

# Regression III

## Lecture 3: Graphical Display of Data and Results

Dave Armstrong

University of Wisconsin – Milwaukee  
Department of Political Science

e: armstrod@uwm.edu  
w: www.quantoid.net/ICPSR.php

1 / 46

### Goals for Today

- Understand how **R** graphs are set up.
- Investigate the different graphical systems in **R**.
- Learn some of the basic elements of **R** graphing.
- Put some of these skills to use representing data and regression results.

2 / 46

### R Graphics

Default Plotting Methods

Adding Elements to Plots

Graphs Instead of Tables

Maps in R

Plotting Regression Results

Trellis (Lattice) Graphics

Grid Graphics

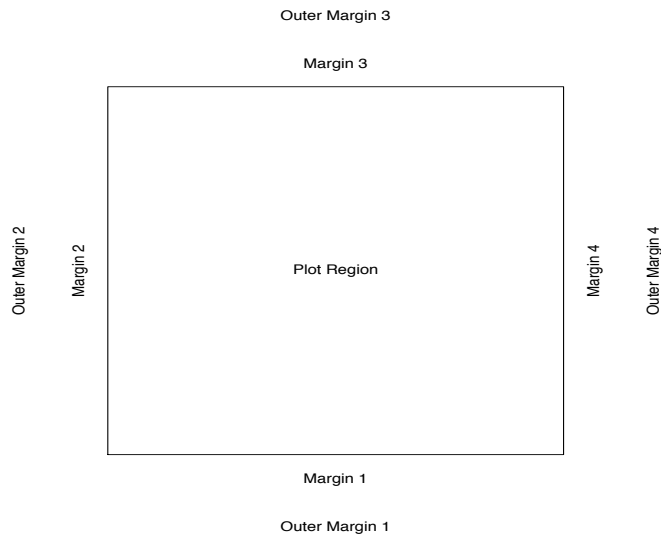
3 / 46

### The Setup

- **R** has three graphical “subsystems”
  - Traditional Graphics Systems operationalized in the `graphics` package which is loaded automatically in the default **R** distribution.
  - Trellis graphics in the package `lattice` by Deepayan Sarkar.
  - Grid graphics in the package `grid` by Paul Murrell.
- These systems work independently.
- Most of the time, any task can be accomplished in any of these environments, given the appropriate amount of ambition.

4 / 46

## Plotting Region



5 / 46

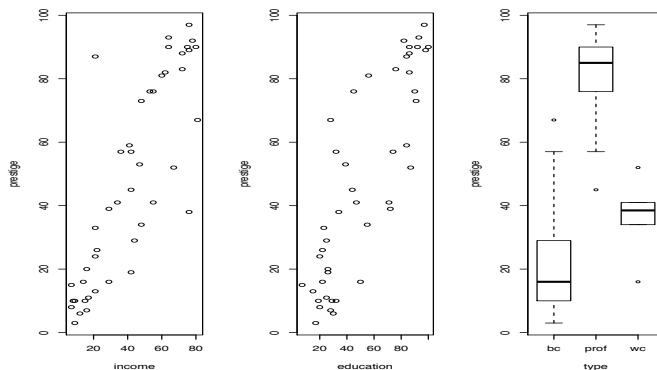
## How R Graphics Work

- Graphs in **R** are made sort of like a painting.
  - New elements write over previously included elements.
  - Graphs are not interactive - items cannot be taken out once they are included.
- To change something, you need to re-run the code that generated the graph after the required changes have been made.
- R does have some features that make it possible to interact with graphs, but these generally involve moving things around rather than putting things in and taking them out.
  - This is a rapidly evolving area of development for the **R**-project.

