Visualizing multivariate data using lattice and direct labels
http://directlabels.r-forge.r-project.org

Toby Dylan Hocking
toby.hocking AT inria.fr

15 October 2009
Outline

The lattice system

Adding direct labels using the latticedl package
Brief history of lattice

- 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.
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

```r
> 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)"
```

![Barley Yield Diagram](attachment:image.png)
Aspect ratio in scatterplots is important.

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

![Yearly Sunspots graph](chart.png)
Lattice also automatically calculates aspect ratio for optimal decoding

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

![Yearly Sunspots Graph](image-url)
Load a data set

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

   lea school student score gender age gcsescore gcsecnt
1    1      1       1     4   F   3   6.625  0.3393
2    1      1       2    10   F  -3   7.625  1.3393
3    1      1       3    10   F  -4   7.250  0.9643
4    1      1       4    10   F  -2   7.500  1.2143
5    1      1       5     8   F  -1   6.444  0.1583
6    1      1       6    10   F   4   7.750  1.4643
```

Simple histogram

\textbf{Sample Code:}

\begin{verbatim}
> histogram(~gcsescore, Chem97)
\end{verbatim}
Histograms conditional on a categorical variable

> histogram(~gcsescore | factor(score),
+Chem97)
Box and whisker plots

```r
> bwplot(gcsescore ~ gender | factor(score),
+       Chem97, layout = c(6, 1))
```
Conditioned plots of kernel density estimates

```r
> densityplot(~gcsescore | factor(score),
+      Chem97)
```
Hide the actual points with the `plot.points` argument

```r
> densityplot(~gcsescore | factor(score),
+ Chem97, plot.points = FALSE)
```
Conditioned and grouped density plots

```r
> densityplot(~gcsescore | factor(score),
+   Chem97, plot.points = FALSE, groups = gender)
```
Add a legend with the auto.key argument

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

```
gcsescore
Density
0.0
0.2
0.4
0.6
0.8
0 2 4 6 8
0 2
0 2 4 6 8
4
6
0 2 4 6 8
8
0.0
0.2
0.4
0.6
0.8
10
M
F
```
Legend layout with the columns argument

```r
> densityplot(~gcsescore | factor(score),
+       Chem97, plot.points = FALSE, groups = gender,
+       auto.key = list(columns = 2))
```
Legend positioning with the space argument

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

> show.settings()
Show settings good for printout

```r
> 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

```r
> 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`

```r
> 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

```r
> print(VADeaths)

<table>
<thead>
<tr>
<th></th>
<th>Rural Male</th>
<th>Rural Female</th>
<th>Urban Male</th>
<th>Urban Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>50-54</td>
<td>11.7</td>
<td>8.7</td>
<td>15.4</td>
<td>8.4</td>
</tr>
<tr>
<td>55-59</td>
<td>18.1</td>
<td>11.7</td>
<td>24.3</td>
<td>13.6</td>
</tr>
<tr>
<td>60-64</td>
<td>26.9</td>
<td>20.3</td>
<td>37.0</td>
<td>19.3</td>
</tr>
<tr>
<td>65-69</td>
<td>41.0</td>
<td>30.9</td>
<td>54.6</td>
<td>35.1</td>
</tr>
<tr>
<td>70-74</td>
<td>66.0</td>
<td>54.3</td>
<td>71.1</td>
<td>50.0</td>
</tr>
</tbody>
</table>
```
> vad <- as.data.frame.table(VADeaths)
> names(vad) <- c("age", "demographic", "deaths")
> head(vad)

<table>
<thead>
<tr>
<th>age</th>
<th>demographic</th>
<th>deaths</th>
</tr>
</thead>
<tbody>
<tr>
<td>50-54</td>
<td>Rural Male</td>
<td>11.7</td>
</tr>
<tr>
<td>55-59</td>
<td>Rural Male</td>
<td>18.1</td>
</tr>
<tr>
<td>60-64</td>
<td>Rural Male</td>
<td>26.9</td>
</tr>
<tr>
<td>65-69</td>
<td>Rural Male</td>
<td>41.0</td>
</tr>
<tr>
<td>70-74</td>
<td>Rural Male</td>
<td>66.0</td>
</tr>
<tr>
<td>50-54</td>
<td>Rural Female</td>
<td>8.7</td>
</tr>
</tbody>
</table>
Grouped dotplots work well for these data

```r
> dotplot(age ~ deaths, vad, groups = demographic,
+ type = "o")
```
Plots can be saved as R objects

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

```r
> dots
```
Saved plots can be updated later

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

![Graph showing updated plot with age groups and deaths](image-url)
Add a confusing legend ... how can we label more intuitively?

