

Demography 213

Lattice graphics

Carl Mason
carlm@demog.berkeley.edu

October 24, 2011

Contents

1 Lattice Graphics	1
2 Assignment	2
2.1 Read most of Lattice Graphics: an Introduction	2
2.1.1 Cutting and pasting from pdf to Emacs	2
2.2 IPUMS yet again	4
2.3 Hints and advice	5

1 Lattice Graphics

Lattice graphics are a collection of clever high level graphics commands for displaying relationships among 3 or more variables. Given the usual 2 dimension limit that paper imposes on us, this is indeed a challenge for the creative scientist.

Lattice graphics functions do this generally, by presenting an array of similar graphs showing a relationship between two variables (e.g. a scatter plot) for cleverly selected subsets of the other variables. The clever subsetting of the data is referred to as *conditioning*—think of “conditional” probability.

Figure 1 shows an example of a `xyplot()` function. *xyplot* is the lattice version of the basic scatter plot produced by the `plot()` function.

Notice that in Figure 1 in place of a single scatter plot, we have 24 plots. Each of the 24 plots show the relationship between “income” and “height” (on the planet Zorch) for a subset of the data. The lowest and leftest plot shows the relationship only for members of the “Amana” religion having a certain range of IQ score. The range of IQ is shown by the darker shading in the region above the plot. Since all 24 plots have the same X and Y range, you can scan across the rows or up and down the columns to compare the shapes and general levels of the relationship between income and height *conditioning* on religion and IQ.

If you scan up any column, you might notice that the IQ range displayed in each of the plots substantially overlap the range in the plots above and below. This represents a sort of compromise between treating IQ as a categorical, vs continuous variable.

There are many other lattice plots which share many syntactical features. All of them require that you load the lattice package via the command:

```
■ > library(lattice)
```

The lattice graphics functions are a bit less intuitive than the graphics functions that we learned last week (if you can believe that). But once you get the hang of `xypot()` you will find the others work the same way.

Note that because lattice graphics generally create many similar graphs at once, the wonderful paradigm of creating a graph and then adding stuff to it is lost. I am sad about this, but so it goes.

2 Assignment

As usual, begin by creating a subdirectory for the week in your 213 directory; cd'ing into that directory and launching emacs so as to create a new buffer with a name like `exN.r`.

2.1 Read most of Lattice Graphics: an Introduction

This document by Deepayan Sakar does a nice job of introducing the big ideas of Lattice Graphics and give nice examples which you can do as you read it. Here's the URL http://www.bioconductor.org/help/course-materials/2008/advanced_R/latticeLab.pdf

See Section 2.1.1 for instructions on how to cut and paste between a pdf and Emacs.

As always when reading computer related stuff, the most important thing is to understand what is possible.

2.1.1 Cutting and pasting from pdf to Emacs

Lattice Graphics: an Introduction is a pdf which contains several example R commands that you might like to execute. Although they are short enough to simply retype – that would be the cowardly thing to do. It may require a little experimentation, but copying from a pdf file into Emacs does work.

Because you are working in an X11 window system, you can generally simply highlight the text that you want (by dragging the mouse over it with the `LEFT` button depressed). Once the text is highlighted you should be able to drop in into Emacs or any other window by way of the middle button.

There are, however, two annoyances:

```

> load(file = "PlanetZorch.rsave")
> library(lattice)
> print(xyplot(income ~ height | religion + equal.count(IQ, overlap = 0.75),
+   data = society, type = c("p", "smooth"), cex = 0.5, col = "black",
+   col.line = "blue"))

