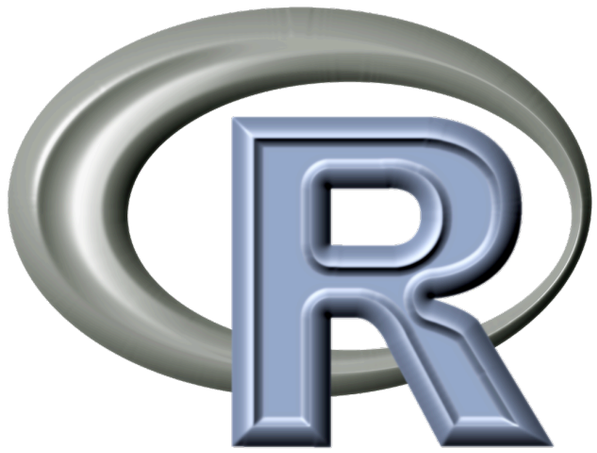

Lattice Graphics

Michael Malick

Updated: 15 May 2013

Graphics Packages

There are three main graphics packages for R:

- `graphics` (base graphics)
  - Don't need to load
- `lattice` (grid graphics)
  - Need to load `library(lattice)`
- `ggplot2` (grid graphics)
  - Need to install and load

Graphics Demos

```
library(lattice)
demo(lattice)
dev.off()
```

Lattice Package

Lattice graphics provide a consistent environment that is useful for visualizing multivariate datasets

- The power of lattice graphics is in displaying multiple plots on a single display
- Lattice graphics are also known as trellis plots because they resemble a garden trellis
- The lattice package was written by Deepayan Sarkar and is documented in his book
  - Lattice: Multivariate data visualization with R

Lattice Package

- The lattice package is not an extension of base graphics but is instead built on the `grid` package created by Paul Murrell (author of "R Graphics")
- You **cannot** "mix-and-match" lattice and base graphics commands
  - E.g., the `par()` function doesn't work with lattice

Lattice Functions

Base Graphics	Description	Lattice
`barplot`	Bargraph	`barchart`
`biplot`	Represents rows and columns of a matrix	
`boxplot`	Box and whisker plot	`bwplot`
`contour`	Contour plot	`contourplot`
`coplot`	Conditional plot	`xyplot`
`dotchart`	Cleveland dot chart	`dotplot`
`filled.contour`	Contour plot filled with colors	`contourplot`
`hist`	Histogram	`histogram`
`image`	High-density image plot	`levelplot`
`interaction.plot`	Interactions plot for two-factors	
`matplot`	Multiple plots specified by columns of a matrix	
`pairs`	All pairwise plots between variables	`splom`
`persp`	3D perspective plot	`wireframe, cloud`
`pie`	Pie chart	
`plot`	Plot	`xyplot`
`qqnorm`	Normal QQ plot	`qqmath`
`qqplot`	Quantile-Quantile plot	`qq`
`scatter.smooth`	Scatterplot with a smooth curve	
`stars`	Star plots for multivariate data	

Lattice Formula

Lattice graphics require that data be specified as a formula

```
y ~ x | a * b
```

- Primary variables are located to the left of the |
- Conditioning variables are located to the right of the |
- At least one primary variable is required but conditioning variables are optional

Conditioning

Lattice graphics use "conditioning" to produce multipanel plots

- Lattice plots are created by conditioning on the levels of one or more variables in a dataset
- Conditioning on categorical variables creates the same plot for each subset or level of the categorical variable
- Conditioning on numeric variables creates the same plot corresponding to intervals of the numeric variable
- Conditioning is **not** a requirement to produce lattice graphics

Example: Conditioning

```
plot(iris$Sepal.Width, iris$Sepal.Length)


xyplot(Sepal.Length ~ Sepal.Width,
    data = iris)
```

```
# Condition on species
xyplot(Sepal.Length ~ Sepal.Width | Species,
    data = iris)
```

You Try...