6 / 46

## Graphs of the Duncan Data

```
> library(car)
> data(Duncan)
> par(mfrow=c(1,3))
> plot
```



7 / 46

## R Graphics

### Default Plotting Methods

### Adding Elements to Plots

### Graphs Instead of Tables

### Maps in R

### Plotting Regression Results

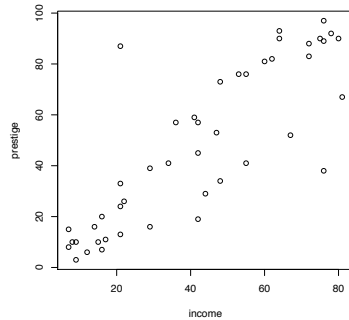
### Trellis (Lattice) Graphics

### Grid Graphics

8 / 46

### Default Plot Methods

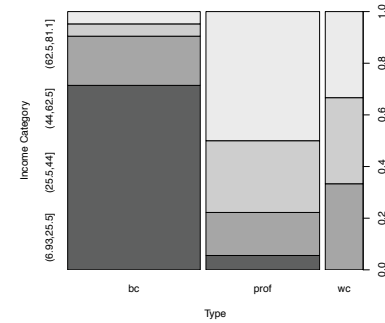
- Two Numeric Variables - scatterplot



- Two Factors - mosaic plot
- One Numeric, One Factor - boxplot
- One Numeric - scatterplot (with index as the x-axis)
- One factor - histogram

### Default Plot Methods

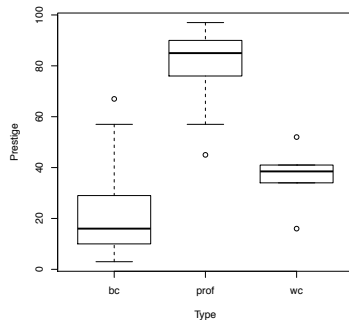
- Two Numeric Variables - scatterplot
- Two Factors - mosaic plot



- One Numeric, One Factor - boxplot
- One Numeric - scatterplot (with index as the x-axis)
- One factor - histogram

### Default Plot Methods

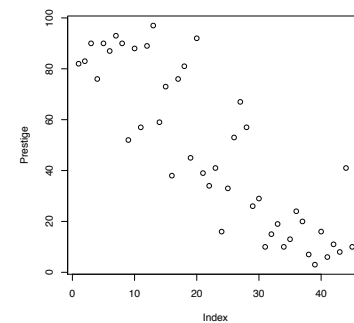
- Two Numeric Variables - scatterplot
- Two Factors - mosaic plot
- One Numeric, One Factor - boxplot



- One Numeric - scatterplot (with index as the x-axis)
- One factor - histogram

### Default Plot Methods

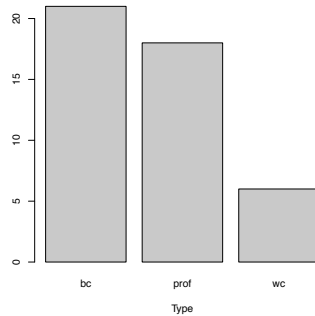
- Two Numeric Variables - scatterplot
- Two Factors - mosaic plot
- One Numeric, One Factor - boxplot
- One Numeric - scatterplot (with index as the x-axis)



- One factor - histogram

## Default Plot Methods

- Two Numeric Variables - scatterplot
- Two Factors - mosaic plot
- One Numeric, One Factor - boxplot
- One Numeric - scatterplot (with `index` as the  $x$ -axis)
- One factor - histogram



13 / 46

## R Graphics

### Default Plotting Methods

### Adding Elements to Plots

### Graphs Instead of Tables

### Maps in R

### Plotting Regression Results

### Trellis (Lattice) Graphics

### Grid Graphics

14 / 46

## Adding Elements to Plots: Points

Syntax: `points(x, y, pch, col, cex)`

- $x$  and  $y$  are (the coordinates at which to draw the point)
- `pch` is the character to be used:

○ △ + × ◇ ▽ ※ \* ◆ ⊕ ⊗ ⊞ ⊠ ⊡ ⊢ ⊣ ⊤ ⊥ ⊦ ⊧ ⊨ ⊩ ⊪ ⊫ ⊬ ⊭ ⊮ ⊯ ⊰ ⊱ ⊲ ⊳ ⊴ ⊵ ⊶ ⊷ ⊸ ⊹ ⊺ ⊻ ⊼ ⊽ ⊾ ⊿ ⊿  
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

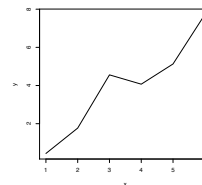
- `col` is the color of the plotting symbols (see help for `rainbow`, `colors`)
- `cex` is the character expansion factor (default=1) - makes points bigger or smaller.

