

## R SCRIPT FOR CREATING LATTICE GRAPHS

The remainder of this handout provides the R script used in the APSA Short Course. The text was created in the WinEdt text processor, and pasted into the R console, using the tools provided in the "RwinEdt" package within R.

---

```
#####  
#####  
###      CREATING POWERFUL AND EFFECTIVE GRAPHICAL DISPLAYS:  
###      AN INTRODUCTION TO LATTICE GRAPHICS IN R  
###  
###      A Short Course at the 2007 Annual Meetings of the  
###      American Political Science Association,  
###      August 29, 2007, Chicago, IL  
###  
###      William G. Jacoby  
###      Department of Political Science  
###      Michigan State University  
#####  
#####  
  
library(lattice)  
  
###      Read data from file "states, with names.txt".  
###      Note header record.  
  
states <- read.table(file.choose(), header = T)  
  
states  
  
###  
###      Reading data from a file that does not contain a header record  
###  
  
states2 <- read.table(file.choose())  
  
states2  
  
###  
###      Use colnames function to assign names to variables  
###  
  
colnames(states2) <- c("state", "polprior", "party",  
  "ideol", "igstr", "govsize", "region", "ptygov")  
  
states2[1:10, ]  
  
###  
###      Using the "foreign" package to  
###      read data files that have been saved  
###      in a different format-- STATA, in this case  
###  
  
library(foreign)  
  
states3 <- read.dta(file.choose())  
  
states3[1:10, ]
```

## R Script for Creating Lattice Graphics

Page 2

```
###
### Which objects current exist in the Workspace?
###

objects()

###
### Clean up, and remove unnecessary objects
###

remove("states2", "states3", "x", "y")

###
### Giving R instructions to find particular variables--
### The first statement will cause an error!
###

polprior
states$polprior
summary(states$polprior)

###
### Attaching data frames can be convenient, but it can
### also cause serious confusion and trouble
###

attach(states)

polprior
polprior <- c(1, 2, 3, 4)

polprior
states$polprior

detach(states)

objects()

remove(polprior)

###
### Show some of the general display
### functions in the lattice package
###

stripplot(~states$polprior)
histogram(~states$polprior)
densityplot(~states$polprior)
xyplot(states$polprior ~ states$ideol)

###
### Show example of a multipanel display--
### histograms of state policy priorities,
### separately for states with Republican
### and Democratic governors

states$ptygov[states$ptygov=="i"] <- "d"

histogram(~states$polprior | states$ptygov)
```

R Script for Creating Lattice Graphics  
Page 3

```
###
### Use the data= argument
###

  histogram(~polprior, data = states)

  xyplot(polprior ~ ideol, data = states)

###
### Set aspect ratio explicitly, and
### add axis labels
###

  histogram(~polprior, data = states,
            aspect = .75,
            xlab = "State policy priorities, 1992")

  xyplot(polprior ~ ideol, data = states,
         aspect = 1,
         xlab = "State electorate ideology",
         ylab = "state policy priorities, 1992")

###
### Create a dot plot and modify data
### to change the plot
###

  dotplot(state ~ polprior, data = states,
         aspect = 1.5,
         xlab = "State policy priorities, 1992")

###
### Decrease size of scale labels
###

  dotplot(state ~ polprior, data = states,
         aspect = 1.5,
         xlab = "State policy priorities, 1992",
         scales = list(cex = .65)
  )

###
### reorder factor "state" by priority scores
###

  states$state <- reorder(states$state, states$polprior)

###
### Create dot plot with reordered factor
###

  dotplot(state ~ polprior, data = states,
         aspect = 1.5,
         xlab = "State policy priorities, 1992",
         scales = list(cex = .65)
  )

###
### The "scales=" parameter can be used to
### change the details of the axes. For example,
### replace the quantitative scale in a histogram
### with textual labels
###

  histogram(~polprior, data = states,
            aspect = .75,
            xlab = "State policy priorities, 1992"
  )
```