1. **Using the** `barley` **dataset, use** `xyplot()` **to plot** `yield` **vs.** `year` **conditioned on** `site`
2. **Using the** `barley` **dataset, use** `dotplot()` **to plot** `site` **vs.** `yield` **conditioned on** `year`

Lattice Graphics

- Every lattice display consists of a series of rectangular panels, organized in a row by column array
- The indexing of the array is right-to-left, top-to-bottom
- The x and y variables of all the panes are identical
- Each panel in the display corresponds to a subset of the conditioning variable

Groups

Lattice graphics use "groups" to differentiate between levels within a panel

```
head(barley)
dotplot(variety ~ yield | site,
    data = barley)
```

```
dotplot(variety ~ yield | site,
    groups = year,  data = barley,
    auto.key = TRUE)
```

```
xyplot(Sepal.Width ~ Sepal.Length,
    groups = Species, data = iris,
    auto.key = TRUE)
```

You Try...

1. **Using the** `barley` **dataset, use** `dotplot()` **to plot** `site` **vs.** `yield` **grouped by** `year`

Dataset Organization

To use conditioning and groups you need to organize your data in a "stacked" manner according to your conditioning or grouping variables

Year	Site1	Site2	Site3
1960	240,000	142,236	332,867
1961	60,000	45,972	47,049
1962	133,800	208,086	194,910
1963	38,081	373,412	127,154

Year	Site	Count
1960	Site1	240,000
1961	Site1	60,000
1962	Site1	133,800
1963	Site1	38,081
1960	Site2	142,236
1961	Site2	45,972
1962	Site2	208,086
1963	Site2	373,412
1960	Site3	332,867
1961	Site3	47,049
1962	Site3	194,910
1963	Site3	127,154

Labels and Titles

Labels and titles in lattice use the same commands as base graphics

```
dotplot(variety ~ yield | site,
     groups = year,  data = barley,
     xlab = "Yield",
     ylab = "Variety",
     main = "Barley Yields")
```

Scales

The lattice argument `scales` allows you to change how the x and y axes are displayed

```
dotplot(variety ~ yield | site, groups = year,
     data = barley)
```

```
dotplot(variety ~ yield | site, groups = year,
     data = barley,
     scales = list(relation = "free"))

dotplot(variety ~ yield | site, groups = year,
     data = barley,
     scales = list(x = list(relation = "free"),
     y = list(relation = "same")))
```

Legends

Lattice has three arguments for creating legends (usually needed when a grouping variable is used)

- `auto.key` = adds a legend that can be minimally customized
- `simpleKey` = used by `auto.key` to produce a simple legend (generally don't use this)
- `key` = adds a legend that can be completely customized

Legends

```
dotplot(variety ~ yield | site,
     groups = year, data = barley,
     auto.key = TRUE)

dotplot(variety ~ yield | site,
     groups = year, data = barley,
     auto.key = list(space = "top", columns = 2))

dotplot(variety ~ yield | site, groups = year,
     data = barley, pch = 19, col = 3:4,
     key = list(space = "right", pch = 19,
     text = list(c("1932", "1931"), cex = 1.1),
     points = list(TRUE, c(1, 3), col = 3:4, lwd
     = 2), border = TRUE))
```

You Try...

1. Using the `barley` dataset, use `dotplot()` to plot `site` vs. `yield` grouped by `year`
2. Add a legend using `auto.key`

Panel Functions

Panel functions allow you to incrementally add elements to each panel in a lattice graphic

- Panel functions are functions that are nested within a lattice plot call
- They are responsible for graphical content inside panels
- They get executed once for every panel
- Every high level function has a default panel function, e.g., `xyplot()` has default panel function `panel.xyplot()`

Example: Panel Function #1

```
xyplot(Sepal.Length ~ Sepal.Width | Species,
       data = iris)
```

```
# Add a horizontal line to each panel
xyplot(Sepal.Length ~ Sepal.Width | Species,
       data = iris,
       panel = function(x, y, ...) {
          panel.abline(h = 6, col = 2)
          panel.xyplot(x, y)
       })
```

Example: Panel Function #2

```
xyplot(Sepal.Length ~ Sepal.Width | Species,
       data = iris)
```

```
# Add a LOESS model to each panel
xyplot(Sepal.Length ~ Sepal.Width | Species,
       data = iris,
       panel = function(x, y, ...) {
          panel.loess(x, y, col = 2, lwd = 3)
          panel.xyplot(x, y)
       })
```