The Political Methodologist

Newsletter of the Political Methodology Section
American Political Science Association
Volume 13, Number 2, Fall 2005

Editors:
Adam J. Berinsky, Massachusetts Institute of Technology
berinsky@mit.edu
Michael C. Herron, Dartmouth College
Michael.C.Herron@dartmouth.edu
Jeffrey B. Lewis, University of California Los Angeles
jblewis@ucla.edu

Editorial Assistant:
Matthew Atkinson, University of California Los Angeles
matthewa@ucla.edu

Contents

Notes From the Editors

Welcome back to TPM. Our issue leads off with three articles in the computing section. Luke Keele walks us through 3-D graphics in R. Based on his own process of trial and error, this article will be useful for novices and experienced users alike. We then move to an overview of the new version of STATA reviewed, as usual, by Neal Beck. Finally, Christopher Lawrence and Dick Eddelbuttel introduce Quantian–an open source computing package.

We then turn to two articles about a topic near and dear (financially speaking) to all our hearts—data entry. Michael Herron discusses how to use Xemacs macros to convert ASCII data files to formats that can be easily imported into spreadsheets or statistical packages. Seth Masket, in turn, describes the ins and outs of hiring overseas professionals to input your data.

We also have in this issue a review of Box-Steffensmeier and Jones' recent book on Event History Models. This book is one of the first in the Analytical Methods for Social Research series, published by Cambridge University Press. We plan to review additional titles from this series in future issues.

We close with a note from our section President, Janet Box-Steffensmeier, and the call for participation for the 2006 Society for Political Methodology Summer Meeting.

The next issue of TPM is beginning to take shape, but we are always on the lookout for more material. Your submissions and ideas for topics to address are most welcome.

The Editors

Computing and Software

3-D Graphics in R

Luke Keele
The Ohio State University
luke.keele@politics.ox.ac.uk

Three dimensional graphics may not be widely used in published work, but in political methodology, three–dimensional plots have a number of uses. The plotting of likelihoods and response surfaces from Monte Carlo experiments are examples of where three–dimensional plots are quite useful.

The options for software are not extensive, however. Excel can produce such graphs but the quality of the plots is not excellent. One would assume that R would be a natural choice for producing such plots given R's generally excellent graphing capabilities. And it is a good choice, but to get publication quality 3-D graphics requires more work than typically required when plotting in R. What I hope to do, here, is save others the time it took me to figure out how to produce quality graphics with the `wireframe` command.

The standard 3-D plot command in R is `persp`. While one can produce good looking plots with `persp`, it has a couple of key deficiencies when it comes to producing publication quality plots. First, the axes do not have the full functionality typical in most R plots. One cannot adjust the tick marks and must rely on automatic axis creation. Second, one cannot use plotmath characters as axis labels. Therefore, one is stuck with "Alpha" instead of "α" for axis labels. While these may seem like small deficiencies, they are problematic when producing publication quality plots.

The next option in R is to use the `wireframe` command from the `lattice` package. The `lattice` library is an R implementation of the `trellis` graphics package in S-plus. While the `lattice` package can do a variety of plots, it is primarily designed to plot relationships between three variables. Typically, this is done with the `xyplot` command to generate scatterplots between two variables conditional on the values of a third variable. The focus on this type of plot is unfortunate, since it means that surface plots as produced in `wireframe` tend to be overlooked in the documentation. In fact, the documentation for all the `lattice` graphics commands is sparse, and for `wireframe` is particularly thin.[1] This is doubly unfortunate since `lattice\trellis` commands tend to be complex.

To start with, let's discuss how one's data needs to be set up. For a single 3-D plot, one needs three vectors of data: two which define the values of the x and y axes and one for the quantity of interest to be plotted. For users familiar with `persp`, this is an important difference. In `persp`, the response variable needs to be in matrix not vector form. To use `wireframe`, the response variable must be in vector form. Fortunately, for many users it is probably more convenient to have data in vector form.

As an example, I use data from a Monte Carlo experiment on misspecification in an OLS regression. I want to plot the bias as a function of two parameters: α and ϕ.

