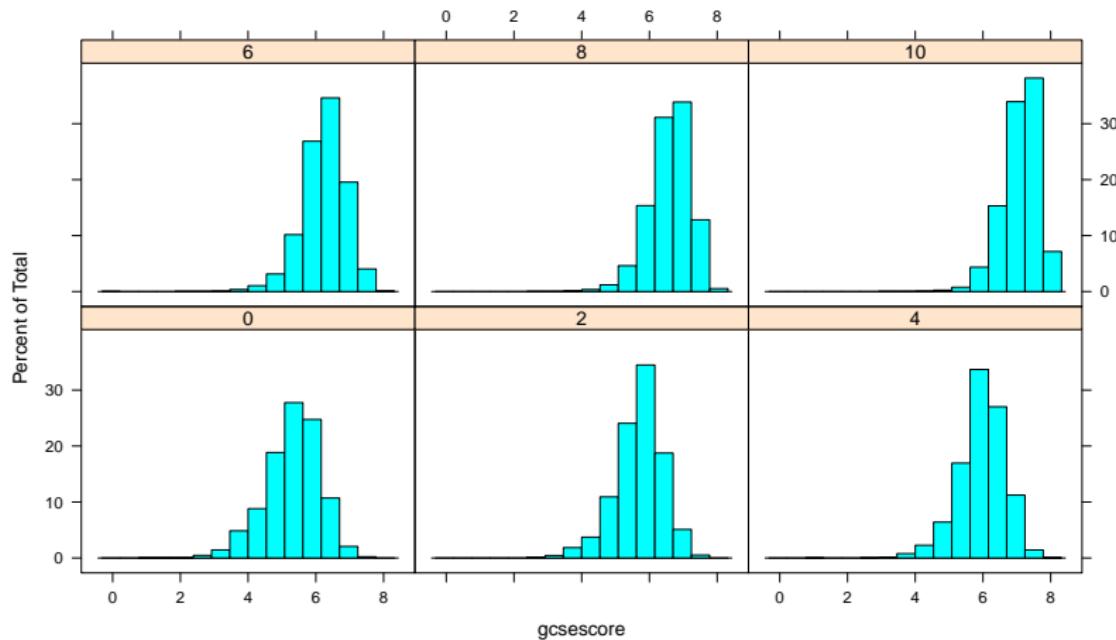
Simple histogram

```
> histogram(~gcse score, Chem97)
```



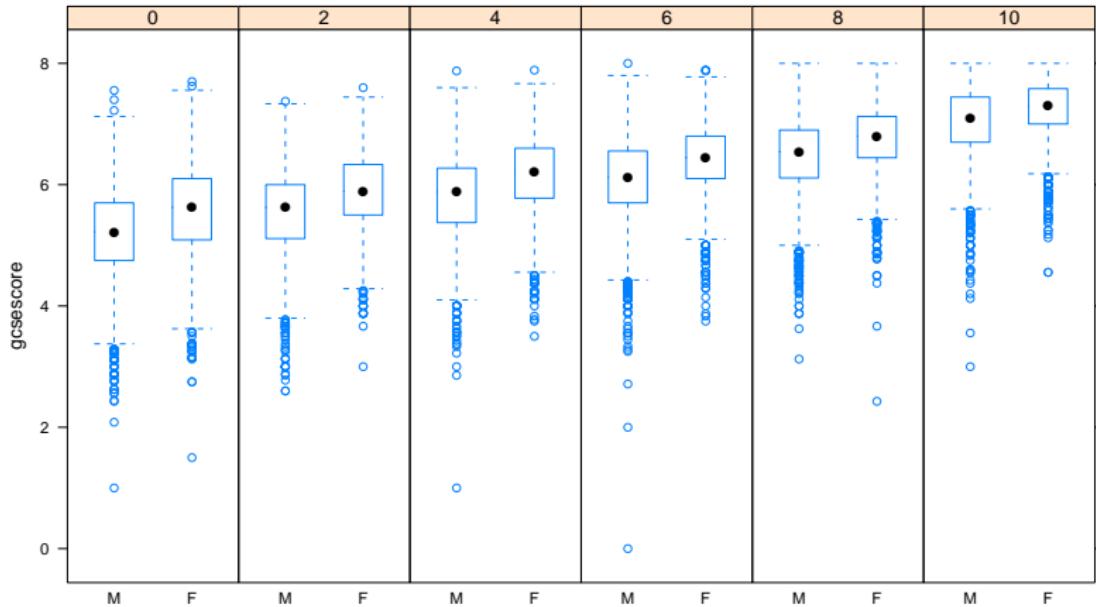
Histograms conditional on a categorical variable

```
> histogram(~gcsescore | factor(score),  
+           Chem97)
```



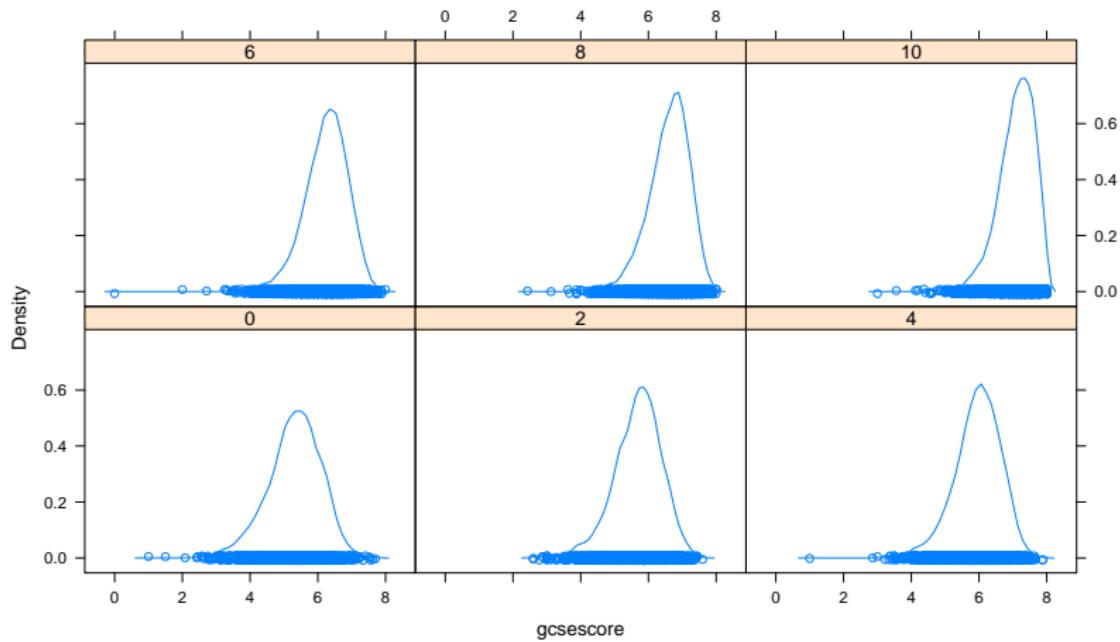
Box and whisker plots

```
> bwplot(gcsescore ~ gender | factor(score),  
+         Chem97, layout = c(6, 1))
```



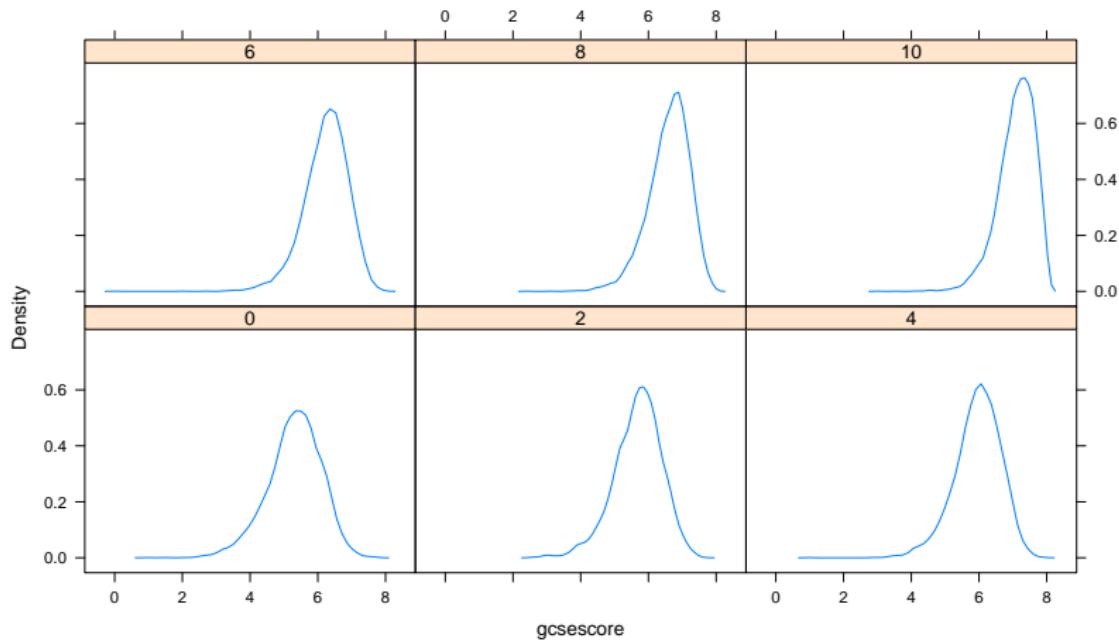
Conditioned plots of kernel density estimates

```
> densityplot(~gcsescore | factor(score),  
+ Chem97)
```



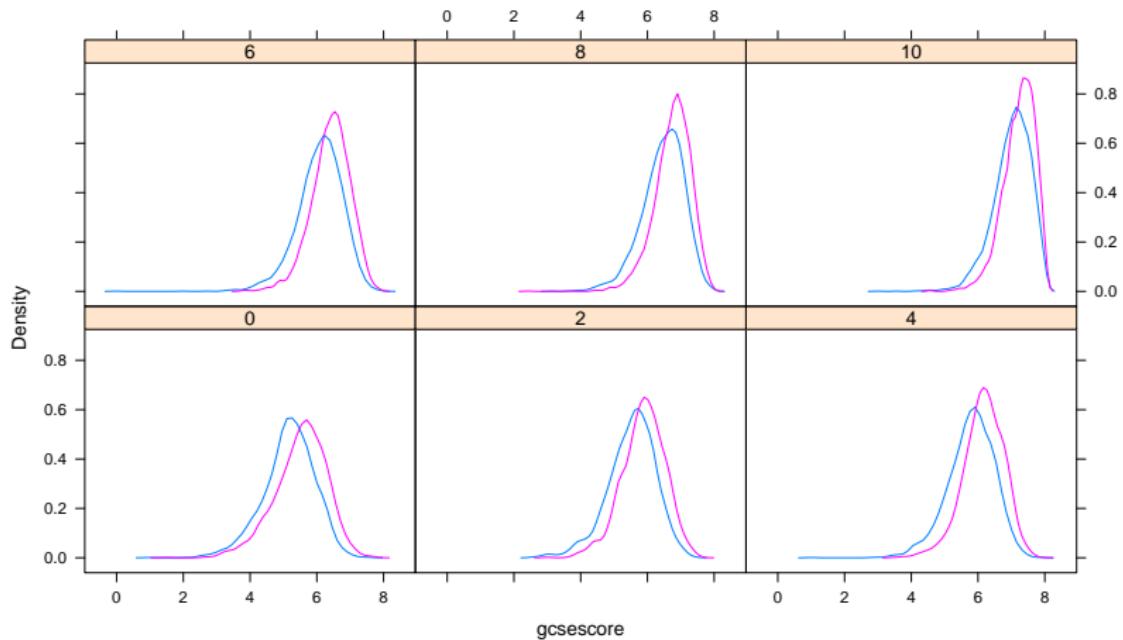
Hide the actual points with the plot.points argument

```
> densityplot(~gcse score | factor(score),  
+ Chem97, plot.points = FALSE)
```