15 / 46

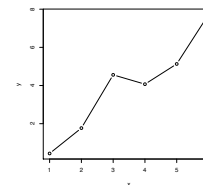
## Adding Elements to Plots: Lines and Segments (1)

Syntax: `segments(x0, y0, x1, y1, col, lty, lwd)`

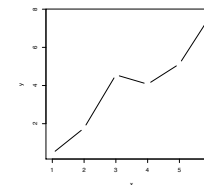
- $x0$  (starting  $x$ -value),  $y0$  (starting  $y$ -value),  $x1$  (ending  $x$ -value),  $y1$  (ending  $y$ -value),
- `type` - type of line-point plotting



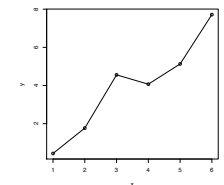
(a) l



(b) b



(c) c



(d) o

16 / 46

## Adding Elements to Plots: Arrows

Syntax: `arrows(x0, y0, x1, y1, length, angle, code, col, lty, lwd)`

- `x0` (starting  $x$ -value), `y0` (starting  $y$ -value), `x1` (ending  $x$ -value), `y1` (ending  $y$ -value),
- `length` - length of the arrow head (in inches)
- `angle` - angle of the arrow head edges from the line (in degrees)
- `code`
  - 1: arrow at  $(x_0, y_0)$
  - 2: arrow at  $(x_1, y_1)$
  - 3: arrow at  $(x_0, y_0)$  and  $(x_1, y_1)$

17 / 46

## Adding Elements to Plots: Text

Syntax: `text(x, y, label, adj, pos, offset)`

- `x` and `y` - coordinates at which to put text
- `label` character string to be put at coordinates (the text to be typed)
- `adj` one or two numbers in  $[0,1]$  representing the  $x$ - and  $y$ - axis adjustment of the labels
- `pos` specifies position of the text: 1) below, 2) left, 3) above, and 4) right
- `offset` offset of label from `pos` in fractions of character width.

18 / 46

## Adding Elements to Plots: Polygons

Syntax: `polygon(x, y, density, angle, border, col)`

- `x` and `y` are each vectors of  $n$  points that completely enclose a space to be shaded. Usually, it's helpful to make `x[n] = x[1]` and `y[n] = y[1]`.
- `density` is the density of shading lines
- `angle` is the angle of the shading lines
- `border` the color of the border (NA to omit the border)

19 / 46

## Saving Graphs from R

There are many ways you can save your graphs from **R**.

- In the graph device opened by **R**, you can go to file → save as → [choose file type].
- You can write your graph out to a device without seeing it on the screen.
  - PDF:  
`pdf('filename', height, width)`  
`[graph commands]`  
`dev.off()`
  - Postscript: `ps('filename', horizontal=F)`
  - Bitmap: `bmp('filename')`
  - Jpeg: `jpeg('filename')`
  - PNG: `png('filename')`
  - TIFF: `tiff('filename')`
  - Windows Metafile: `win.metafile('filename')`

20 / 46

## Graphs and “vectorization”

Most of the graphical commands in **R** are “vectorized” - meaning that you can do a bunch of the same command at the same time by giving an argument a vector instead of a scalar.

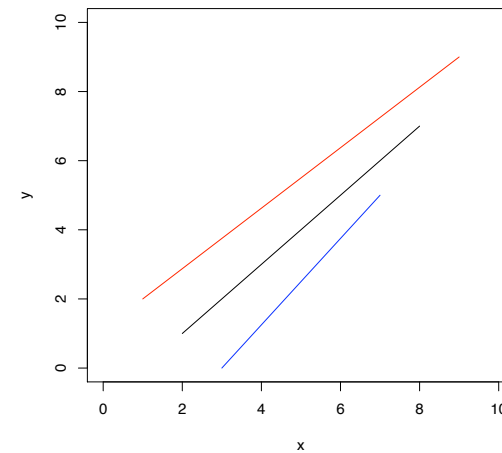
```
> plot(c(0,10), c(0,10), type="n", xlab="x", ylab="y")
> segments(c(1,2,3), c(2,1,0), c(9,8,7), c(9,7,5),
+ col=c('red','black','blue'))
```

The command above will draw three lines starting at (1,2), (2,1), and (3,0) and finishing at (9,9), (8,7), and (7,5), respectively.