## R Script for Creating Lattice Graphics

Page 4

```
    histogram(~polprior, data = states,
      aspect = .75,
      xlab = "State policy priorities, 1992",
      scales = list(x = list(tick.number = 3,
        at = c(10, 50, 90),
        labels = c("Particularized Benefits", "Mixture", "Collective Goods")))
    )

###
###   Create two-line textual labels
###

    histogram(~polprior, data = states,
      aspect = .75,
      xlab = "State policy priorities, 1992",
      scales = list(x = list(tick.number = 3,
        at = c(10, 50, 90),
        labels = c("Particularized\nBenefits",
          "Mixed\nPriorities",
          "Collective\nGoods")))
    )

###
###   Show that characteristics of plotting region
###   elements are controlled by panel function
###

    xyplot(polprior ~ igstr, data = states,
      aspect = 1,
      xlab = "Strength of interest groups in state",
      ylab = "State policy priorities"
    )

    xyplot(polprior ~ igstr, data = states,
      aspect = 1,
      xlab = "Strength of interest groups in state",
      ylab = "State policy priorities",
      col = "black",
      pch = 4
    )

###
###   Preceding changes were actually made within the panel function.
###   The following function creates an identical graph
###

    xyplot(polprior ~ igstr, data = states,
      aspect = 1,
      xlab = "Strength of interest groups in state",
      ylab = "State policy priorities",
      panel = function (x, y) {
        panel.xyplot(x, y, pch=4, col = "black")}
    )

###
###   The following "empty" panel function shows how
###   the panel function controls the plotting region
###

    xyplot(polprior ~ igstr, data = states,
      aspect = 1,
      xlab = "Strength of interest groups in state",
      ylab = "State policy priorities",
      panel = function (x, y) { }
    )
```

## R Script for Creating Lattice Graphics

Page 5

```
###
### Use panel function to change line style in dot plot
###

dotplot(state ~ polprior, data = states,
  aspect = 1.5,
  panel = function (x, y) {
    panel.dotplot(x, y, lty = 2)
  },
  xlab = "State policy priorities, 1992",
  scales = list(cex = .65)
)

###
### More complex use of panel functions to
### change both point and line color
### in dot plot
###

dotplot(state ~ polprior, data = states,
  aspect = 1.5,
  panel = function (x, y) {
    panel.abline(h = as.numeric(states$state), lty = 2, col = "black")
    panel.xyplot(x, as.numeric(y), pch = 16, col = "black")
  },
  xlab = "state policy priorities, 1992",
  scales = list(cex = .65)
)

###
### Panel function can be used to plot two variables simultaneously
###

xyplot(polprior ~ party, data = states,
  aspect = 1,
  xlab = "Orientations of state electorate",
  ylab = "State policy priorities",
  panel = function (x, y) {
    panel.xyplot(x, y, pch=1, col = "red")
    panel.xyplot(states$ideol, y, pch=4, col="green")}
)

###
### A graph which illustrates twenty possible plotting symbols
###

symbols <- c(1:20)

symbols

xyplot(symbols ~ symbols,
  aspect = 1,
  panel = function(x, y) {
    panel.xyplot(x, y, pch = symbols, col = "black")
  }
)

###
### Make symbols larger with "cex" parameter
###

xyplot(symbols ~ symbols,
  aspect = 1,
  panel = function(x, y) {
    panel.xyplot(x, y, pch = symbols, col = "black",
      cex = 1.25)
  }
)
```

## R Script for Creating Lattice Graphics

Page 6

```
###
### Add the symbol numbers to the
### graph of plotting symbols
###

xyplot(symbols ~ symbols,
       aspect = 1,
       panel = function(x, y) {
         panel.xyplot(x, y, pch = symbols, col = "black",
                      cex = 1.25)
         panel.text(x, y + .6, labels = symbols, cex = .75)
       }
)

###
### Panel functions can insert additional elements into the plotting region
###

###
### A reference line at Y = 50
###

xyplot(polprior ~ igstr, data = states,
       aspect = 1,
       xlab = "Strength of interest groups in state",
       ylab = "State policy priorities",
       panel = function (x, y) {
         panel.xyplot(x, y, pch=1, col = "black")
         panel.abline(h = 50, lty = 2, col = "blue")}
)

###
### A line showing the OLS fit
###

xyplot(polprior ~ igstr, data = states,
       aspect = 1,
       xlab = "Strength of interest groups in state",
       ylab = "State policy priorities",
       panel = function (x, y) {
         panel.xyplot(x, y, pch=1, col = "black")
         panel.lmline(x, y, lty = 1, col = "red")}
)

###
### A loess curve
###

xyplot(polprior ~ igstr, data = states,
       aspect = 1,
       xlab = "Strength of interest groups in state",
       ylab = "State policy priorities",
       panel = function (x, y) {
         panel.xyplot(x, y, pch=1, col = "black")
         panel.loess(x, y, span = .75, family="symmetric",
                    degree = 1, col = "green")}
)

###
### Place labels just to right of each plotted point
### DISCLAIMER: For illustration only. The results look bad!
###
```