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

![Chart showing the number of deaths for different age and gender groups. The chart includes lines for Rural Male, Rural Female, Urban Male, and Urban Female. The x-axis represents deaths, and the y-axis represents age groups from 50-54 to 70-74.]
Load some earthquake measurements

```r
> 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

```r
> xyplot(accel ~ distance, Earthquake,
+       scales = list(log = TRUE))
```
Type "p" is the default

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

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

```r
> 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

(log scale)
Distance from epicenter (km)
Maximum horizontal acceleration (g)
Volcano elevation data in matrix form

> dim(volcano)
[1] 87 61
> print(volcano[1:5, 1:5])

[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

```r
> my.colors <- sapply(0:100, function(l) hcl(l = l))
> levelplot(volcano, col.regions = my.colors)
```
Use 3d wireframe plots

```r
> wireframe(volcano, drape = TRUE, col.regions = my.colors)
```
Combine plots using latticeExtra

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

```
row

<table>
<thead>
<tr>
<th>volcano</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
</tr>
<tr>
<td>120</td>
</tr>
<tr>
<td>140</td>
</tr>
<tr>
<td>160</td>
</tr>
<tr>
<td>180</td>
</tr>
<tr>
<td>200</td>
</tr>
</tbody>
</table>

column

row
```
Globally change the plot parameters

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

> `both`
Longitudinal data

```r
> 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

```r
> xyplot(weight ~ Time | Diet, BodyWeight,
+       groups = Rat, type = "l", layout = c(3,
+       1))
```
Legends with more than a few items are very confusing

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

Adding direct labels using the latticedl package
Why use direct labels instead of legends?

▶ Edward Tufte, professor emeritus of statistics at Yale.

▶ 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:

  ```r
  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))
  }
  ```

<table>
<thead>
<tr>
<th>groups</th>
<th>x</th>
<th>y</th>
<th>hjust</th>
<th>vjust</th>
<th>rot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rural Male</td>
<td>66.0</td>
<td>5</td>
<td>0</td>
<td>0.5</td>
<td>30</td>
</tr>
<tr>
<td>Rural Female</td>
<td>54.3</td>
<td>5</td>
<td>0</td>
<td>0.5</td>
<td>30</td>
</tr>
<tr>
<td>Urban Male</td>
<td>71.1</td>
<td>5</td>
<td>0</td>
<td>0.5</td>
<td>30</td>
</tr>
<tr>
<td>Urban Female</td>
<td>50.0</td>
<td>5</td>
<td>0</td>
<td>0.5</td>
<td>30</td>
</tr>
</tbody>
</table>

- `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

```r
> 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

```r
> 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`

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

![Graph showing weight over time with labeled points](image)
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

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

<table>
<thead>
<tr>
<th>manufacturer</th>
<th>model</th>
<th>displ</th>
<th>year</th>
<th>cyl</th>
<th>trans</th>
<th>drv</th>
<th>cty</th>
</tr>
</thead>
<tbody>
<tr>
<td>audi</td>
<td>a4</td>
<td>1.8</td>
<td>1999</td>
<td>4</td>
<td>auto(l5)</td>
<td>f</td>
<td>18</td>
</tr>
<tr>
<td>audi</td>
<td>a4</td>
<td>1.8</td>
<td>1999</td>
<td>4</td>
<td>manual(m5)</td>
<td>f</td>
<td>21</td>
</tr>
<tr>
<td>audi</td>
<td>a4</td>
<td>2.0</td>
<td>2008</td>
<td>4</td>
<td>manual(m6)</td>
<td>f</td>
<td>20</td>
</tr>
<tr>
<td>audi</td>
<td>a4</td>
<td>2.0</td>
<td>2008</td>
<td>4</td>
<td>auto(av)</td>
<td>f</td>
<td>21</td>
</tr>
<tr>
<td>audi</td>
<td>a4</td>
<td>2.8</td>
<td>1999</td>
<td>6</td>
<td>auto(l5)</td>
<td>f</td>
<td>16</td>
</tr>
<tr>
<td>audi</td>
<td>a4</td>
<td>2.8</td>
<td>1999</td>
<td>6</td>
<td>manual(m5)</td>
<td>f</td>
<td>18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>hwy</th>
<th>fl</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>p</td>
<td>compact</td>
</tr>
<tr>
<td>29</td>
<td>p</td>
<td>compact</td>
</tr>
<tr>
<td>31</td>
<td>p</td>
<td>compact</td>
</tr>
<tr>
<td>30</td>
<td>p</td>
<td>compact</td>
</tr>
<tr>
<td>26</td>
<td>p</td>
<td>compact</td>
</tr>
<tr>
<td>26</td>
<td>p</td>
<td>compact</td>
</tr>
</tbody>
</table>
```
Plot city versus highway fuel efficiency

> xyplot(cty ~ hwy, mpg, aspect = 1)
Add a reference line $x=y$

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

```r
hwy
cty
10
15
20
25
30
35
20 30 40
●
●
●
●
●
● ...
● ●
●
●
●
●
●
●
●
●
●●●
●
●
●
●
●
●
● ●
●
●
●
●
●
●
●
●
```
Jitter the data to see all the points

```r
> xyplot(jitter(cty) ~ jitter(hwy), mpg, 
+     aspect = 1, panel = panel.xyref)
```
Group data by number of cylinders in the engine

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

Group data by car class

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

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


R code from the slides available on the web:
   http://directlabels.r-forge.r-project.org

Email me directly: toby.hocking AT inria.fr