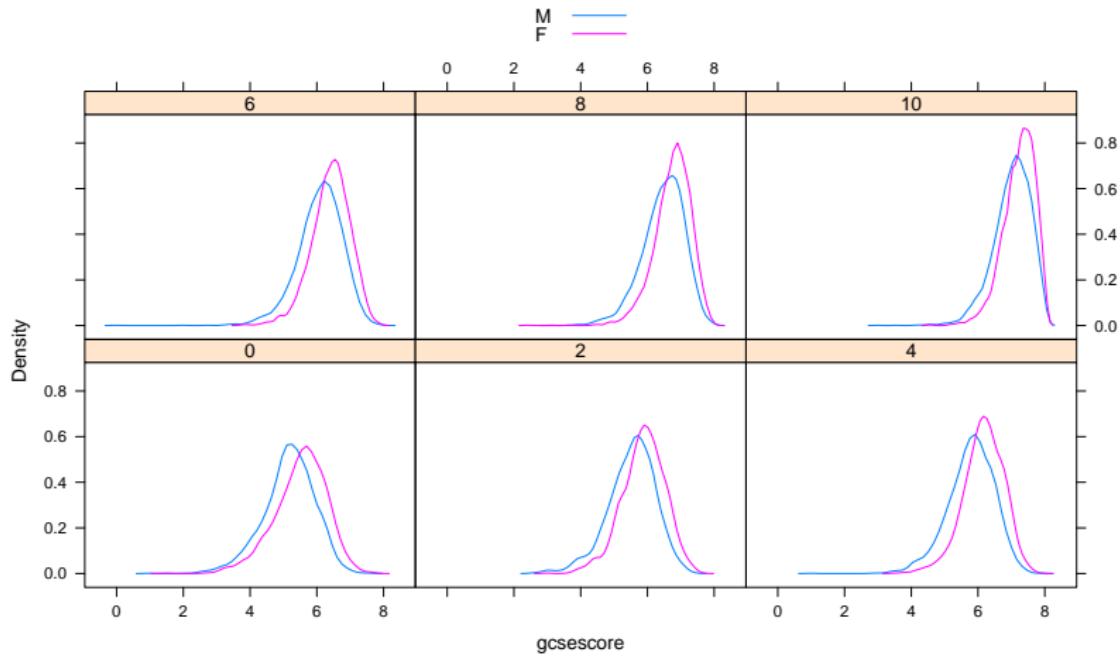
Conditioned and grouped density plots

```
> densityplot(~gcsescore | factor(score),  
+   Chem97, plot.points = FALSE, groups = gender)
```



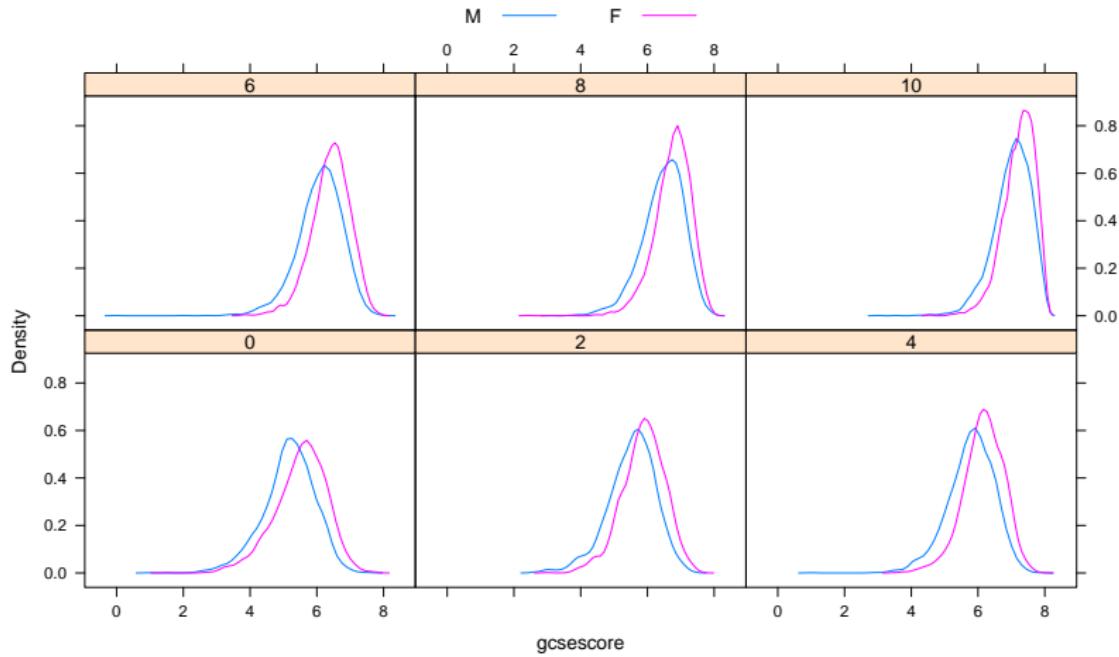
Add a legend with the auto.key argument

```
> densityplot(~gcsescore | factor(score),  
+             Chem97, plot.points = FALSE, groups = gender,  
+             auto.key = list())
```



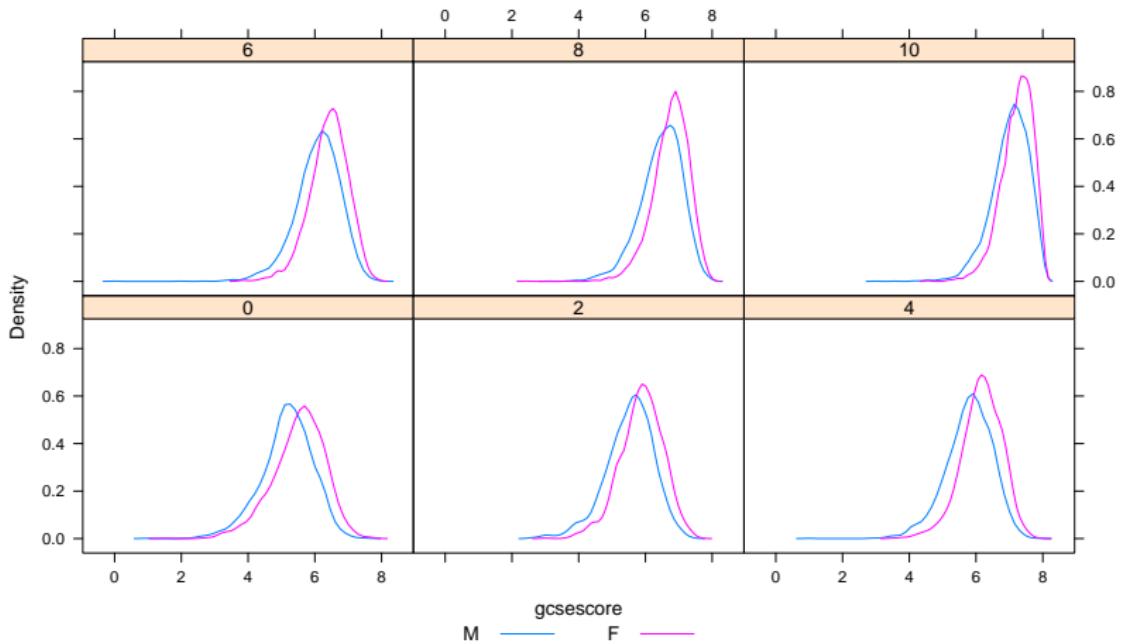
Legend layout with the columns argument

```
> densityplot(~gcsescore | factor(score),  
+             Chem97, plot.points = FALSE, groups = gender,  
+             auto.key = list(columns = 2))
```



Legend positioning with the space argument

```
> densityplot(~gcsescore | factor(score),  
+             Chem97, plot.points = FALSE, groups = gender,  
+             auto.key = list(columns = 2, space = "bottom"))
```



Show all default settings

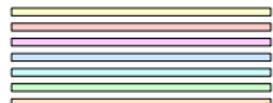
```
> show.settings()
```



superpose.symbol



superpose.line



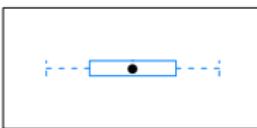
strip.background



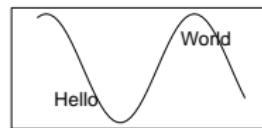
strip.shingle



dot[symbol, line]



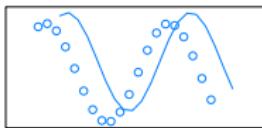
box[dot, rectangle, umbrella]



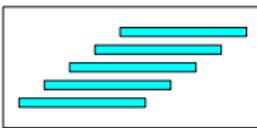
add[line, text]



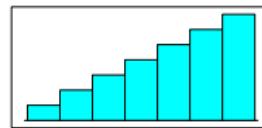
reference.line



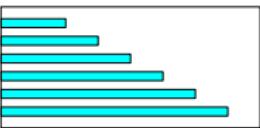
plot[symbol, line]



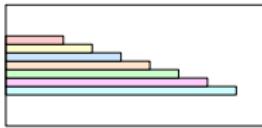
plot.shingle[plot.polygon]



histogram[plot.polygon]



barchart[plot.polygon]



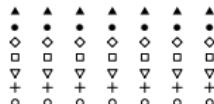
superpose.polygon



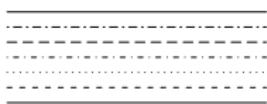
regions

Show settings good for printout

```
> show.settings(standard.theme(color = FALSE))
```



superpose.symbol



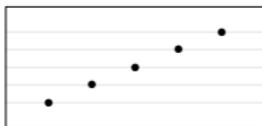
superpose.line



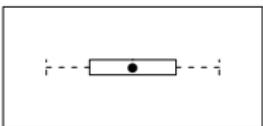
strip.background



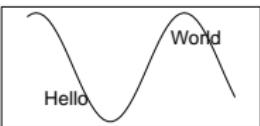
strip.shingle



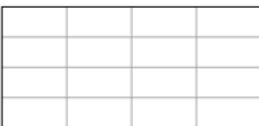
dot[symbol, line]



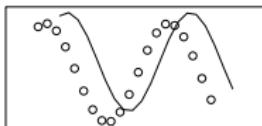
box.[dot, rectangle, umbrella]



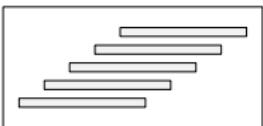
add.[line, text]



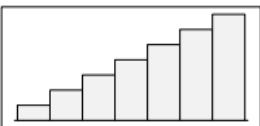
reference.line



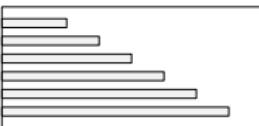
plot.[symbol, line]



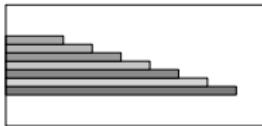
plot.shingle[plot.polygon]



histogram[plot.polygon]



barchart[plot.polygon]



superpose.polygon



regions

Change the settings

```
> br <- simpleTheme(col = c("black", "red"))
> show.settings(br)
```



superpose.symbol



superpose.line



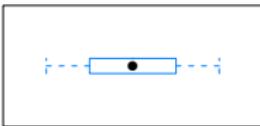
strip.background



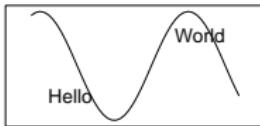
strip.shingle



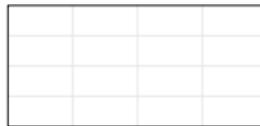
dot[symbol, line]