## R Script for Creating Lattice Graphics

Page 7

```
xyplot(polprior ~ igstr, data = states,
  aspect = 1,
  xlab = "Strength of interest groups in state",
  ylab = "State policy priorities",
  panel = function (x, y) {
    panel.xyplot(x, y, pch=1, col = "black")
    panel.text(x+15, y, labels = states$state, cex = .75)}
)

###
### Use logical subscript operators to label points selectively.
### Results look much better!
###

xyplot(polprior ~ igstr, data = states,
  aspect = 1,
  xlab = "Strength of interest groups in state",
  ylab = "State policy priorities",
  panel = function (x, y) {
    panel.xyplot(x, y, pch=1, col = "black")
    panel.text(x[x>400 | y<20]-15, y[x>400 | y<20],
      labels = states$state[x>400 | y<20], cex = .75)}
)

###
### Use panel functions to modify plotting symbols selectively
###

xyplot(polprior ~ igstr, data = states,
  aspect = 1,
  xlab = "Strength of interest groups in state",
  ylab = "State policy priorities",
  panel = function (x, y) {
    panel.xyplot(x[states$party < mean(states$party)],
      y[states$party < mean(states$party)],
      pch=4, col = "red")
    panel.xyplot(x[states$party >= mean(states$party)],
      y[states$party >= mean(states$party)],
      pch=1, col = "blue")}
)

###
### Return to scatterplot with multiple symbols, create a key
###

xyplot(polprior ~ igstr, data = states,
  aspect = 1,
  xlab = "Strength of interest groups in state",
  ylab = "State policy priorities",
  panel = function (x, y) {
    panel.xyplot(x[states$party < mean(states$party)],
      y[states$party < mean(states$party)],
      pch=4, col = "red")
    panel.xyplot(x[states$party >= mean(states$party)],
      y[states$party >= mean(states$party)],
      pch=1, col = "blue")},
  key = list(text = list(text = c("Republican electorates", "Democratic
electorates"), cex = .75),
  points = list(pch = c(4, 1), col = c("red", "blue"), cex = .75),
  space = "top", border = T)
)

###
### Create multipanel display, showing
### scatterplot of two variables, while
### controlling for a third variable
###
```

## R Script for Creating Lattice Graphics

Page 8

```
### First, create the conditioning variable, using
### a special data form called a "shingle".
### Basically, this is a categorical variable with
### overlapping categories

given.party <- equal.count(states$party, number = 6, overlap = .75)

plot(given.party)

### Next, create scatterplot, conditioning on the
### shingle variable

xyplot(polprior ~ igstr | given.party, data = states,
       aspect = 1,
       xlab = "Strength of interest groups in state",
       ylab = "State policy priorities",
       panel = function(x, y) {
         panel.xyplot(x, y, col = "black")
         panel.lmline(x, y, col = "red")}
       )

### Calculate median policy priority scores
### within each region, and create dot plot
### of region medians

medvals <- tapply(states$polprior, states$region, FUN = median)

region <- as.ordered(names(medvals))

region

region <- reorder(region, medvals)

region

dotplot(region ~ medvals, aspect = 1.5)

### Perform multiple regression analysis
### of state policy priorities, and
### use results to construct a residual plot,
### with loess curve fitted to residuals

modell <- lm(polprior ~ ideol + party + igstr + govsize, data = states)

summary(modell)

xyplot(residuals(modell) ~ predict(modell),
       aspect = 1,
       panel = function(x, y) {
         panel.xyplot(x, y, col = "black")
         panel.abline(h = 0, lty = 2)
         panel.loess(x, y, span = .75,
                    degree = 1, family = "gaussian")}
       )

### Use the expression function to employ
### mathematical expressions in axis labels
###
```