I need my data in R to take the following form:

```
> ols[1:20,]
      a       bias  phi
1  0.05 0.01716063 0.05
2  0.10 0.04543059 0.05
3  0.15 0.06780803 0.05
4  0.20 0.09976802 0.05
5  0.25 0.12303042 0.05
6  0.30 0.15644003 0.05
7  0.35 0.19504754 0.05
8  0.40 0.22869483 0.05
9  0.45 0.27677437 0.05
10 0.50 0.32982011 0.05
11 0.55 0.39086068 0.05
12 0.60 0.45282179 0.05
13 0.65 0.54175071 0.05
14 0.70 0.61998347 0.05
15 0.75 0.72936067 0.05
16 0.05 0.02583256 0.10
17 0.10 0.04417769 0.10
18 0.15 0.06806339 0.10
19 0.20 0.09310019 0.10
20 0.25 0.12514266 0.10
```

The two parameters, α and ϕ, both run from 0.05 to 0.75 in increments of 0.05. If one only has the response variable in vector format, the `expand.grid` command can be used to create the parameter vectors. See the R documentation or Fox (2002) for examples. Notice that I have used close increments between the values of α and ϕ. This

[1] One can use the freely available `trellis` manual when using `lattice`, but it too only gives sparse consideration to the the `wireframe` command.

will make the surface plot smoother. Next, I need to load the `lattice` package:

```
> library(lattice)
```

To change the background color of the plotting area to white from the default grey, I use a high level plotting command:

```
> trellis.par.set(theme = col.whitebg())
```

The command `trellis.par.set` can be used to set a large number of graphing parameters. A call to `trellis.par.get()` will list at least 50 default graphing parameters and their values. Now let's produce a basic 3-D plot with the `wireframe` command:

```
> fig1 <- wireframe(bias ~ a * phi,
+         data = ols, screen =
+         list(z = -245, x = -75),
+         xlab = expression(paste(alpha)),
+         ylab = expression(paste(phi)),
+         zlab = "Bias",
+         zlim = range(seq(0.0, 0.85,
+         by=0.20)))
```

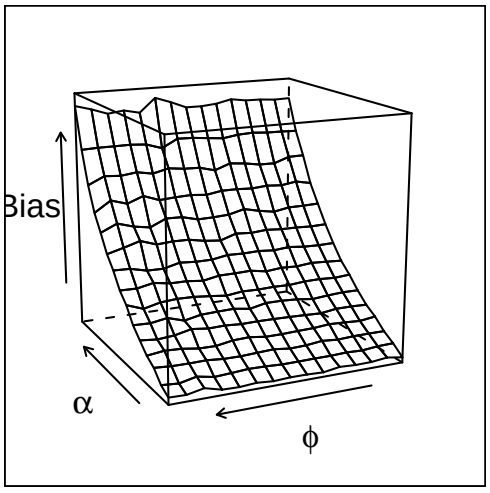

Figure 1: Basic Plot

Many of the normal plotting options such as `xlab` and `ylab` work as expected. One obvious addition to these commands is `zlab` and `zlim`. These two options work as one would expect them to. Notice also that I can include `plotmath` characters. One extra option available for all the label commands in the `rot` option. This rotates the axis labels. This option is only available if you are using R version 2.1.0 or newer. To rotate the z-axis label you would use the following: `zlab = list(label = "Bias", rot = 90)`. This is useful when the z-axis label is very long, or to rotate the x- and y-axis labels to be parallel to the axis when the plot is at an angle. Other changes to the axis labels can be made in a similar fashion. See the next plot for an example.