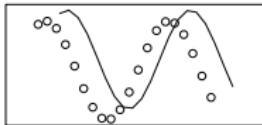
box.[dot, rectangle, umbrella]



add.[line, text]



reference.line



plot[symbol, line]



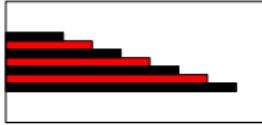
plot.shingle[plot.polygon]



histogram[plot.polygon]



barchart[plot.polygon]



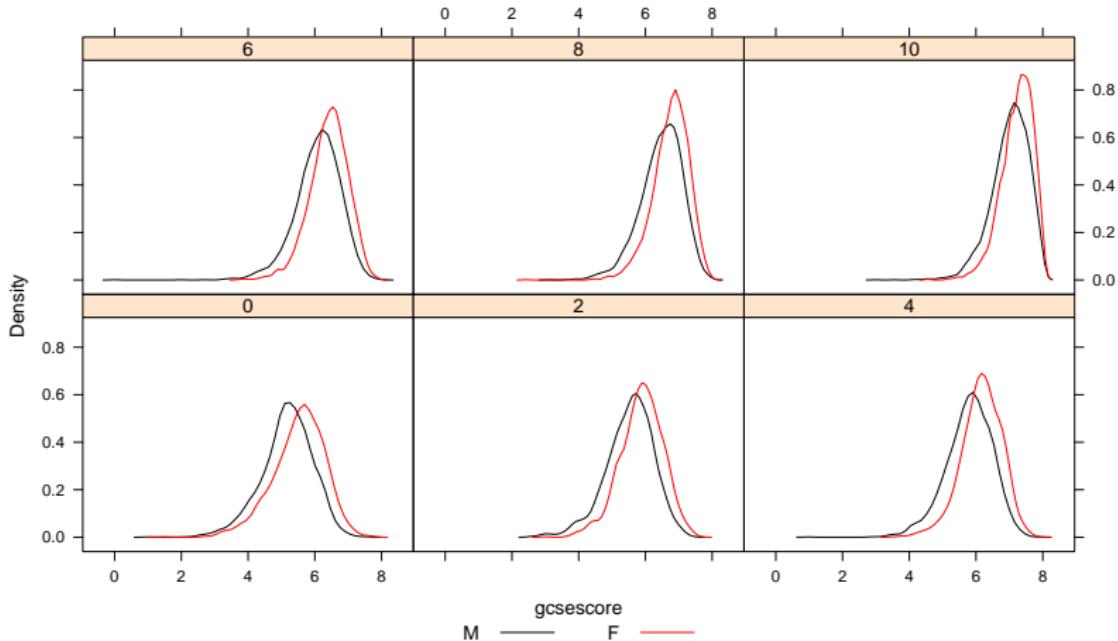
superpose.polygon



regions

Change group colors with par.settings

```
> densityplot(~gcsescore | factor(score),  
+   Chem97, plot.points = FALSE, groups = gender,  
+   auto.key = list(columns = 2, space = "bottom"),  
+   par.settings = br)
```



Load a tabular data set

```
> print(VADeaths)
```

	Rural Male	Rural Female	Urban Male	Urban Female
50-54	11.7	8.7	15.4	8.4
55-59	18.1	11.7	24.3	13.6
60-64	26.9	20.3	37.0	19.3
65-69	41.0	30.9	54.6	35.1
70-74	66.0	54.3	71.1	50.0

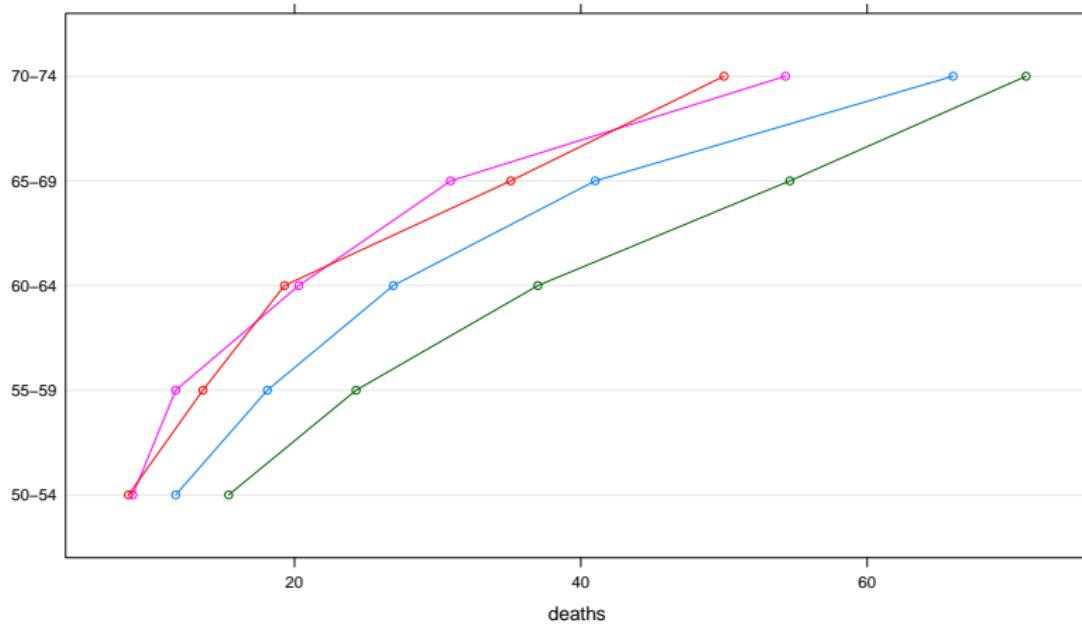
Convert to data frame to work with lattice

```
> vad <- as.data.frame.table(VADeaths)
> names(vad) <- c("age", "demographic", "deaths")
> head(vad)
```

	age	demographic	deaths
1	50-54	Rural Male	11.7
2	55-59	Rural Male	18.1
3	60-64	Rural Male	26.9
4	65-69	Rural Male	41.0
5	70-74	Rural Male	66.0
6	50-54	Rural Female	8.7

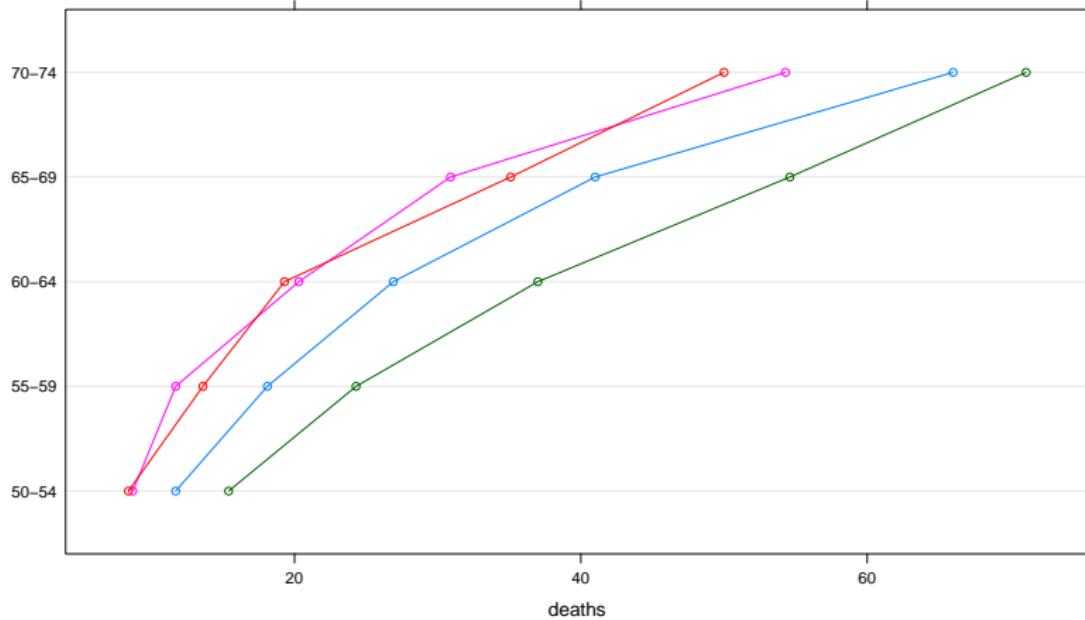
Grouped dotplots work well for these data

```
> dotplot(age ~ deaths, vad, groups = demographic,  
+           type = "o")
```



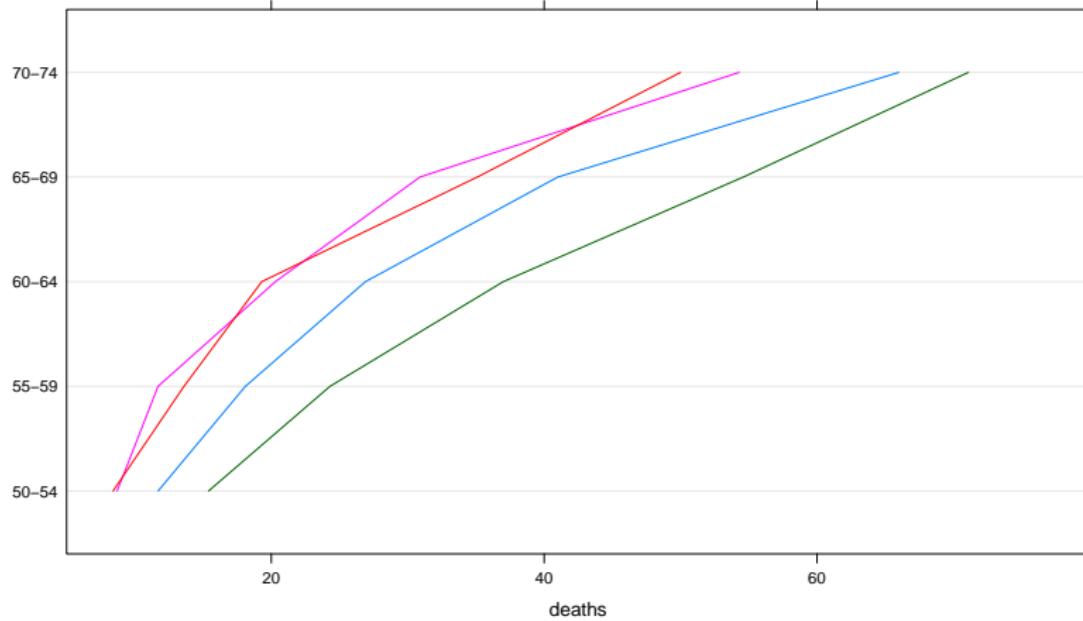
Plots can be saved as R objects

```
> dots <- dotplot(age ~ deaths, vad, groups = demographic,  
+ type = "o")  
> dots
```