We could make them different colors by adding `c('red', 'black', 'blue')`

21 / 46

## Segments with Vectorized Arguments



22 / 46

## R Graphics

Default Plotting Methods

Adding Elements to Plots

Graphs Instead of Tables

Maps in R

Plotting Regression Results

Trellis (Lattice) Graphics

Grid Graphics

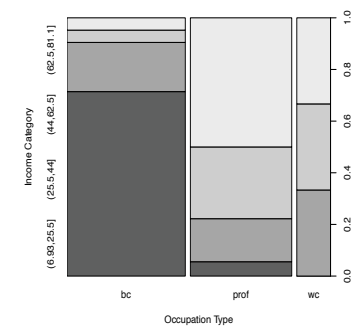
23 / 46

## Graphs Instead of Tables

Many like Cleveland (1993) and more recently in political science Kestelerc and Leoni (2007) argue that graphs are a better way than tables to present numerical information.

Tip 1: Replace Cross-Tabs with Mosaic Plots

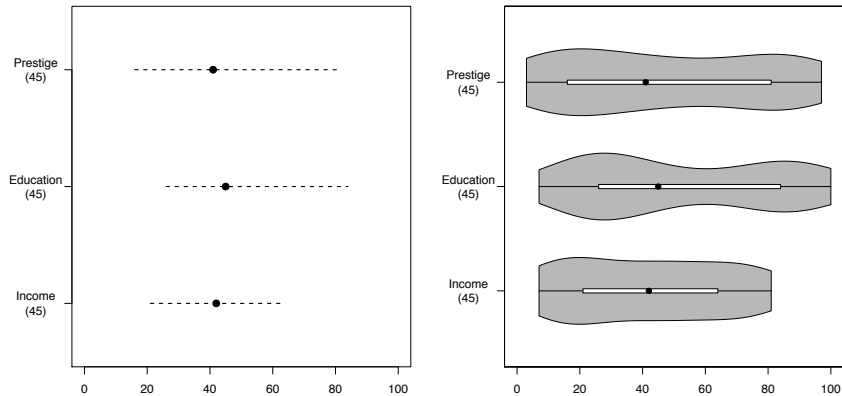
	bc	prof	wc
(6.93,25.5]	15.00	1.00	0.00
(25.5,44]	4.00	3.00	2.00
(44,62.5]	0.19	0.17	0.33
(62.5,81.1]	0.05	0.28	0.33
	1.00	9.00	2.00
	0.05	0.50	0.33



24 / 46

## Summary Statistics to Dot Plots

	min	25th	50th	75th	max
income	7.00	21.00	42.00	64.00	81.00
education	7.00	26.00	45.00	84.00	100.00
prestige	3.00	16.00	41.00	81.00	97.00



25 / 46

## Code for Dot Plot and Violin Plot

### Dot Plot

```
> ss <- t(apply(Duncan[,2:4], 2, fivenum))
> colnames(ss) <- c("min", "25th", "50th", "75th", "max")
> par(mar=c(3,6, 2,.5))
> with(Duncan, plot(c(0, max(c(income, education, prestige))), c(1,3),
+ type="n", xlab="", ylab="", axes=F))
> axis(1)
> box()
> axis(2, at=c(1.3, 2, 2.7), line=0,
+ labels=c("Income\n(45)", "Education\n(45)", "Prestige\n(45)"),
+ hadj=.5, las=1, mgp=c(0, 2.5, 0))
> points(ss[,3], c(1.3, 2, 2.7), pch=19, cex=1.25)
> segments(ss[,2], c(1.3, 2, 2.7), ss[,4], c(1.3, 2, 2.7), lty=2, lwd=1.5)
```