Let's start to clean the plot up. First, you probably noticed the box that goes around the outside of the entire plot. Removing it is quite easy, but finding the command to do so is not. It is controlled by a high level command for the axes. To remove it, I use `trellis.par.set`. I also clean up the axis labeling in the next example:

```
> trellis.par.set("axis.line",
+ list(col="transparent"))

> fig2 <- wireframe(bias ~ a * phi,
+      data = ols,
+      screen = list(z = -245, x = -75),
+      xlab = expression(paste(alpha)),
+      ylab = expression(paste(phi)),
+      zlab = list(label = "Bias",
+      font = 1, cex = 0.60),
+      zlim = range(seq(0.0, 0.85,
+      by=0.20)))
```

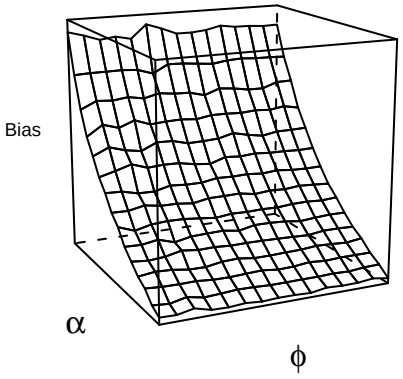

Figure 2: Removing The Border and Changing Axis Features

The call to `trellis.par.set` removes the box around the plot. Notice now that the axis tick marks are also gone. This is because the command I used to remove the border is an axis command. We'll get the tick marks back in a moment. Let's now work through the more important options available in `wireframe`.

The most important option is probably `screen`, which controls the rotation of the plot. The first value is the z value which rotates the plot in either clockwise or counter-clockwise directions around a vertical axis. A value

of 0 situates the plot so it sits perpendicular to the viewer. 180 or -180 will rotate the plot 180 degrees from the 0 angle in either direction. I find that a corner view is often best, but it will depend on the values of the surface you are plotting. For example, compare a head–on view in Figure 3 to the viewing angle in Figure 2, where the viewing angle of the plot is set such that we view the surface from the corner:

```
> fig3 <- wireframe(bias ~ a * phi,
+        data = ols,
+        screen = list(z = 0, x = -75),
+        xlab = expression(paste(alpha)),
+        ylab = expression(paste(phi)),
+        zlab = list(label = "Bias",
+        font = 1, cex = 0.60),
+        zlim = range(seq(0.0, 0.85,
+        by=0.20)))
```

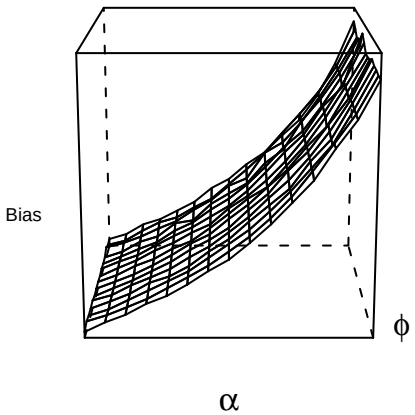

Figure 3: Rotating the plot, Z set to 0

In general, I find values around 50, 150, etc. to be quite useful. In Figure 2, for example, I use -245. Generally some experimentation is required depending on the plot.[2] The x option rotates the plot about a horizontal axis. The value 90 or -90, makes it flat so to speak as in Figure 4. Values of 0 or 180 will will show it from the top or the bottom, but this usually isn't very useful. I find a value of -75, a slight angle, is a good viewing angle but this may vary from plot to plot.

```
> fig4 <- wireframe(bias ~ a * phi,
+        data = ols,
+        screen = list(z = -245, x = 90),
+        xlab = expression(paste(alpha)),
+        ylab = expression(paste(phi)),
+        zlab = list(label = "Bias",
+        font = 1, cex = 0.60),
+        zlim = range(seq(0.0, 0.85,
+        by=0.20)))
```

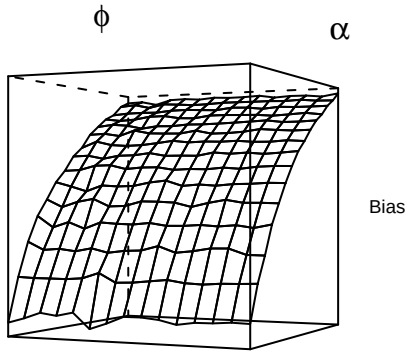

Figure 4: The Plot At A Level Viewing Angle

Another important option command is `scales`. It controls the tick marks and the labels for the tick marks. Table 1 contains a list of the options for the scales command. The list in Table 1 is not exhaustive but contains the options I find most useful.