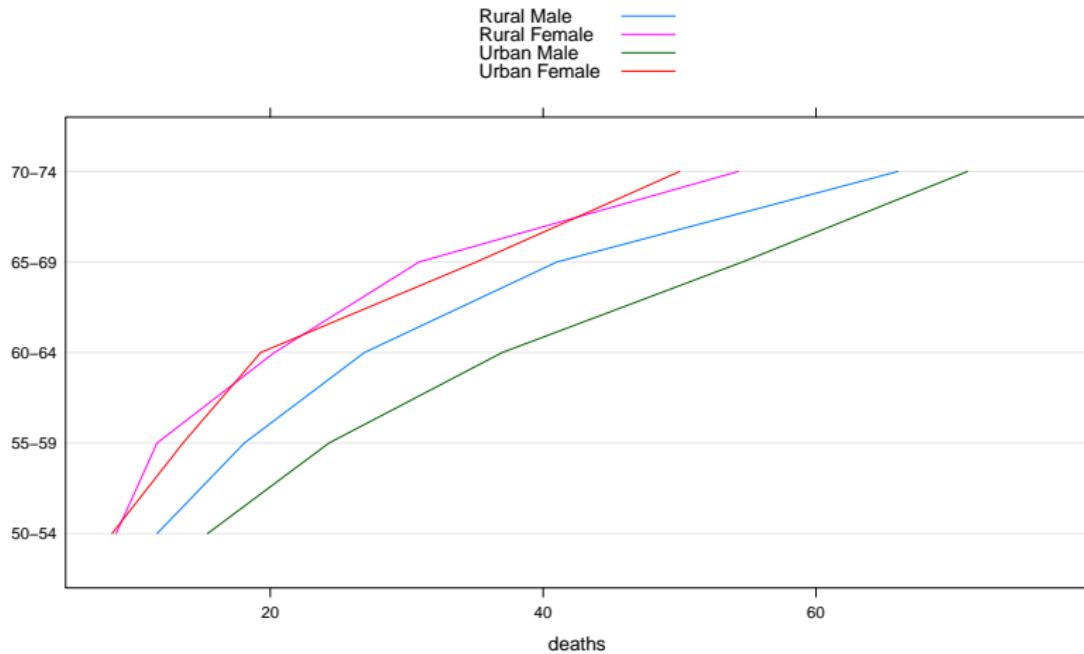
Saved plots can be updated later

```
> dots2 <- update(dots, type = "l", xlim = c(5,
+     80))
> dots2
```



Add a confusing legend ... how can we label more intuitively?

```
> update(dots2, auto.key = list(points = FALSE,  
+      lines = TRUE))
```



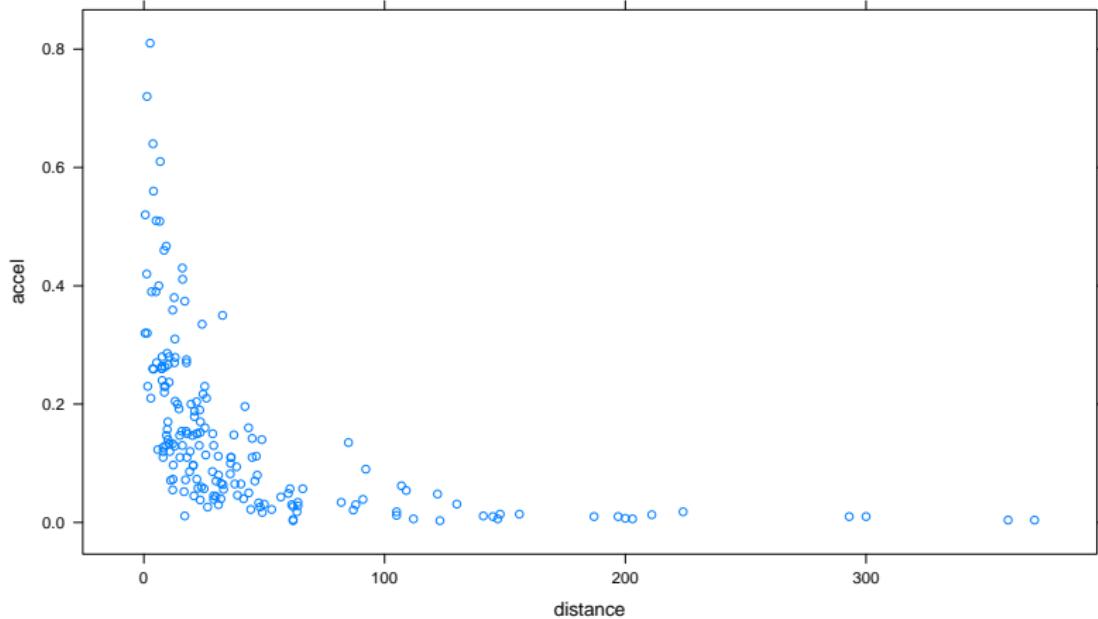
Load some earthquake measurements

```
> data(Earthquake, package = "nlme")
> head(Earthquake)
```

	Quake	Richter	distance	soil	accel
132	20	5	7.5	1	0.264
133	20	5	8.8	1	0.263
134	20	5	8.9	1	0.230
135	20	5	9.4	1	0.147
136	20	5	9.7	1	0.286
137	20	5	9.7	1	0.157

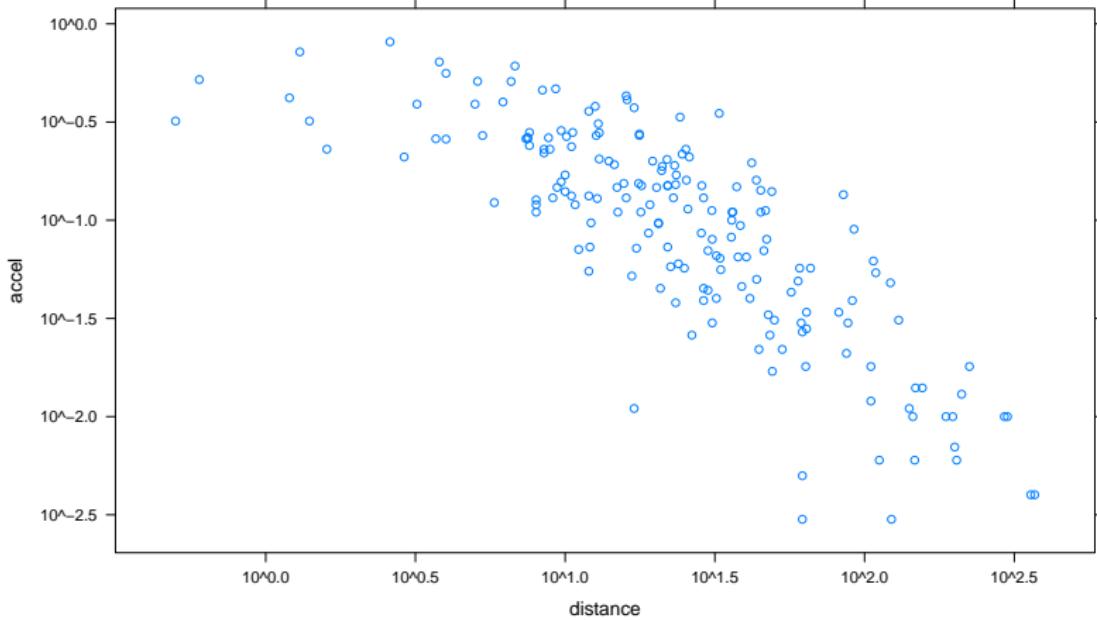
Scatterplot with xyplot

```
> xyplot(accel ~ distance, Earthquake)
```



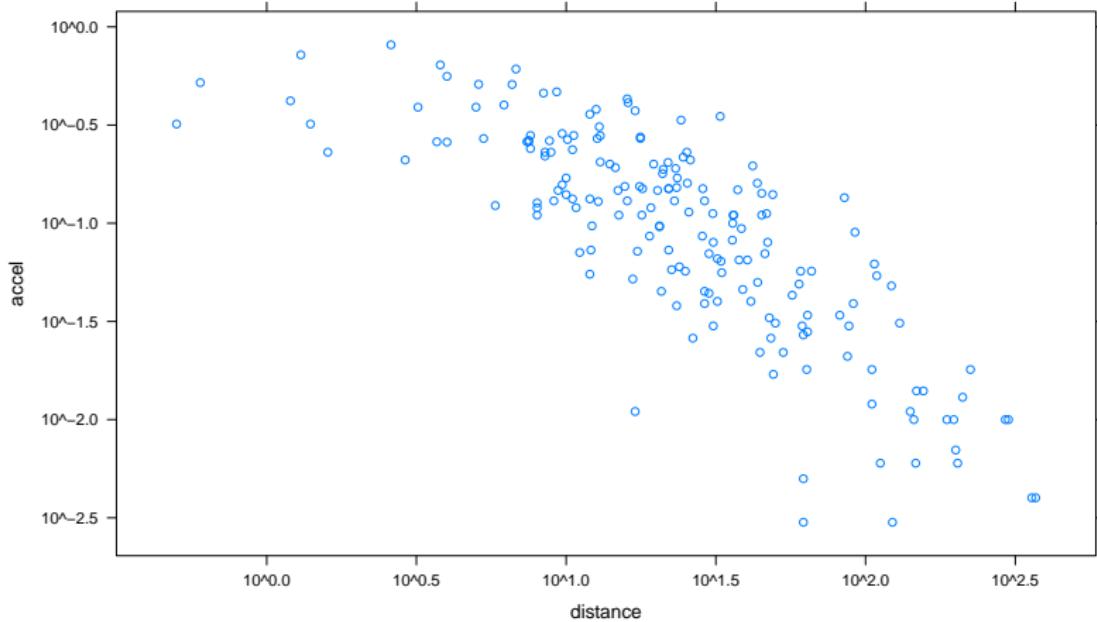
Log scales with scales argument

```
> xyplot(accel ~ distance, Earthquake,  
+         scales = list(log = TRUE))
```



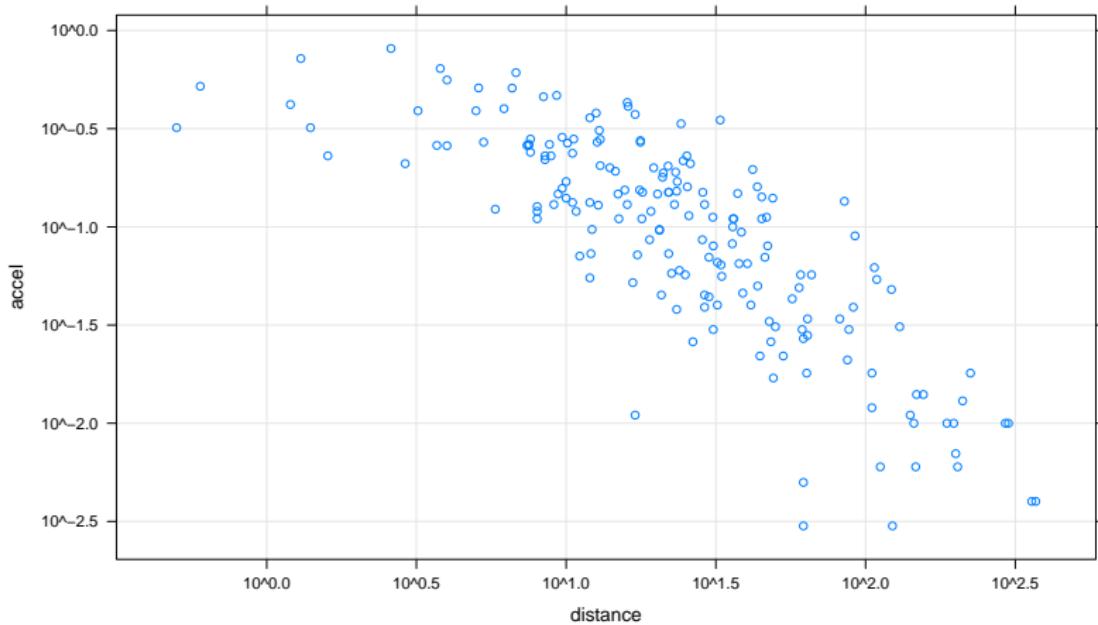
Type "p" is the default

```
> xyplot(accel ~ distance, Earthquake,  
+         scales = list(log = TRUE), type = c("p"))
```