### Violin Plot

```
> par(mar=c(3,6, 2,.5))
> with(Duncan, plot(c(0, max(c(income, education, prestige))), c(0.5,4.5),
+ type="n", xlab="", ylab="", axes=F))
> axis(1)
> box()
> axis(2, at=c(1.25, 2.5, 3.75), line=0,
+ labels=c("Income\n(45)", "Education\n(45)", "Prestige\n(45)"),
+ hadj=.5, las=1, mgp=c(0, 2.5, 0))
> with(Duncan, vioplot(income, horizontal = T, col = "dark gray", names = c("", "", ""),
+ add = T, rectCol = "white", at=1.25, colMed="black"))
> with(Duncan, vioplot(prestige, horizontal = T, col = "dark gray", names = c("", "", ""),
+ add = T, rectCol = "white", at=3.75, colMed="black"))
> with(Duncan, vioplot(education, horizontal = T, col = "dark gray", names = c("", "", ""),
+ add = T, rectCol = "white", at=2.5, colMed="black"))
```

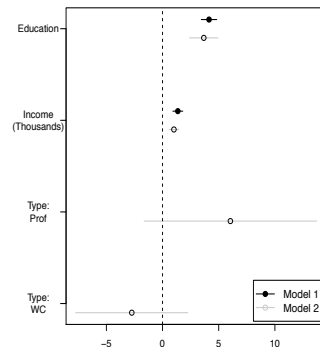
26 / 46

## Graphs Instead of Coefficient Tables

Table:

(Intercept)	-6.85*	-0.62
	(3.22)	(5.23)
education	4.14*	3.67*
	(0.35)	(0.64)
I(income/1000)	1.36*	1.01*
	(0.22)	(0.22)
typeprof		6.04
		(3.87)
typewc		-2.74
		(2.51)
N	102	98
R <sup>2</sup>	0.80	0.83
adj. R <sup>2</sup>	0.79	0.83
Resid. sd	7.81	7.09

Standard errors in parentheses  
\* indicates significance at  $p < 0.05$



27 / 46

## Code for Coefficient Plot

```
> library(apsrtable)
> data(Prestige)
> mod1 <- lm(prestige ~ education + I(income/1000), data=PreStige)
> mod2 <- lm(prestige ~ education + I(income/1000) + type, data=PreStige)
> ci1 <- confint(mod1)[-1,]
> b1 <- coef(mod1)[-1]
> ci2 <- confint(mod2)[-1,]
> b2 <- coef(mod2)[-1]
> par(mar=c(3,6, 1.5, .5))
> plot(c(min(c(ci1[,1], ci2[,1])), max(c(ci1[,2], ci2[,2]))),
+ c(.9,4.1), type="n", xlab="", ylab="", axes=F)
> segments(ci1[,1], c(4,3)+.1, ci1[,2], c(4,3)+.1, lty=1)
> segments(ci2[,1], c(4,3,2,1)-.1, ci2[,2], c(4,3,2,1)-.1,
+ lty=1, col="gray75")
> axis(1)
> box()
> varnames <- c("Education", "Income\n(Thousands)", "Type:\nProf", "Type:\nWC")
> axis(2, at=c(4,3,2,1), line=0, labels=varnames, hadj=.5, las=1, mgp=c(0, 2.5, 0))
> points(b1, c(4,3)+.1, pch=19, cex=1.1)
> points(b2, c(4,3,2,1)-.1, cex=1.1)
> abline(v=0, lty=2)
> legend("bottomright", c("Model 1", "Model 2"), lty=c(1,1),
+ col=c("black", "gray75"), pch=c(19,1), cex=1.1, inset=.01)
```

