Trellis Graphics

Trellis graphics are an advanced library of graphics functions that offer additional features in graphing. Trellis graphs allow you to analyze multiple variables by conditioning of 1, 2, or 3 variables on additional (conditioning) variables. In this section, we give a very brief introduction to Trellis graphics.

The trellis.device function sets up a graphics device with appropriate settings. This calls graphsheet on Windows, motif on Unix, or postscript if specified.

```
> trellis.device()
```

If you call a Trellis plot without opening this device, Trellis commands will open one by default in recent versions of S-PLUS or give an error in older versions. However, by invoking the trellis.device command, you ensure that a compatible device is opened.
Trellis plots use a variation on model formula syntax to specify the plots. The basic syntax is: y ~ x | g1*g2*.... Notice that the y variable is specified first, then the x variable. The variables g1, g2, ... are the optional conditioning variables.

We will create a histogram of Mileage conditioned on Type from the fuel.frame data set.

```
> attach(fuel.frame)
> histogram( ~Mileage | Type)
```

Each panel in this plot corresponds to a level in the factor variable Type.
To condition on a numeric variable, one option is to use the cut command to create a factor.

```
> quantile(Disp.)
  0%   25%   50%  75% 100%
  73 113.75 144.5  180  305
> cut(Disp.,quantile(Disp.),include.lowest=T)
 [1] 1 2 1 1 1 1 1 1 1 1 1 1 1 4 3 4 2 1 2 3 1 1 2 3 2 4 2 2 2 2 4 3 3 2
[35] 2 2 2 3 3 3 4 3 4 4 2 3 3 3 4 3 4 4 4 3 4 4 4 2 3 3
attr(, "levels"):
[1] " 73.00+ thru 113.75" "113.75+ thru 144.50" "144.50+ thru 180.00"
[4] "180.00+ thru 305.00"
> my.cut <-cut(Disp.,quantile(Disp.),include.lowest=T)
> xyplot(Mileage~Weight|my.cut)
> detach("fuel.frame")
```

Shingles are the other option for conditioning on continuous variables. Shingles are S-PLUS objects consisting of a series of intervals, with overlap of intervals allowed.

The shingle function can be used to convert a continuous variable into a shingle. You must specify the sets of ranges (intervals) to use and these ranges may be either overlapping or non-overlapping.

The equal.count function also creates shingles, but chooses non-uniform ranges to make each sub-interval contain approximately the same number of points. You specify the number of shingles to create and the amount that you want them to overlap.

You can plot a shingle object to see the ranges and overlap.

Examples

```
> trellis.device()  # set up graphics device
> attach(fuel.frame)  # make data convenient
```

We can create shingles for conditioning in different ways:

```
> D1 <- equal.count(Disp., number=6, overlap=1/2)
> tmp <- seq( min(Disp.), max(Disp.), length=4 )
> D2 <- shingle(Disp., intervals=cbind( tmp[-4], tmp[-1] ) ) # no overlap
> tmp <- seq( min(Disp.), max(Disp.), length=5 )
> D3 <- shingle(Disp., intervals=cbind( tmp[-(4:5)],tmp[-(1:2)]) ) # overlap
> print(plot(D1),split=c(1,1,1,3),more=T)
> print(plot(D2),split=c(1,2,1,3),more=T) # print plots trellis objects
> print(plot(D3),split=c(1,3,1,3),more=F)
```

Now we make some trellis plots:

```
> xyplot( Mileage ~ Weight | D1, panel=panel.smooth )

 > xyplot( Mileage ~ Weight | D2, panel=panel.smooth )

> xyplot( Mileage ~ Weight | D3, panel=panel.smooth )
> detach("fuel.frame")
```

Notice that all of the panels are on the same scale. This allows us to see how the conditioning variable is related to the other two variables. In the above plot, we see that cars with high displacement are also heavy and have low mileage. The overlapping option gives us more points per panel (giving more accuracy to the smooth), but also means that individual points can be represented in more than one panel.

Trellis plots can convey large amounts of information but can also be complex to program. First look at a basic plot of "Frost" vs. "Murder".

```
> plot(state.x77[,"Frost"], state.x77[,"Murder"])
```

There is a fairly strong negative trend. Does that mean that murderers are afraid of the cold or could there be other reasons? Perhaps a more likely scenario is that the murder rate is highest in cities

with high population density, and these cities are mostly located on the coasts (which are warmer than the inland states). Or perhaps the issue is one of low income states versus high income state.

Let's do a trellis plot conditioning on income. We will also indicate the region into which each state falls. The key argument creates a key (legend) to tell us which symbol/color corresponds to each region:

```
> attach( as.data.frame( state.x77 ) )
> xyplot( Murder ~ Frost | equal.count( Income ),
+ groups = state.region, pch=1:4, panel=panel.superpose,
+ key=list( points=list(pch=1:4), col=2:5, text=levels(state.region),
+ columns=4 ) )
> detach(2)
```

This tells a different story, with lots of information and detail that suggest further directions for study.

See the help on trellis.args for the arguments that are common to all trellis plots. In addition, and see the help on trellis.examples

```
> ?trellis.examples
```

for a list of built-in examples of the various functions. Typing a function name with parentheses will create the plot; without parentheses, the code for the plot is displayed.