Table 1: Parameters of Scale Option

cex	Font Size: 1 is Default
font	1 Plain, 2 Bold, 3 Italic, 4 Bold Italic
tck	Tick Length: 1 Default
col	Color: "black" Draws tick marks
arrows	Draws Arrows Instead of Axis
draw	Draw Axis

In the next plot, I use the `scales` option to put the tick marks back on the plot by adding "black" as a color, and I adjust the size of the tick mark labels and the font.

```
> fig5 <- wireframe(bias ~ a * phi,
+        data = ols,
+        scales = list(arrows=FALSE,
+        cex= .45, col = "black",
+        font = 3, tck),
+        screen = list(z = -245, x = -75),
+        xlab = expression(paste(alpha)),
+        ylab = expression(paste(phi)),
+        zlab = list(label = "Bias",
+        font = 1, cex = 0.60), zlim =
+        range(seq(0.0, 0.85, by=0.20)))
```

[2]There is an R library called RGL to make graphs interactive including the ability to rotate the plot with the mouse.

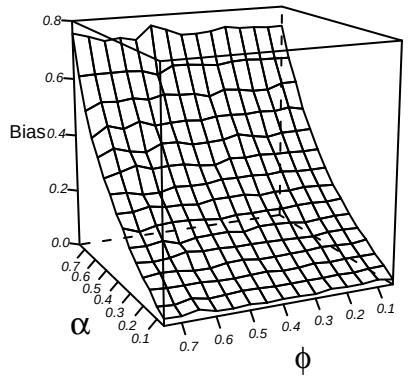

Figure 5: Controlling the Axis

Next, we'll cover some options for filling in the surface. There are two options for the surface. The first is to use the `drape` option. This adds a grid and colors to the surface. The color is chosen by R and is set to shades of red and orange. The next plot demonstrates this default coloring.

```
> fig6 <- wireframe(bias ~ a * phi,
=         data = ols,
+         scales = list(arrows=FALSE,
+         cex= .45, col = "black", font = 3),
+         drape = TRUE, colorkey = FALSE,
+         screen = list(z = -245, x = -75),
+         xlab = expression(paste(alpha)),
+         ylab = expression(paste(phi)),
+         zlab = list(label = "Bias",
+         font = 1, cex = 0.60),
+         zlim = range(seq(0.0, 0.85,
+         by=0.20)))
```

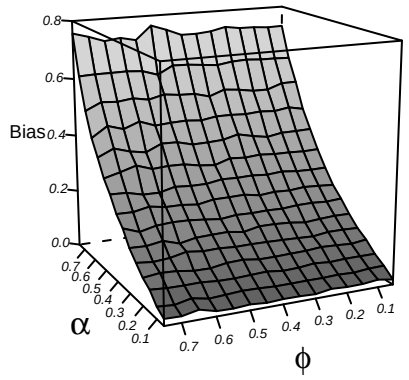

Figure 6: Adding Color To The Surface Area

This looks fine when printed in black and white. The `drape` coloring doesn't accentuate the actual surface much.

To add colors, first add the option `shade = TRUE`, this option automatically overrides the `drape` option even if it is included. The default color is a rainbow like pattern. One may add a `shade.colors` function to produce any color. Figure 7 adds grey coloring. The code for `shade.colors` is complicated but to adjust the shade of grey, one need not understand all the settings with the exception of the w parameter[3] I have it set to 0.5, which produces a medium-light grey. Shifting it downward toward 0 darkens the grey and increasing it lightens the grey. Using this `shade.colors` function yields a nice looking surface when printed in black and white, as can be seen below. The grey color scheme tends to emphasize how smooth or rough the surface is. Setting `shade` to `FALSE` produces a grid surface with no color.

```
> fig7 <- wireframe(bias ~ a * phi,
+         data = ols,
+         scales = list(arrows=FALSE,
+         cex= 0.45, col = "black",
+         font = 3, tck = 1),
+         screen = list(z = -245, x = -75),
+         colorkey = FALSE,
+         xlab = expression(paste(alpha)),
+         ylab = expression(paste(phi)),
+         zlab = list(label = "Bias",
+         font = 1, cex = 0.60),
+         shade=TRUE,
+         light.source= c(0,10,10),
+         zlim = range(seq(0.0, 0.85,
+         by=0.20)),
+         shade.colors = function(irr, ref,
+         height, w = 0.4)
+         grey(w * irr + (1 - w) *
+         (1 - (1 - ref)^0.4)))
```