28 / 46

## R Graphics

Default Plotting Methods

Adding Elements to Plots

Graphs Instead of Tables

## Maps in R

Plotting Regression Results

Trellis (Lattice) Graphics

Grid Graphics

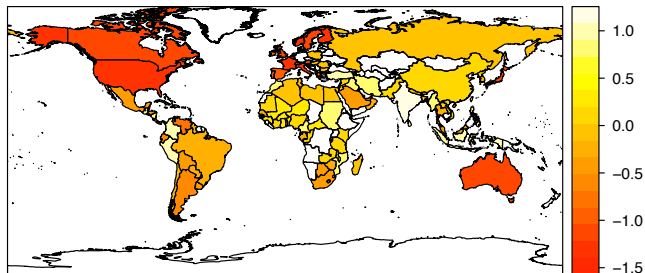
29 / 46

## Maps in R

- **R** has the capability to read in shape files and produce maps with relative ease.
- These are not quite as pretty and polished as maps you could make in Arc/GIS, but they may be easier, and certainly cheaper if you don't have GIS software already.
- First, you need a shapefile or polygon file. These are most readily available either for the US or the World. Other individual countries could likely be found with a google search.

```
> library(maptools)
> world <- readShapePoly("WorldCountries.shp")
> dat <- read.dta("hwl_new.dta")
> mod <- lm(repression ~ vdisssum + rgdpch + democracy*vdisssum, data=dat)
> preds <- predict(mod)
> names(preds) <- with(dat, ccode)
> world$pred_rep <- preds[match(world$ccode, names(preds))]
> print(splot(world["pred_rep"], col.regions=heat.colors(16)))
```

30 / 46



31 / 46

## R Graphics

Default Plotting Methods

Adding Elements to Plots

Graphs Instead of Tables

Maps in R

Plotting Regression Results

Trellis (Lattice) Graphics

Grid Graphics

32 / 46

## Plotting Regression Predictions

Let's generate the model and get predictions holding education constant at its mean, changing income over its range for each type of occupation:

```
> library(car)
> data(Duncan)
> mod <- lm(prestige ~ income + education + type, data=Duncan)
> income.seq <- with(Duncan, seq(from=min(income, na.rm=T),
+ to=max(income, na.rm=T), length=100))
> bc.dat <- with(Duncan, data.frame(income=income.seq, education=
+ mean(education, na.rm=T), type="bc"))
> prof.dat <- with(Duncan, data.frame(income=income.seq, education=
+ mean(education, na.rm=T), type="prof"))
> wc.dat <- with(Duncan, data.frame(income=income.seq, education=
+ mean(education, na.rm=T), type="wc"))
> bc.pred <- as.data.frame(predict(mod, bc.dat, interval="confidence"))
> prof.pred <- as.data.frame(predict(mod, prof.dat, interval="confidence"))
> wc.pred <- as.data.frame(predict(mod, wc.dat, interval="confidence"))
```

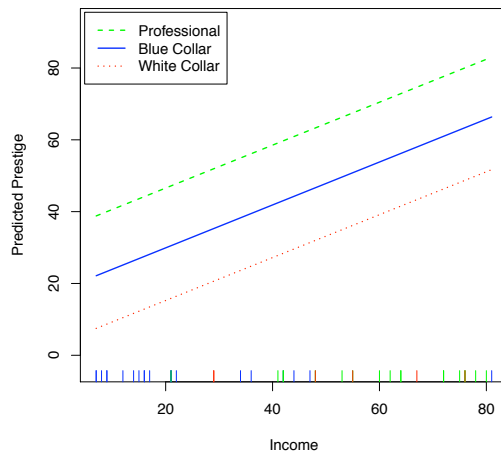
33 / 46

## Code for Regression Graph: Making the Graph

```
> plot(bc.pred[["fit"]] ~ bc.dat[["income"]], type="l", lwd=1.5, lty=1, col="blue",
+ xlab="Income", ylab="Predicted Prestige", ylim=
+ c(min(bc.pred[["lwr"]], wc.pred[["lwr"]], prof.pred[["lwr"]]),
+ max(bc.pred[["upr"]], wc.pred[["upr"]], prof.pred[["upr"]]))
> lines(prof.dat[["income"]], prof.pred[["fit"]], lty=2, lwd=1.5, col="green")
> lines(wc.dat[["income"]], wc.pred[["fit"]], lty=3, lwd=1.5, col="red")
> legend("topleft", c("Professional", "Blue Collar", "White Collar"),
+ lty=c(2,1,3), lwd=rep(1.5,3), col=c("green", "blue", "red"),
+ inset=.01)
> rug(Duncan[Duncan[["type"]] == "bc", "income"], col="blue")
> rug(Duncan[Duncan[["type"]] == "prof", "income"], col="green")
> rug(Duncan[Duncan[["type"]] == "wc", "income"], col="red")
```