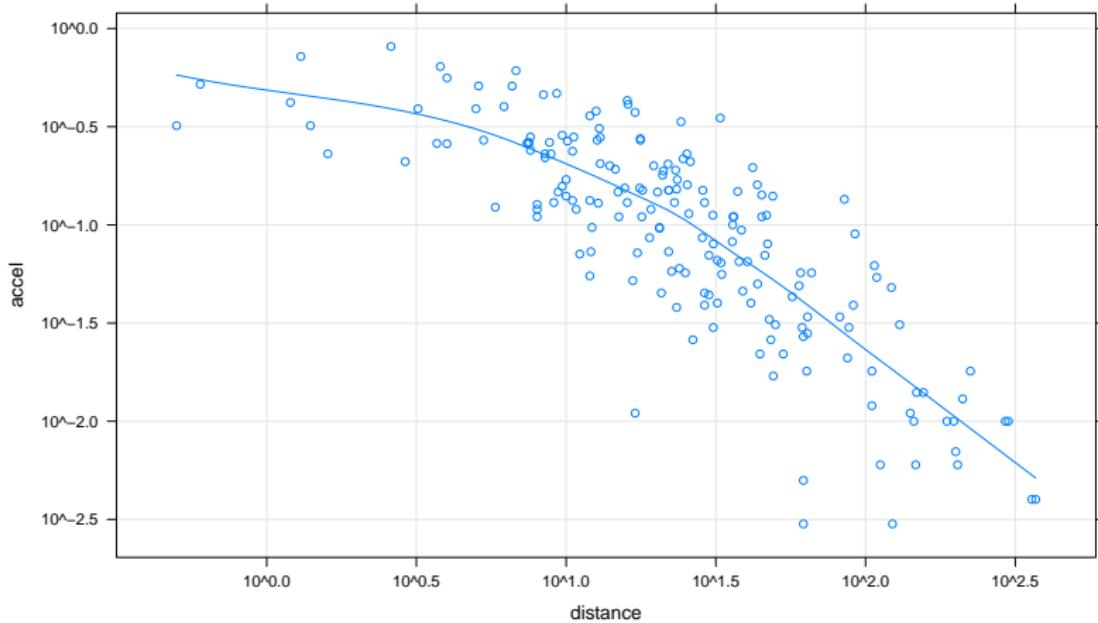
Type "g" adds a grid

```
> xyplot(accel ~ distance, Earthquake,  
+         scales = list(log = TRUE), type = c("p",  
+         "g"))
```



Type "smooth" adds a smooth line

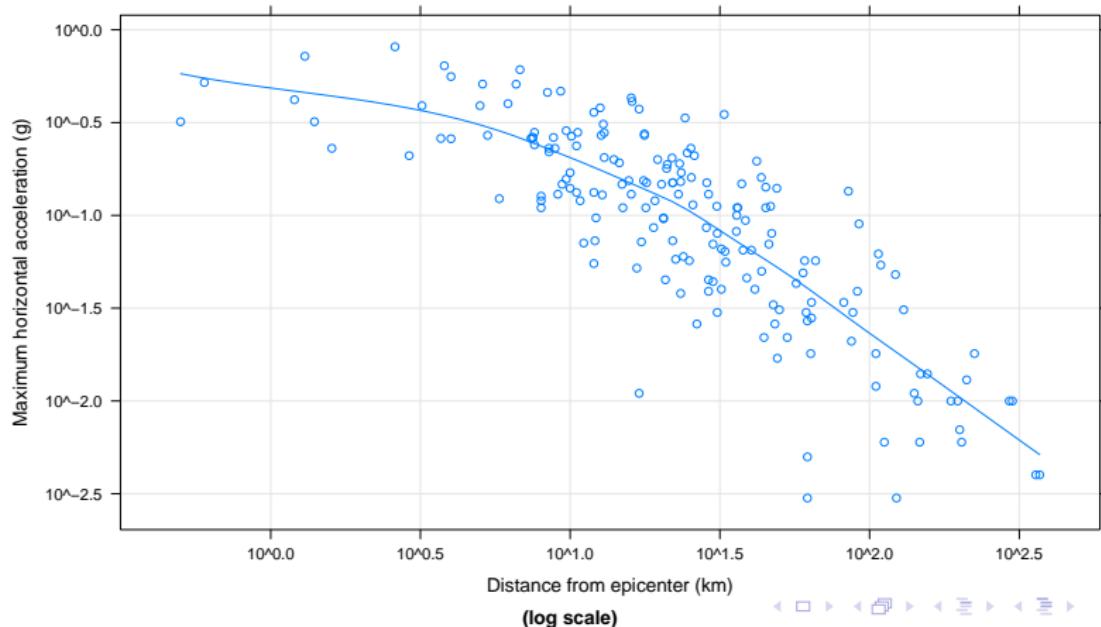
```
> xyplot(accel ~ distance, Earthquake,  
+        scales = list(log = TRUE), type = c("p",  
+        "g", "smooth"))
```



Add some labels

```
> xyplot(accel ~ distance, Earthquake,  
+         scales = list(log = TRUE), type = c("p",  
+             "g", "smooth"), sub = "(log scale)",  
+             xlab = "Distance from epicenter (km)",  
+             ylab = "Maximum horizontal acceleration (g)",  
+             main = "Larger quakes are felt closer to the epicenter")
```

Larger quakes are felt closer to the epicenter

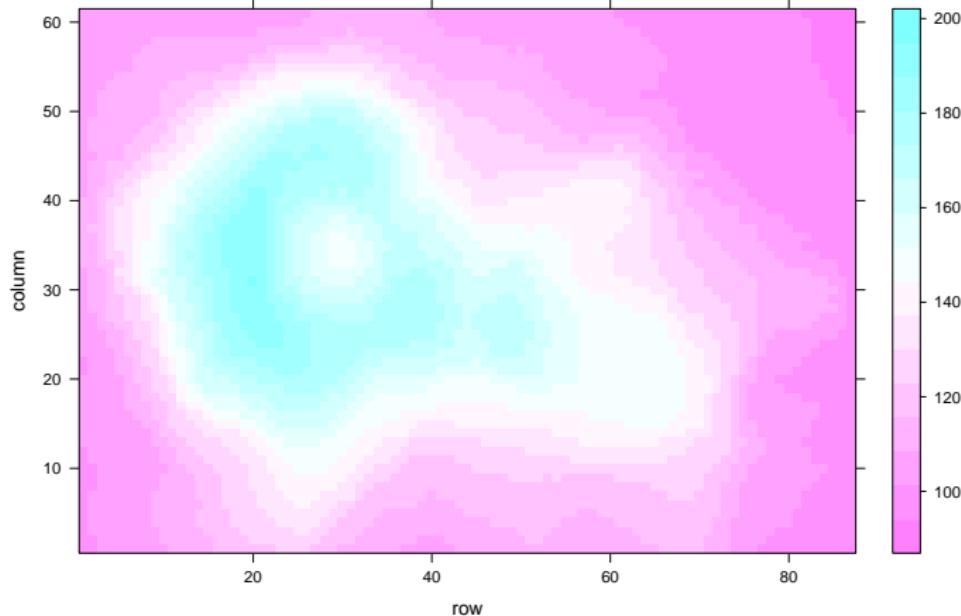


Volcano elevation data in matrix form

```
> dim(volcano)
[1] 87 61
> print(volcano[1:5, 1:5])
 [,1] [,2] [,3] [,4] [,5]
 [1,] 100 100 101 101 101
 [2,] 101 101 102 102 102
 [3,] 102 102 103 103 103
 [4,] 103 103 104 104 104
 [5,] 104 104 105 105 105
```

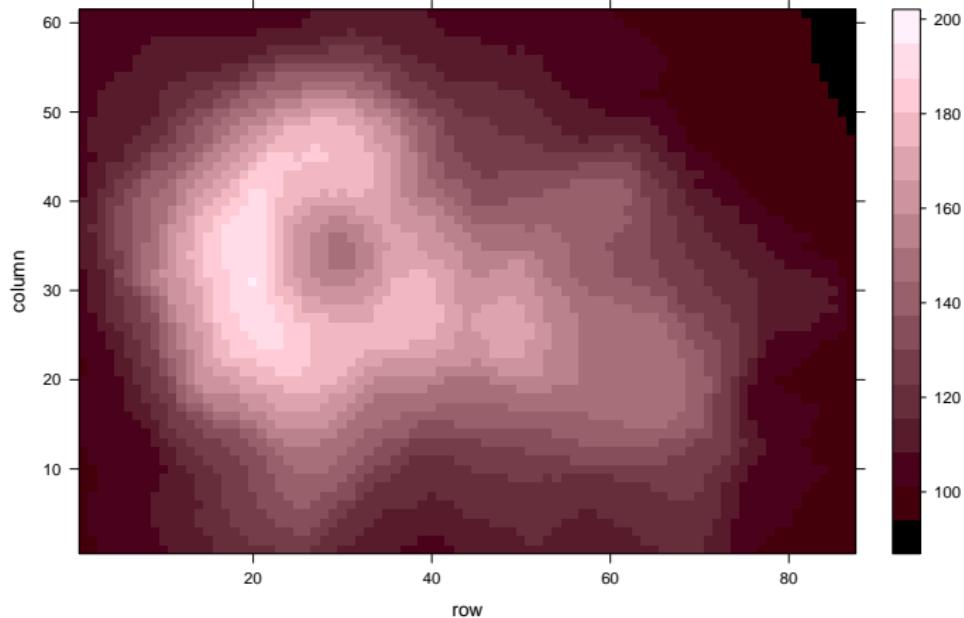
Plot volcano elevations in a matrix using color

```
> levelplot(volcano)
```



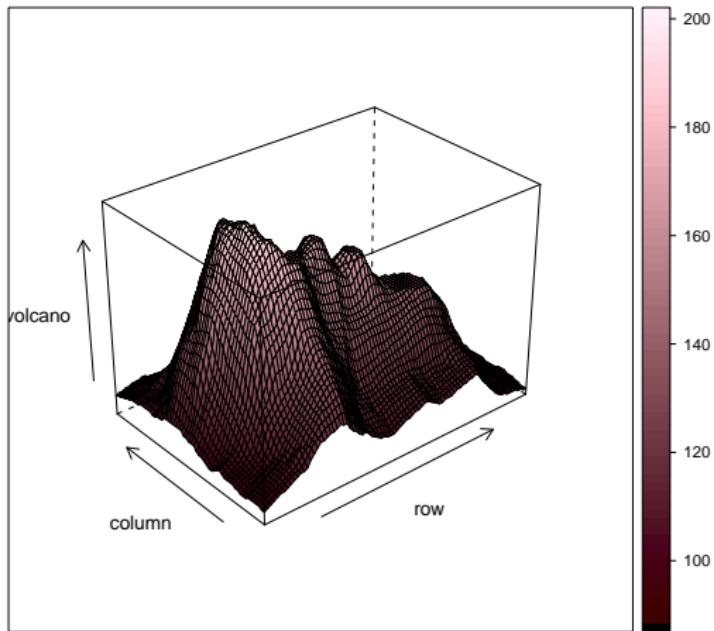
Use a different color scale

```
> my.colors <- sapply(0:100, function(l) hcl(l = 1))  
> levelplot(volcano, col.regions = my.colors)
```