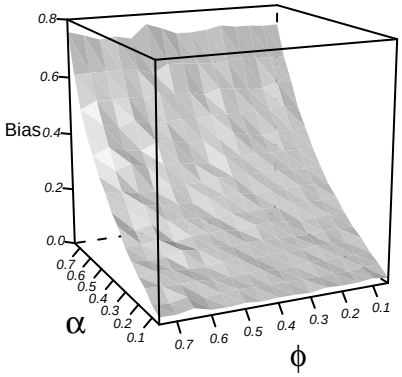

Figure 7: Greyscale Coloring For the Surface Area

[3]See the documentation for `panel.cloud` for a description of the parameters in this color function.

There is one last option that is of only limited usefulness for a grey surface. The `lightsource` option controls the direction of the lighting in the plot, and is accompanied by a vector of Cartesian coordinates as seen in the above examples. With a grey surface changing these values only serves to darken different parts of the surface. Changing the first coordinate darkens the entire surface, changing the second value darkens one end and changing the third darkens the other end. One can use the `lightsource` option to further accentuate the roughness of the surface, if so desired.

Another useful command is the `zoom` option. The default is set to 0.75. Decreasing it will shrink the plot. Increasing it much above 1 makes the plot bigger than is probably ever useful unless for a poster display. One can also adjust the aspect ratio. The `aspect= c(1,1)` option takes two values. When set to (1,1) the plot is exactly square. Decreasing the first value stretches the length of the plot, while decreasing the second value stretches the height of the plot. In the next example, I stretch the length of the plot.

```
> fig8 <- wireframe(bias ~ a * phi,
+         data = ols, scales =
+         list(arrows=FALSE, cex= 0.45,
+         col = "black", font = 3, tck = 1),
+         screen = list(z = -245, x = -75),
+         colorkey = FALSE,
+         xlab = expression(paste(alpha)),
+         ylab = expression(paste(phi)),
+         zlab = list(label = "Bias",
+         font = 1, cex = 0.60),
+         shade=TRUE,
+         light.source= c(0,10,10),
+         zlim = range(seq(0.0, 0.85,
+         by=0.20)),
+         shade.colors = function(irr, ref,
+         height, w = 0.4)
+         grey(w * irr + (1 - w) *
+         (1 - (1 - ref)^0.4)),
+         aspect = c(1, 0.65))
```

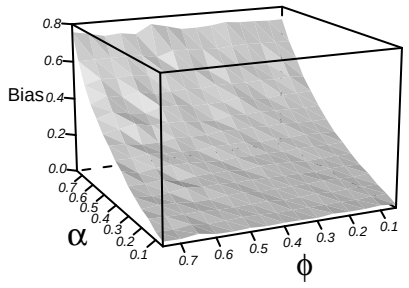

Figure 8: Changing the Aspect Ratio

Finally, I turn to getting your `lattice` plot out of R and into your favorite word processing program. If using the GUI version of R in Windows, one can always right-click and save on the plot device and save the plot as a Windows metafile or postscript file. This works fine but doesn't give the user any control over the size of the plot other than eyeballing it. Moreover, this method isn't useful for those who don't run the GUI Windows version of R. When using `lattice` graphics one can't use the usual graphics devices such as `postscript`. In truth, they will still work but the results are unsatisfying. Instead the `lattice` package has its own graphics device called `trellis.device`. Fortunately, it works much like the other graphics devices. Below is a code snippet the contains an example:

```
> trellis.device(postscript,
+    file="3dfig8.eps",
+    onefile = FALSE, paper = "special",
+    horizontal = FALSE,
+    width = 4, height = 4)
> trellis.par.set("axis.line",
+    list(col="transparent"))
> print(fig8)
> dev.off()
```