34 / 46

## Prediction Graph (No Intervals)



35 / 46

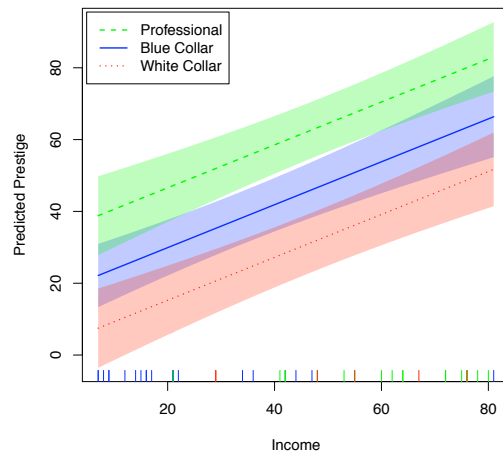
## Plotting Prediction Intervals

By adding the following few commands, you can add a prediction interval:

```
> plot(bc.pred[["fit"]] ~ bc.dat[["income"]], type="l", lwd=1.5, lty=1, col="blue",
+ xlab="Income", ylab="Predicted Prestige", ylim=
+ c(min(bc.pred[["lwr"]], wc.pred[["lwr"]], prof.pred[["lwr"]]),
+ max(bc.pred[["upr"]], wc.pred[["upr"]], prof.pred[["upr"]]))
> lines(prof.dat[["income"]], prof.pred[["fit"]], lty=2, lwd=1.5, col="green")
> lines(wc.dat[["income"]], wc.pred[["fit"]], lty=3, lwd=1.5, col="red")
> legend("topleft", c("Professional", "Blue Collar", "White Collar"),
+ lty=c(2,1,3), lwd=rep(1.5,3), col=c("green", "blue", "red"),
+ inset=.01)
> rug(Duncan[Duncan[["type"]] == "bc", "income"], col="blue")
> rug(Duncan[Duncan[["type"]] == "prof", "income"], col="green")
> rug(Duncan[Duncan[["type"]] == "wc", "income"], col="red")
> with(bc.pred, polygon(c(income.seq, rev(income.seq), income.seq[1]), c(lwr,
+ rev(upr), lwr[1]), col= rgb(0,0,1,.25, 1), border=NA))
> with(prof.pred, polygon(c(income.seq, rev(income.seq), income.seq[1]), c(lwr,
+ rev(upr), lwr[1]), col= rgb(0,1,0,.25, 1), border=NA))
> with(wc.pred, polygon(c(income.seq, rev(income.seq), income.seq[1]), c(lwr,
+ rev(upr), lwr[1]), col= rgb(1,0,0,.25, 1), border=NA))
```

36 / 46

## Predictions with Confidence Bands



37 / 46

## R Graphics

Default Plotting Methods

Adding Elements to Plots

Graphs Instead of Tables

Maps in R

Plotting Regression Results

Trellis (Lattice) Graphics

Grid Graphics

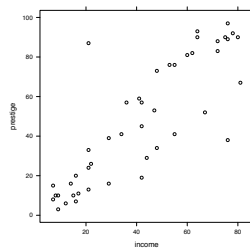
38 / 46

## Trellis Graphics

This example gives us a nice segue into the lattice graphics system.

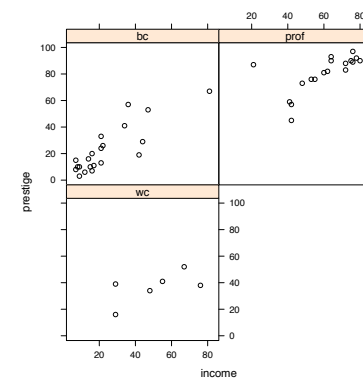
- `xyplot()` is the workhorse of the the lattice package.
- We can make a simple scatterplot as follows:

```
> library(lattice)
> print(xyplot
```



39 / 46

```
> print(xyplot
```



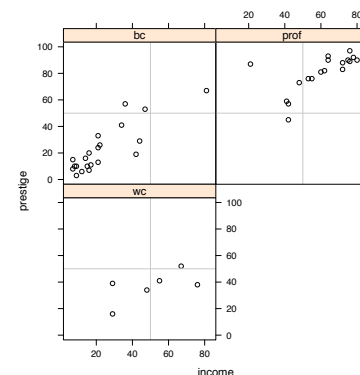
40 / 46

## Panels in Lattice Graphs

- Lattice knows what to do in each panel through a series of panel commands.
- The panel command does the same thing in each in each panel.
- It is a bit tricky to get it to do variations on the same theme in each panel (when the tasks differ across the panels by some attribute other than those used to generate the plot).
- By adding the `panel.abline()` command we can get the graph on the next page.

