Trellis Plots

Trellis plots (S-Plus) and Lattice plots in R also create layouts for multiple plots. A *trellis* of plots is generated as a sequence of plots that are then arranged in rows, columns and pages. The sequence is determined by the conditioning factors in the *formula*

\[
\sim X \\
Y \sim X \\
Y \sim X \mid Z \\
Y \sim X \mid Z \star W
\]

where \( Z \) and \( W \) are conditioning factors, \( Y \) is on the y-axis, and \( X \) is on the x-axis.
Lattice Functions

library(lattice)  # already loaded in .First()

help(Lattice)

example(xyplot)

use example(functionname) to see demos
Examples

# side by side boxplots
   bwplot(factor ~ numeric, ...)
# probability plots:
   qqmath(factor ~ numeric, ...)
# histograms:
   histogram( ~ numeric, ...)
# smoothed histogram:
   densityplot( ~ numeric, ...)
# xy plots
   xyplot( y ~ x | factor, ...)
# scatter plot matrices
   splom( ~ dataframe, ...)

The quake example uses a numeric variable as a conditioning variable
Mercury in Bass

Mercury concentrations in bass fillets from the Wacamaw and Lumber rivers in NC.

```r
fish = read.table("../datasets/fish", header=T)
fish$RIVER = factor(fish$RIVER)
fish$STATION = factor(fish$STATION)
summary(fish)
levels(fish$RIVER) = c("Lumber", "Wacamaw")
summary(fish)
```

The commands above change RIVER and STATION to a factor (from a numeric variable) and assign more meaningful levels. The summary commands are there to make sure that the factor names still have the correct count.
**Conditioning in Boxplots** \( \texttt{bwplot()} \)

\[
\texttt{bwplot(STATION} \sim \texttt{MERCURY} \mid \texttt{RIVER, data=fish)}
\]
Conditioning in Scatterplots `xyplot()`

`xyplot(WEIGHT ~ LENGTH | STATION, data=fish)`
Conditioning Scatter Plots `coplot()`

The function `coplot(y ~ x | a*b, ...)` is an alternative to the lattice `xyplot` functions and is slightly easier to use with continuous conditioning variables.

```r
> coplot(water81 ~ income | retired, data=concord)
```

```r
> coplot(water81 ~ income | retired*educat, data=concord, panel=panel.smooth, xlab=c("1981 Income (in $1000)", "Given: Retired Status"), ylab=c("1981 Water Usage (cubic feet)", "Given: Education in Years"))
```
Coplot Example

Given: Retired Status

1981 Income (in $1000)

1981 Water Usage (cubic feet)

Exploratory Data Analysis - Part 2 – p. 9/20
Other Graphics Commands

- adding a smooth line to a plot
Adding a Smooth Trend

- The function `lowess` fits a one dimensional smooth trend to (x,y) pairs

- To construct a scatter plot with smooth trend
  1. `plot(duration, interval)`
  2. `lines(lowess(duration, interval))`
     or
  3. `scatter.smooth(duration, interval)`
Default Lowess Smooth

Old Faithful Eruptions

Waiting Time (mins) vs. Duration (mins)
lowess

The algorithm behind lowess and its cousin loess is quite complex, but the idea is to use robust local estimates of the curve

- A window is placed around a point $x$
- data points in the window are weighted so that points near $x$ get the most weight
- using the points in the window and weights, use a robust weighted regression to predict at $x$
- the parameter $f$ gives the proportion of points in the plot which influence the smooth at each value. Larger values give more smoothness. Default is $2/3$. 

Exploratory Data Analysis - Part 2 – p. 13/20
Choice of f in lowess

Old Faithful Eruptions

![Graph showing Old Faithful Eruptions with Waiting Time (mins) on the y-axis and Duration (mins) on the x-axis. Different values of f are indicated by different line styles and colors.](image-url)
legend(x, y = NULL, legend, fill=NULL, 
col = "black", lty, lwd, pch, 
angle = NULL, density = NULL, 
bty = "o", bg = par("bg"), 
pt.bg = NA, cex = 1, xjust = 0, 
yjust = 1, 
x.intersp = 1, y.intersp = 1, 
adj = c(0, 0.5), 
text.width = NULL, 
text.col = par("col"), 
merge = do.lines && has.pch, 
trace = FALSE, 
plot = TRUE, ncol=1, horiz=FALSE)
Making a Legend

```r
legend(min(duration), max(interval),
       paste("f =",
             c("1", "f=2/3", "f=.5", "f=.1")),
       col=2:5
       lty=c(2,1,3,4),
       lwd=rep(2,4))
```
Identifying Points

Draw plot, then use either

- `identify(x, y, labels)` to label points
- `locator(n)` to find coordinates of n locations
- left-click with mouse at locations
- to stop, right-click the mouse in the plot

Use `text()` to add other text or legends at locations.
Transformations

For plots to be more effective, we may need to transform variables.

Common transformations

- $\log$ positive continuous measures, concentrations, size
- $\sqrt{}$ counts
- $\hat{}^{-1}$

Typically need the largest observation to be at least 10 times larger than the smallest case for transformations to be effective.
Log Transformations

plot(brain ~ body, data=Animals, log="xy")
identify(body, brain, rownames(Animals))
(see plot.default)
Multiple Plots per Page

Use the `mfrow` option to control the number of plots in a page.

The command

\[
\text{par(mfrow=c(3,2))}
\]

creates a figure with 3 rows of plots and 2 columns.

See `help(par)` for more graphical control parameters.