## R Script for Creating Lattice Graphics

Page 9

```
xyplot(residuals(modell) ~ predict(modell),
       aspect = 1,
       panel = function(x, y) {
         panel.xyplot(x, y, col = "black")
         panel.abline(h = 0, lty = 2)
         panel.loess(x, y, span = .75,
                    degree = 1, family = "gaussian")
       },
       xlab = expression(hat(Y)),
       ylab = expression(e[i]),
       ylim = c(-40, 40)
)

###
###   A slightly fancier version of the axis labels
###

xyplot(residuals(modell) ~ predict(modell),
       aspect = 1,
       panel = function(x, y) {
         panel.xyplot(x, y, col = "black")
         panel.abline(h = 0, lty = 2)
         panel.loess(x, y, span = .75,
                    degree = 1, family = "gaussian")
       },
       xlab = expression("Predicted values,"~~ hat(Y) [i]),
       ylab = expression("Residuals,"~~e[i]),
       ylim = c(-40, 40)
)

###
###   Experimental evidence indicates that visual perception
###   of curves is optimized when the mean absolute angle of
###   the curve is 45 degrees. The process of "banking" is
###   used to change the aspect ratio to bring the mean absolute
###   angle as close to 45 degrees as possible
###

###
###   A line showing the OLS fit,
###   aspect ratio set to 1.0
###

xyplot(polprior ~ igstr, data = states,
       aspect = 1,
       xlab = "Strength of interest groups in state",
       ylab = "State policy priorities",
       panel = function(x, y) {
         panel.xyplot(x, y, pch=1, col = "black")
         panel.lmline(x, y, lty = 1, col = "red")}
)

###
###   Same scatterplot, with regression line
###   set to 45 degrees. Note use of "aspect="
###   argument and the "prepanel" function
###

xyplot(polprior ~ igstr, data = states,
       aspect = "xy",
       prepanel = function(x, y) prepanel.lmline(x, y),
       xlab = "Strength of interest groups in state",
       ylab = "State policy priorities",
       panel = function(x, y) {
         panel.xyplot(x, y, pch=1, col = "black")
         panel.lmline(x, y, lty = 1, col = "red")}
)
```

## R Script for Creating Lattice Graphics

Page 10

```
###
###  A loess curve, with aspect ratio of 1.0
###

xyplot(polprior ~ igstr, data = states,
       aspect = 1,
       xlab = "Strength of interest groups in state",
       ylab = "State policy priorities",
       panel = function (x, y) {
         panel.xyplot(x, y, pch=1, col = "black")
         panel.loess(x, y, span = .75, family="symmetric",
                    degree = 1, col = "green")}
       )

###
###  Same scatterplot, banked to 45 degrees (that is, the mean
###  absolute angle of the line segments that make up the loess
###  curve is 45 degrees)
###

xyplot(polprior ~ igstr, data = states,
       aspect = "xy",
       prepanel = function (x, y) prepanel.loess(x, y,span = .75, family="symmetric",
                                                degree = 1),
       xlab = "Strength of interest groups in state",
       ylab = "State policy priorities",
       panel = function (x, y) {
         panel.xyplot(x, y, pch=1, col = "black")
         panel.loess(x, y, span = .75, family="symmetric",
                    degree = 1, col = "green")}
       )

###
###  Print the last graph to a PDF file.
###

pdf(file = "e://lattice graphics demo//icpsr 2007//banked scatter 1.pdf")

xyplot(polprior ~ igstr, data = states,
       aspect = "xy",
       prepanel = function (x, y) prepanel.loess(x, y,span = .75, family="symmetric",
                                                degree = 1),
       xlab = "Strength of interest groups in state",
       ylab = "State policy priorities",
       panel = function (x, y) {
         panel.xyplot(x, y, pch=1, col = "black")
         panel.loess(x, y, span = .75, family="symmetric",
                    degree = 1, col = "green")}
       )

dev.off()

###
###  Use the scatterplot function from the car package
###

library(car)

scatterplot(polprior ~ igstr, data = states)
```