In the example above, I print a postscript file, but `trellis.device` supports Windows metafiles, Mac, png, pdf, x11, and a few others. The documentation doesn't say this, but all the options from the standard graphics devices work as normal. When `trellis.device` is invoked the high level options are reset, which is why they must be included as in the example above. These can be set within the `trellis.device` command but I find it easier to keep track of the changes by making separate calls after the device is turned on. Finally, I use the `print` command to send a copy of the plot to my working directory. The `print` command has added functionality in that it can be used to print more than one plot in a grid layout. While useful in theory, trying to fit more than one or two plots on a page usually looks bad.

One can have much finer control of `wireframe` plots, but this requires exhaustive reading of the dense and unhelpful documentation. The purpose, here, has been to give the reader enough to produce publication quality plots without being driven to distraction by the `lattice` documentation.

References

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

Stata 9: Another First Look

Nathaniel Beck
New York University
nathaniel.beck@nyu.edu

As most readers of TPM are aware, Stata Corp. began shipping Version 9 in early spring (2005). Following a long standing tradition (going back at least one version), I would like to share my first impressions of V9. The note is from my perspective of how I use Stata in both teaching and research, and my guesses about common uses by readers of TPM. I clearly give short shrift to things that I do not plan to use (and my guess is that are less commonly used in our discipline).

V9 is still very clearly Stata. Stata has a very nice philosophy about updating to new versions: anything new that you do not care about is very unlikely to get in your way. A user of V8 who happened to walk into a V9 equipped lab could happily work under the assumption that the lab had V8 (though the lack of some annoying error messages would signal some improvement). And the modular construction of most of Stata means that almost all of the new features require no resources unless those features are used. (The footprint of V9 is a tad bigger than that of V8, but is still, by current standard, very small.)

This is not to say that Stata has not improved the interface. The user who wants to take advantage of those improvements may be much happier using V9 than V8. Everyone has pet peeves about interfaces; mine was that Stata did not allow one bunch of settings for use on a laptop and then another, with much larger fonts, for projecting that laptop in a classroom. Fiddling with fontsizes in front of an audience is not my idea of fun. V9 allows for multiple preference files, so one can have one set of preferences for ideal laptop usage and another for ideal projection usage. The new interface also allows for multiple graph windows to be on the screen (so that one can, say, compare plots of variously generated time series). V9 also allows for more clever use of the review and variables window, which has also proved useful when one has the limited real estate of a typical laptop. There are dozens of new such features, some of which are likely to make some cranky user at least a bit happier. And the happy user of V8 can simply ignore all these improvements.

V9 continues the long standing Stata tradition that each new version requires at least 20% more manual pages. By my count V9 comes with 15 manuals and takes up about a foot and a half of shelf space. Some day Stata may come up with the stunning innovation of providing documentation on a CD-ROM, or even downloadable pdf's on their web site. As in previous versions, the on-line help system is pretty good when one is on the road, but it is hard for me not to be annoyed when searching for the Reference A-J volume which is hidden under all those papers on my desk. Even more annoying is when I cannot find the Quick Reference and Index volume to even figure out which is the appropriate manual I will not be able to find. More seriously, the potential user of the new hierarchical routines who failed to buy the XT manual, is going to have trouble understanding those routines; similarly, the graduate student who has only the basic documentation is not going to be able to fully take advantage of Stata.