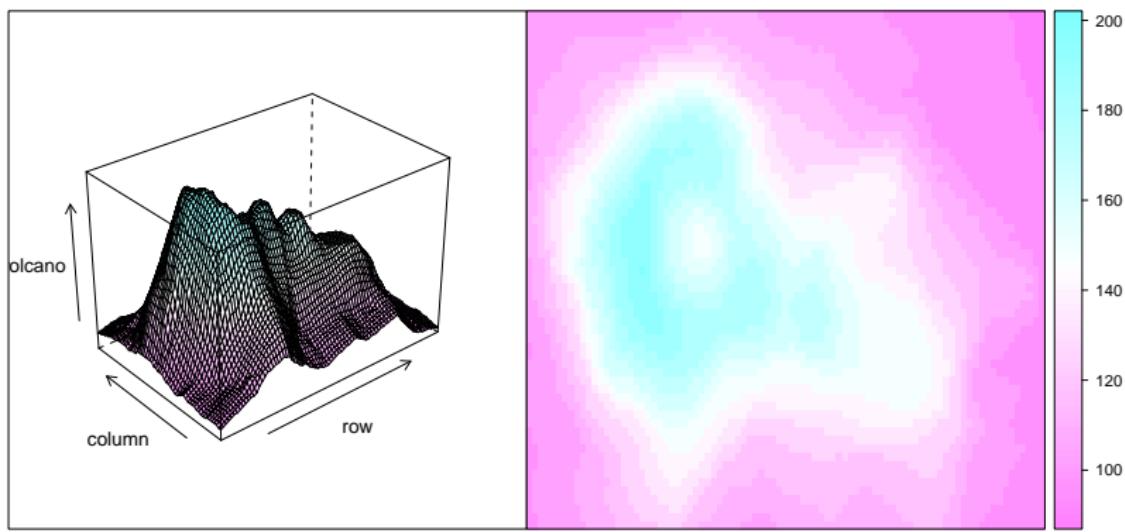
Use 3d wireframe plots

```
> wireframe(volcano, drape = TRUE, col.regions = my.colors)
```



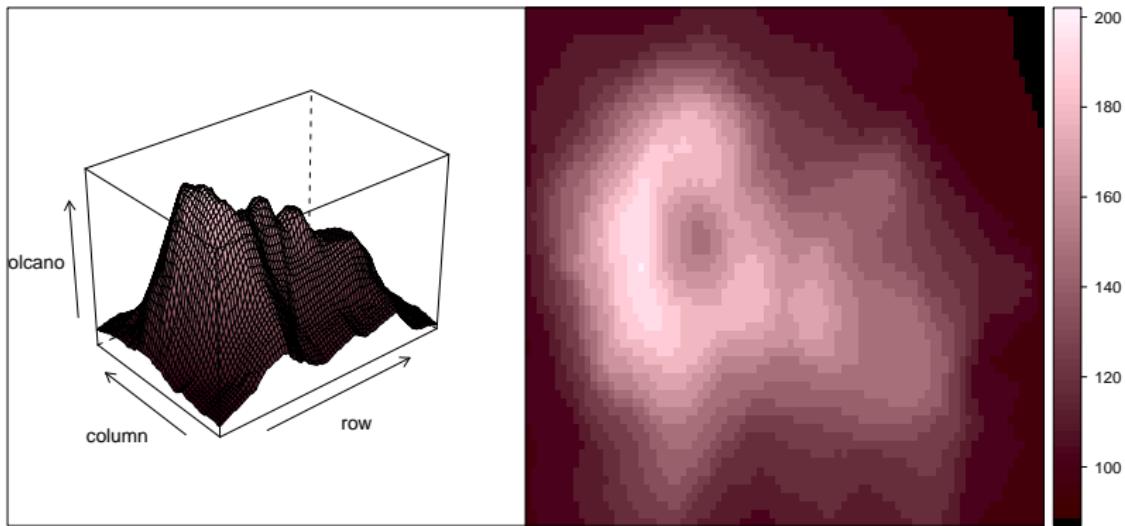
Combine plots using latticeExtra

```
> library(latticeExtra)
> both <- c(wireframe(volcano, drape = TRUE),
+           levelplot(volcano))
> both
```



Globally change the plot parameters

```
> trellis.par.set(regions = list(col = my.colors))  
> both
```



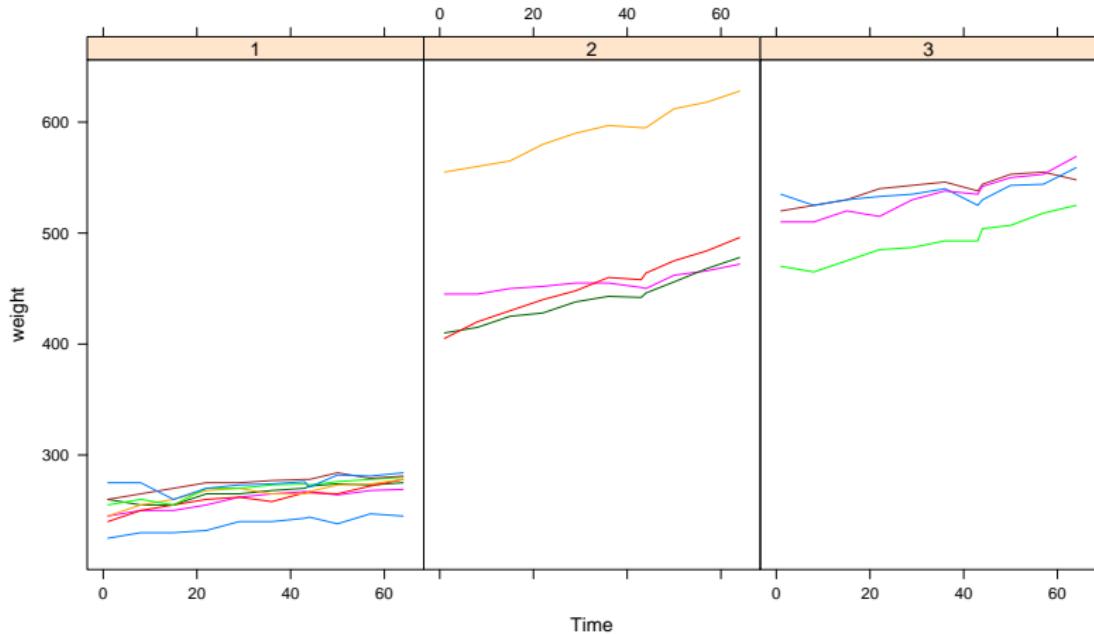
Longitudinal data

```
> data(BodyWeight, package = "nlme")
> head(BodyWeight)
```

	weight	Time	Rat	Diet
1	240	1	1	1
2	250	8	1	1
3	255	15	1	1
4	260	22	1	1
5	262	29	1	1
6	258	36	1	1

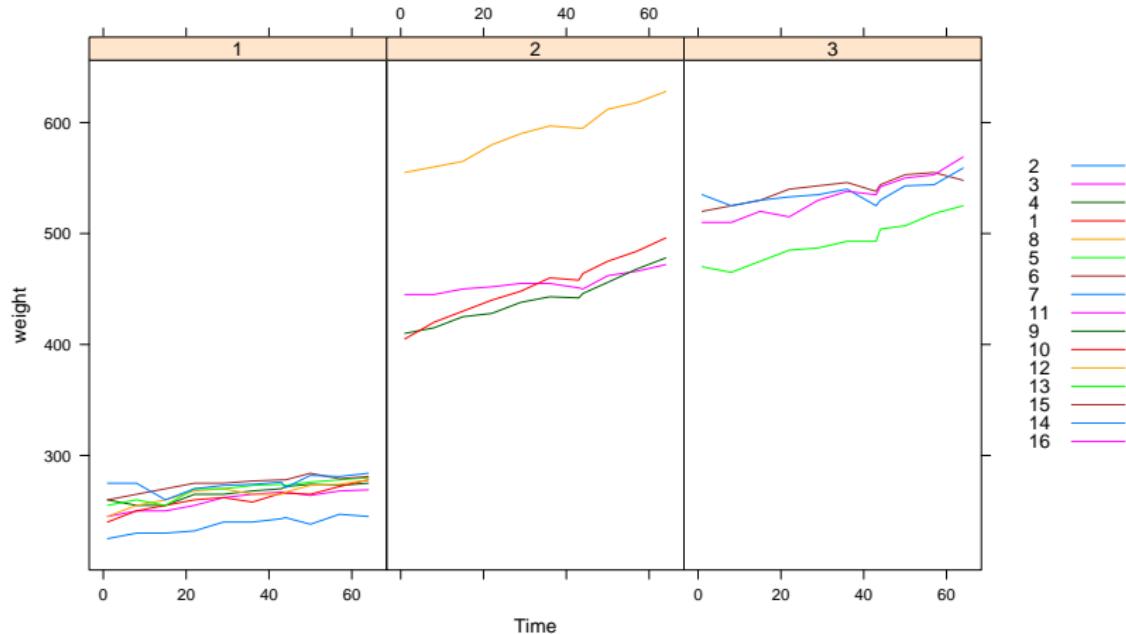
Conditional scatterplots reveal difference between treatments

```
> xyplot(weight ~ Time | Diet, BodyWeight,  
+         groups = Rat, type = "l", layout = c(3,  
+         1))
```



Legends with more than a few items are very confusing

```
> xyplot(weight ~ Time | Diet, BodyWeight,  
+         groups = Rat, type = "l", layout = c(3,  
+             1), auto.key = list(space = "right",  
+             points = FALSE, lines = TRUE))
```



Outline

The lattice system

Adding direct labels using the lattice<code>dl</code> package

Why use direct labels instead of legends?

- ▶ Edward Tufte, professor emeritus of statistics at Yale.
- ▶ *The Visual Display of Quantitative Information* (1983).



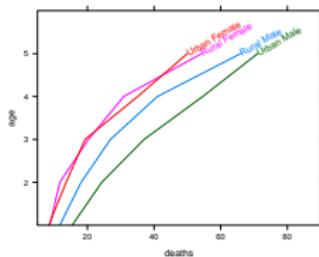
- ▶ One of his points: legends make it harder to decode a statistical graphic.
- ▶ Use direct labels whenever possible.

How to plot direct labels in R?