41 / 46

```
> print(xyplot(prestige ~ income | type, data= Duncan, as.table=T,  
+   panel=function(x,y){  
+     panel.points(x,y, col="black")  
+     panel.abline(h=50, v=50, col="gray75")  
+   }  
+ ))
```



42 / 46

## R Graphics

Default Plotting Methods

Adding Elements to Plots

Graphs Instead of Tables

Maps in R

Plotting Regression Results

Trellis (Lattice) Graphics

Grid Graphics

43 / 46

## Grid Graphics

- The Grid graphics system encompasses the Trellis (Lattice) system.
- Lattice does some data pre-processing and the like (a non-trivial task), but Lattice uses grid objects to build its graphs.
- Given the lack of space to do this today, I'm not going to go into Grid graphics, however there are a couple of good chapters on this in Murrell (2006)
- I haven't found anything yet that I couldn't do in either in the traditional graphics system (usually) or lattice (when dealing with grouped data).

44 / 46

## Readings

### Today:

- \* Fox (2002) Chapter 7
- \* Murrell (2006) Chapters 1-4
- \* Jacoby (1997, 1998, 2006)
- \* Kastlelec and Leoni (2007)
- Venables and Ripley (2002) Chapter 4

### Tomorrow: OLS II: Effective Presentation

- \* Firth (2003)
- Firth and Menzes (2004)
- \* Brambor, Clark and Golder (2006)
- \* Braumoeller (2004)
- Kam and Franzese (2007)
- \* Silber, Rosenbaum and Ross (1995)

Brambor, Thomas, William Clark and Matt Golder. 2006. "Understanding Interaction Models: Improving Empirical Analyses." *Political Analysis* 14(1):63–82.

Braumoeller, Bear F. 2004. "Hypothesis Testing and Multiplicative Interaction Terms." *International Organization* 58(4):807–820.

Cleveland, William S. 1993. *Visualizing Data*. Summit, NJ: Hobart Press.

Firth, David. 2003. "Overcoming the Reference Category Problem in the Presentation of Statistical Models." *Sociological Methodology* 33:1–18.

Firth, David and Renee X. De Menzes. 2004. "Quasi-Variations." *Biometrika* 91(1):65–80.

Fox, John. 2002. *An R and S-Plus Companion to Applied Regression*. Thousand Oaks: Sage Publications.

Jacoby, William G. 1997. *Statistical Graphics for Univariate and Bivariate Data*. Thousand Oaks, CA: Sage.

Jacoby, William G. 1998. *Statistical Graphics for Visualizing Multivariate Data*. Thousand Oaks, CA: Sage.

Jacoby, William G. 2006. "The Dot Plot: A Graphical Display for Labeled QUantitative Values." *The Political Methodologist* 14(1):6–14.

Kam, Cindy and Robert J. Franzese. 2007. *Modeling and Interpreting Interactive Hypotheses in Regression Analyses*. Ann Arbor: University of Michigan Press.

Kastlelec, Jonathan P and Eduardo L. Leoni. 2007. "Using Graphs Instead of Tables in Political Science." *Perspectives on Politics* 5(4):755–771.

Murrell, Paul. 2006. *R Graphics*. Boca Raton, FL: Chapman & Hall/CRC.

Silber, Jeffrey H., Paul R. Rosenbaum and Richard N. Ross. 1995. "Comparing the Contributions of Groups of Predictors: Which Outcomes Vary with Hospital Rather Than Patient Characteristics." *Journal of the American Statistical Association* 90(429):7–18.

Venables, William N. and Brian D. Ripley. 2002. *Modern Applied Statistics with S, 4<sup>th</sup> edition*. 3 ed. New York: Springer.