```

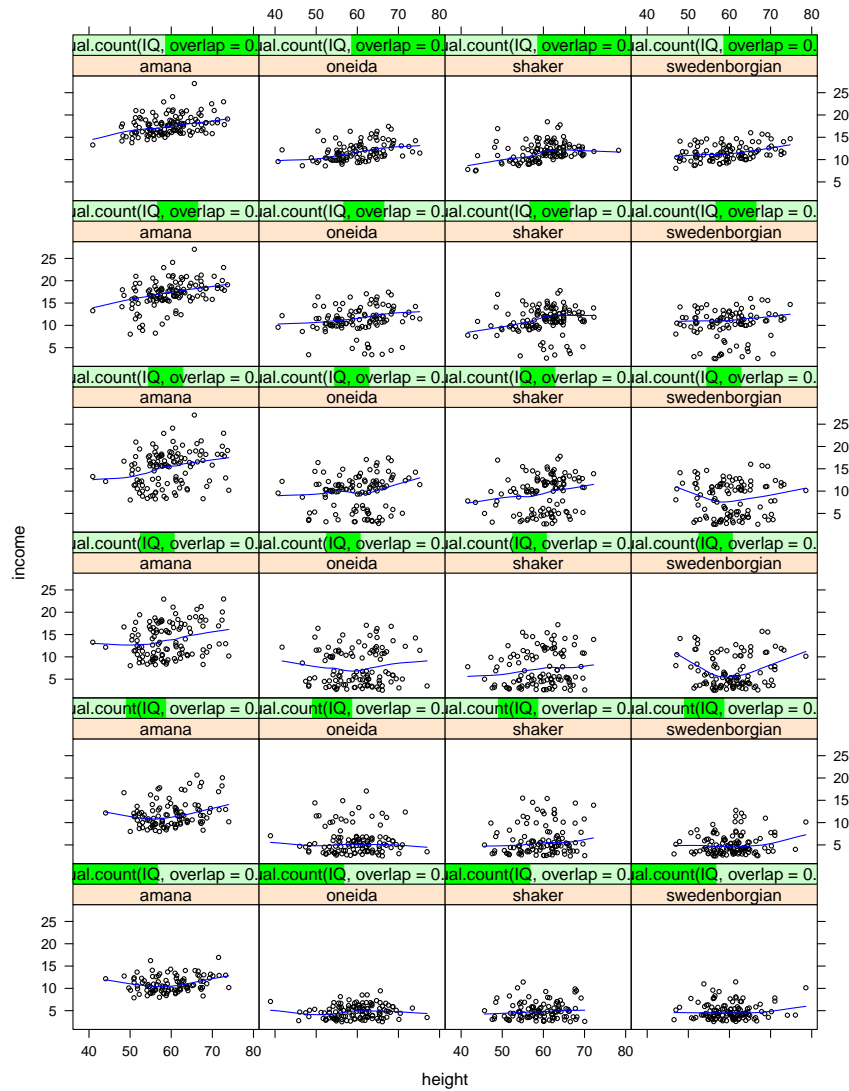


Figure 1: an xyplot showing relationship among four variables

1. When you drop the text into Emacs via the `MIDDLE` button, it can windup in unexpected places. This is because there are two pointer locations in Emacs – one for the mouse and the other for keyboard input. You generally want your pasted text to enter as though you typed it ... rather than wherever the mouse happens to be. The solution to this problem is to use `C-y` rather than the `MIDDLE` button to “paste” the text.
2. Although selecting with the `LEFT` button usually works, sometimes it does not, and then one must resort to the the Windows convention of using either `Ctrl+c` or a choice under the Edit menu (generally “Copy”) to select text for pasting elsewhere. Most of the pdf displaying functions support one or more of these procedures. Once you have done the `Ctrl+c` dance, then you should be able to paste the text into the Emacs window via `C-y`.

The point is that the method for cutting and pasting between pdfs and Emacs is not consistent – it depends on the how the pdf is displayed and on the mood of the computer that you are working on and also whether you are working on a Linux machine or on a windows machine with it’s own peculiar X11 server. But it is possible—even easy to cut and paste between pdfs and Emacs, it simply takes some experimenting.

2.2 IPUMS yet again

Explore some real IPUMS census data graphically. Once again, the main point is learning how to use the lattice graphics system making an original and significant contribution to science is optional.

When you construct your IPUMS extract, you will thank yourself if you include the variable `statefip`. The easiest way to speed up your work without sacrificing coherence, is to reduce your data set to a single state.

Please be sure to include some comments with your code describing briefly what your graphs do or say.

Now, using the ACS data, create at least one of each of the following types of trellis plots each of which should reveal an important relationship¹in the data.

- `bwplot`
- `xyplot`
- `histogram`
- `dotplot`

¹well, a relationship anyway

2.3 Hints and advice

You are now an expert on all things related to IPUMS, but I shall nonetheless presume to remind you of a few things:

- In your extract, it cannot hurt to include the `statefip` variable as this can allow you to easily select a single state to work on – at least while you perfect your code.
- After importing the data from your `.dta` file, many factors include levels that with zero observations. These phantom levels can be removed most easily by running:

```
■ > acs$nameOfVariable<-factor(as.character(acs$nameOfVariable
```

- Many IPUMS variables have missing value codes that are numeric. Check the the codebook and then turn those missing values into NAs:

```
■ > acs$someVariable[acs$someVariable == 99999999999999] <-NA
```