- ▶ Lattice + latticedl: `direct.label(xyplot(y~x,data,groups=z),method=f)`
- ▶ Positions of direct labels can be specified as a function of the data:

```
f <- function(d,...){  
  # d is a data frame with columns x,y,groups of the data points  
  #... analyze the points and return the label positions:  
  return(data.frame(x=a,y=b,groups=c))  
}
```

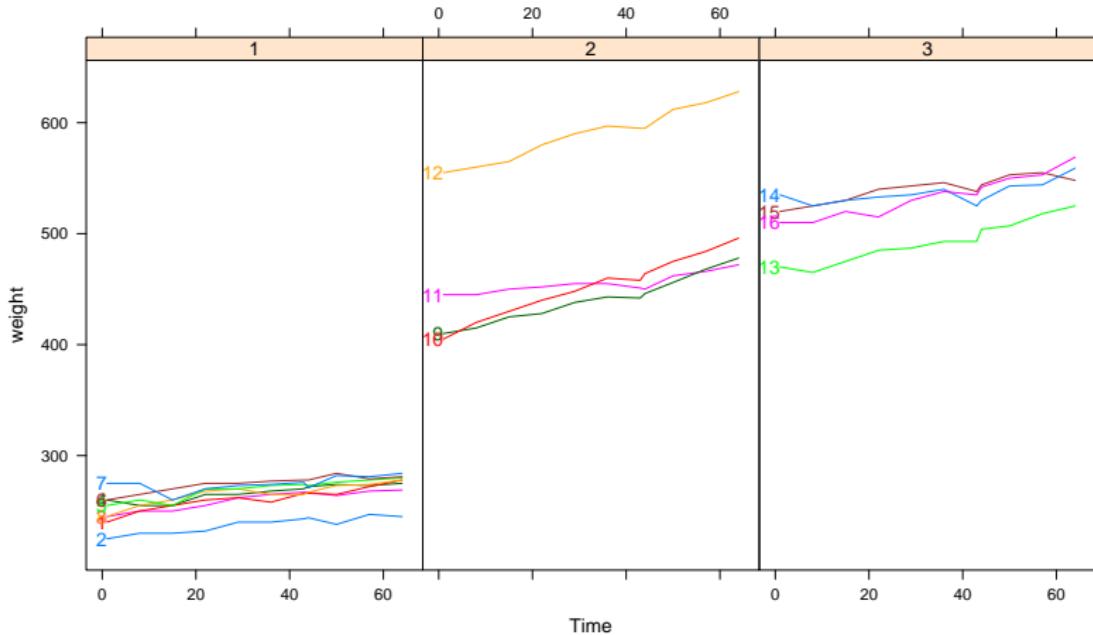
	groups	x	y	hjust	vjust	rot
1	Rural Male	66.0	5	0	0.5	30
2	Rural Female	54.3	5	0	0.5	30
3	Urban Male	71.1	5	0	0.5	30
4	Urban Female	50.0	5	0	0.5	30



- ▶ `latticedl` does the labeling for you, keeping track of the correct colors.
- ▶ Common plot types have default direct labeling methods.

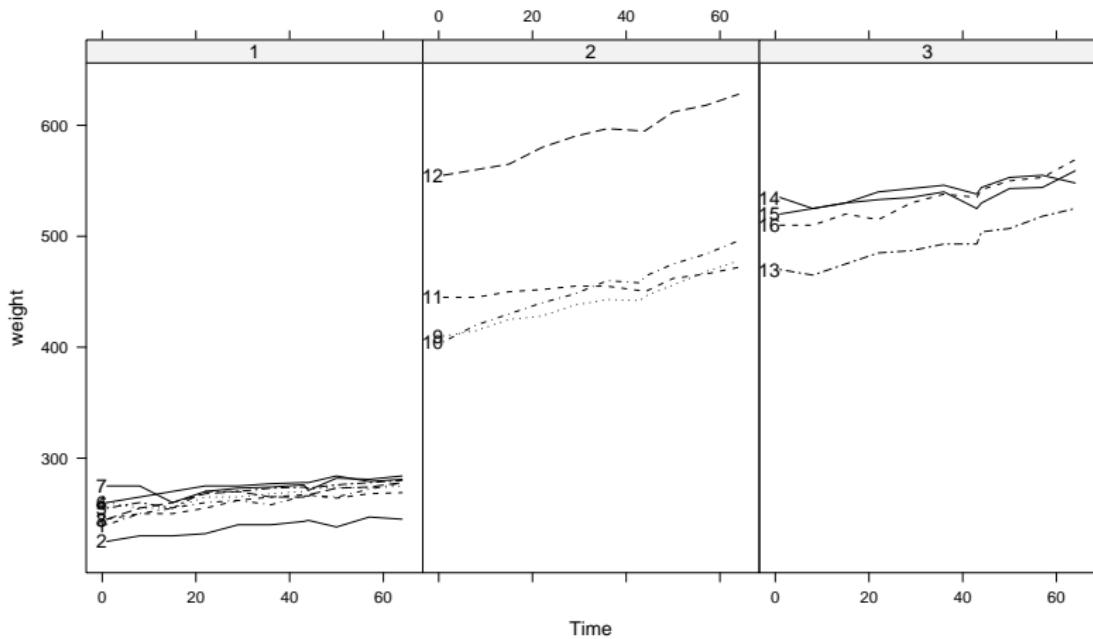
Easy fix for confusing legend: direct labels

```
> library(lattice)
> long <- xyplot(weight ~ Time | Diet, BodyWeight,
+   groups = Rat, type = "l", layout = c(3,
+   1))
> direct.label(long)
```



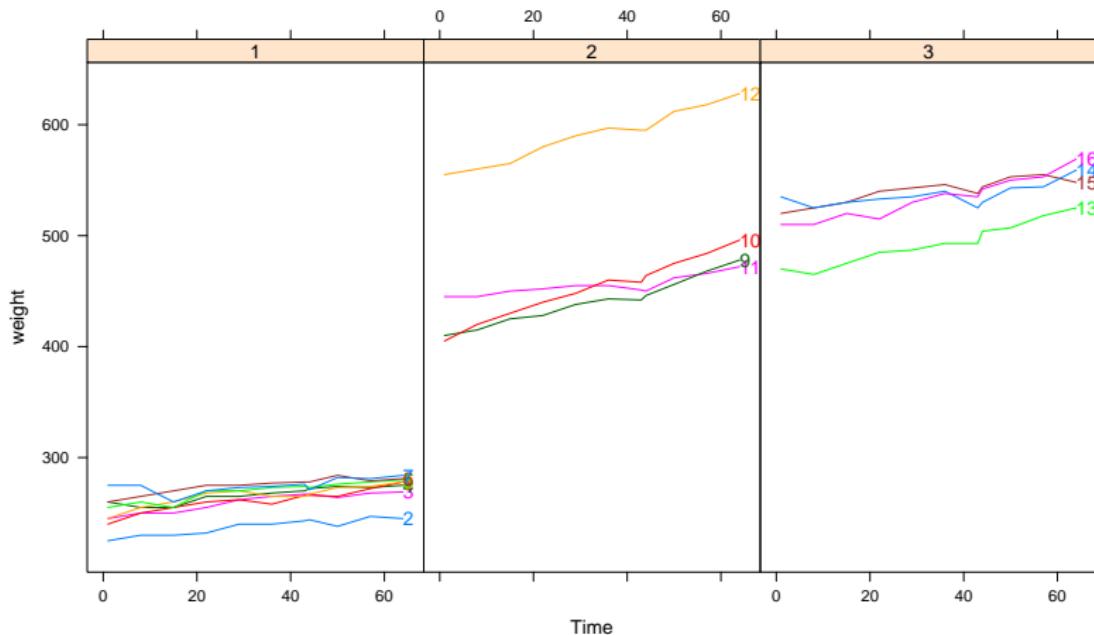
Even works in black and white

```
> longbw <- update(long, par.settings = standard.theme(color = FALSE))  
> direct.label(longbw)
```



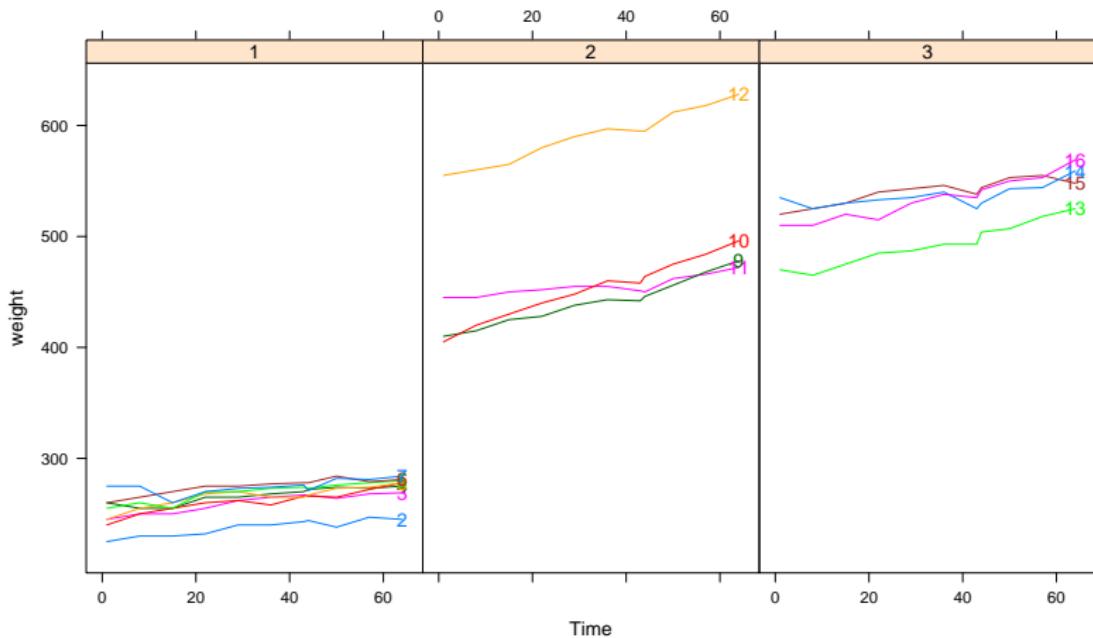
Change label positions with the method argument

```
> direct.label(long, method = last.points)
```



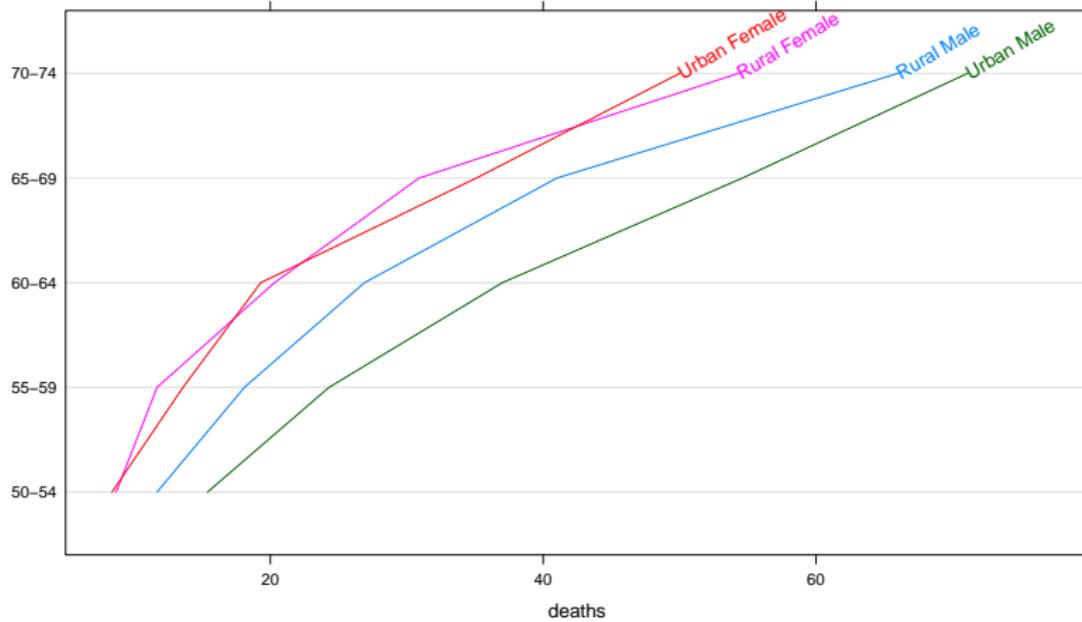
Make your own positioning function using `dl.indep`

```
> direct.label(long, method = dl.indep(d[which.max(d$x),  
+ ]))
```



You can change text parameters (same as `grid::grid.text`)

```
> direct.label(dots2, method = list("last.points",
+      rot = 30))
```



Load some data on car fuel efficiency

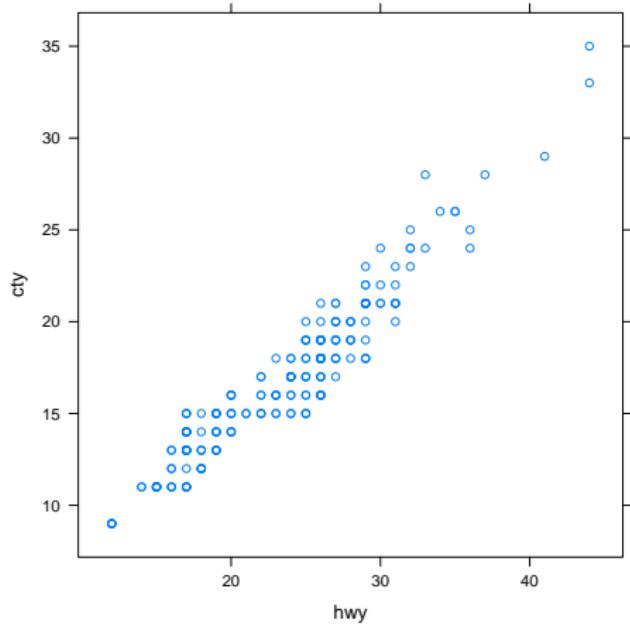
```
> data(mpg, package = "ggplot2")
> head(mpg)

  manufacturer model displ year cyl      trans drv cty
1       audi     a4    1.8 1999    4   auto(l5)   f 18
2       audi     a4    1.8 1999    4 manual(m5)   f 21
3       audi     a4    2.0 2008    4 manual(m6)   f 20
4       audi     a4    2.0 2008    4   auto(av)   f 21
5       audi     a4    2.8 1999    6   auto(l5)   f 16
6       audi     a4    2.8 1999    6 manual(m5)   f 18

  hwy fl   class
1 29 p compact
2 29 p compact
3 31 p compact
4 30 p compact
5 26 p compact
6 26 p compact
```

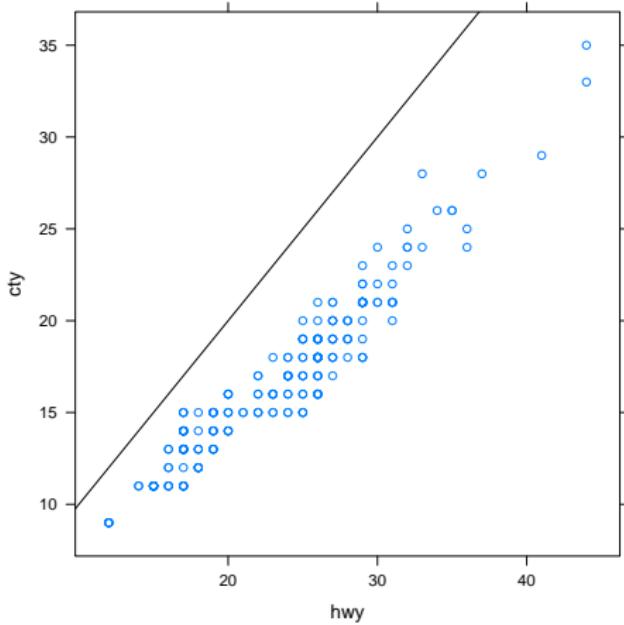
Plot city versus highway fuel efficiency

```
> xyplot(cty ~ hwy, mpg, aspect = 1)
```



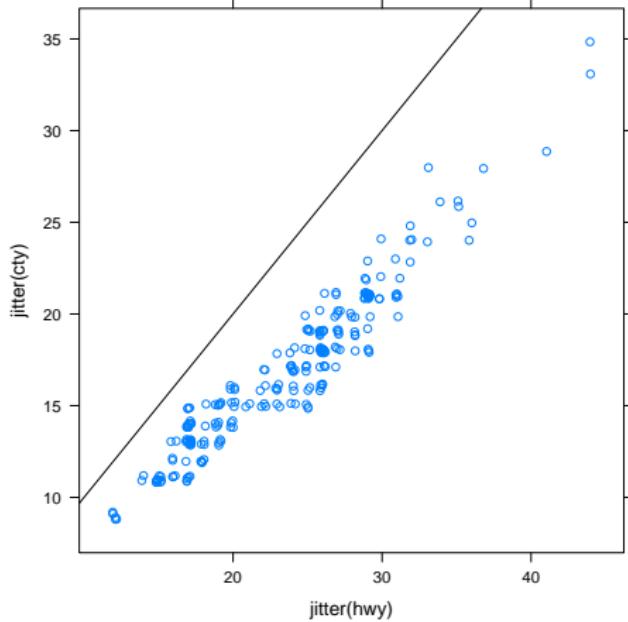
Add a reference line $x=y$

```
> panel.xyref <- function(...) {  
+   panel.xyplot(...)  
+   panel.abline(0, 1)  
+ }  
> xyplot(cty ~ hwy, mpg, aspect = 1, panel = panel.xyref)
```



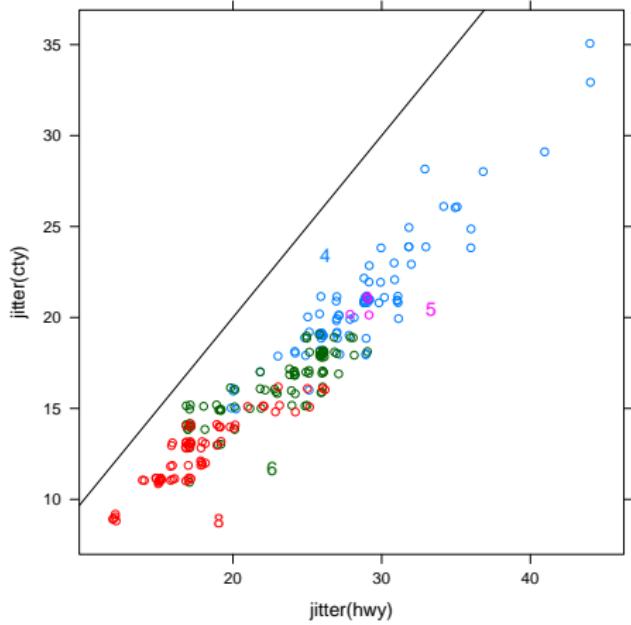
Jitter the data to see all the points

```
> xyplot(jitter(cty) ~ jitter(hwy), mpg,  
+         aspect = 1, panel = panel.xyref)
```



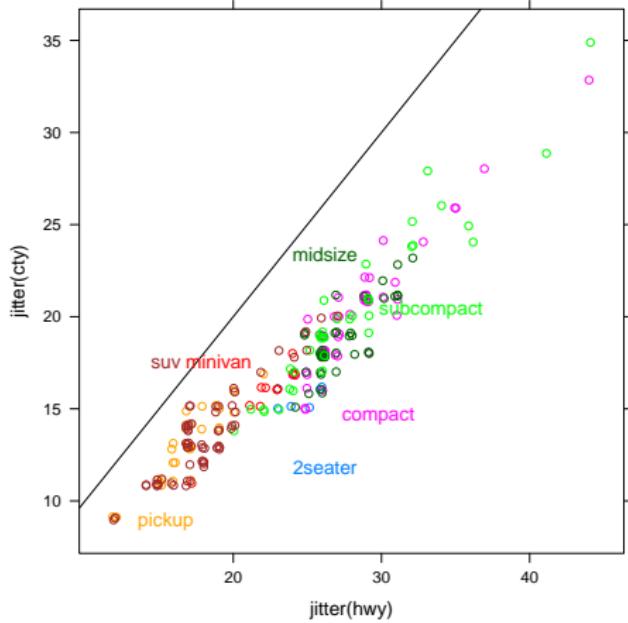
Group data by number of cylinders in the engine

```
> direct.label(xyplot(jitter(cty) ~ jitter(hwy),  
+      mpg, aspect = 1, panel = panel.xyref,  
+      groups = factor(cyl)))
```



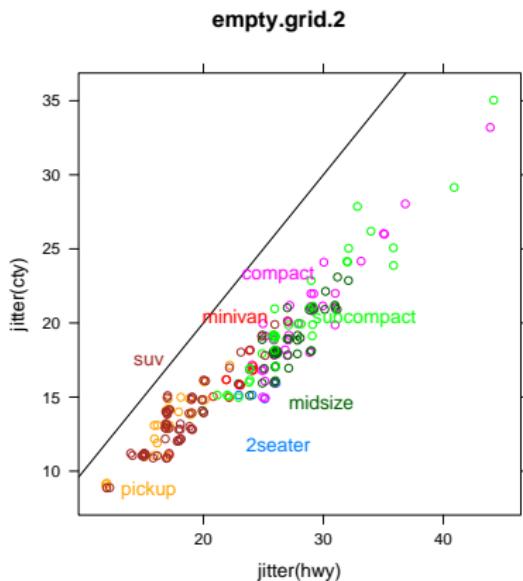
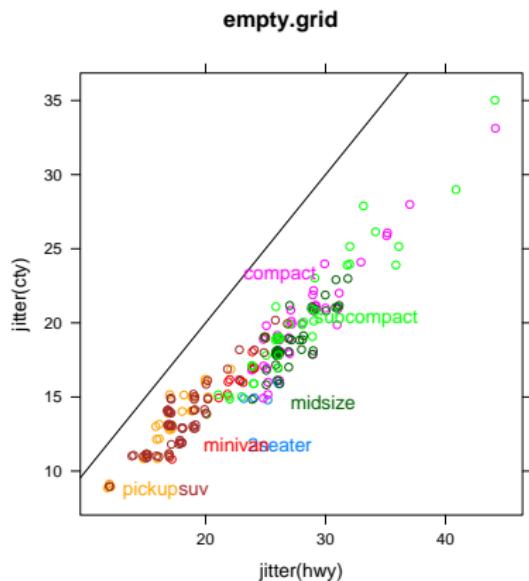
Group data by car class

```
> direct.label(xyplot(jitter(cty) ~ jitter(hwy),  
+      mpg, aspect = 1, panel = panel.xyref,  
+      groups = class))
```



Compare direct labeling methods

```
> compare.methods(c("empty.grid", "empty.grid.2"),
+     xyplot, mpg, jitter(cty) ~ jitter(hwy),
+     class, aspect = 1, panel = panel.xyref,
+     horiz = TRUE)
```



References for learning more about lattice and latticedl

- ▶ First load the libraries in R
 - ▶ `library(lattice)`
 - ▶ `library(latticeExtra)`
 - ▶ `library(latticedl)`
- ▶ Then you can look at the interactive help pages

[Overview](#) `?Lattice`

[Customizing plots](#) `?xyplot`

[Included panel functions](#) `?panel.functions`, `?llines`

[Multiple plots per page](#) `?plot.trellis`, `?c.trellis`

[Direct labeling](#) `?direct.label`

- ▶ Deepayan Sarkar (2008) Lattice: Multivariate Data Visualization with R, Springer.
- ▶ R code from the slides available on the web:
<http://directlabels.r-forge.r-project.org>
- ▶ Email me directly: toby.hocking@inria.fr