Turning to the stuff we actually use Stata for, V9 has a variety of new modules that will excite various portions of the discipline, a new matrix language (Mata) that is *potentially* very exciting, and a wide variety of new routines, improvements and extensions to current routines and fixes to almost all parts of Stata. The "what's new" in the User's Guide takes up over 30 pages. What is irrelevant to one user is likely critical to another. But even before discussing new routines and modules, Stata should be congratulated on having made the command structure and related options much more consistent, eliminating lots of annoying messages saying something perfectly sensible is not allowed (my least favorite having been that lag operators were not allowed in procedures such as various estimation routines).

For me the really big new feature is the module for estimating mixed (hierarchical, multilevel, random coefficient) models. The reader looking for the hierarchical estimation routines should note that they come under Stata's XT rubric, which normally connotes panel/time-series–cross-section analysis. The XT rubric is because the hierarchical model has two (or higher) dimensional data, but the hierarchical model otherwise shares little of the time component in other XT routines. One downside of Stata's choice is that the hierarchical routines are in the XT manual, a manual not included with the base set of reference manuals. (It is also unfortunate that each field or subfield seems to have a different name for the same thing. Many political scientists would not think to look for mixed models, and it would have been nice if the index, at least, had a reference to both hierarchical and multilevel models.)

Given the recent issue of *Political Analysis* is devoted to multilevel models in comparative politics (Kedar and Shively, 2005), and my own interest in random coefficient models, I obviously think that this module is very important in our discipline. Putting mixed models (estimated via maximum likelihood) into a commonly used package will both make the estimation of such models in our field more

common, and eliminate the discussion of kludgy two–step estimators for such models. The `xtmixed` procedure appears very flexible (and looks like it can more or less do everything that can be done in the Pinheiro and Bates (2000) R module, though only for linear models). I will leave it to the Bayesians to decide whether an easy to use maximum likelihood routine for the estimation of mixed models meets their demands.

While of more limited interest, V9 allows for estimation of the multinomial probit model (Alvarez and Nagler, 1998). This is a difficult model to estimate (at least by maximum likelihood) and Stata appears to have taken advantage of recent computational advances to do a good job of estimating this model (though the data do not always cooperate). As with the hierarchical model, it now becomes feasible to add multinomial probit to a standard course and have the students actually do data exercises using that model. While only a small percentage of political scientists will ever want to estimate a multinomial probit model, that small percentage will be very grateful to Stata.

An open question is whether a general purpose package such as Stata can ever support a very complicated model such as multinomial probit as well as a more specialized package such as LIMDEP. At first glance it appears that the setup of LIMDEP may be somewhat more flexible. Though of course that is irrelevant for most political scientists, who never will see LIMDEP. But, for example, the analyst who wants to used the mixed logit (Glasgow, 2001) instead of multinomial probit will have to use LIMDEP (or GAUSS).

Multinomial probit is sexy, and Stata gives it pride of place in marketing the new version, but new, though less sexy, routines may be much important to the working life of many political scientists who estimate models with limited or qualitative dependent variable. Thus Stata can now estimate limited dependent variables with endogenous regressors. This is done by full maximum likelihood, eliminating the kludges of iv estimation based on linearization. A method previously unknown to me, stereotype logistic regression, which leans less heavily on getting the correct ordering of the dependent variable than does ordered logit, may also prove to be of great value in day–to–day research. Others will be very happy to see the implementation of various zero-inflated count models.

Another new "module" (big enough for its own manual) is less mainstream for TPM readers: measurement models, including multidimensional scaling and various flavors of factor analysis. While these seem to have gone out of fashion, there are many times when we want to construct a scale that goes beyond simple summation (and we might want to teach our students how to do this). While serious consideration of measurement in our discipline still remains a dream, having this module in Stata must be a good thing.

There are many other new routines and features in V9. Thus the survey analysis routines have been greatly extended. Someone other than me will have to discuss how useful those are. More relevant to me is that it is now easy to produce jackknifed or bootstrapped standard errors for all statistical routines, and it also appears to be a good deal easier for the user to bootstrap (or jackknife) whatever they want to. The non-linear least squares routine has also been rewritten so that one simply types, for, say, a linear model with a Box-Cox transformed variable

```
nl (y={b0} + {b1}*x1 + {b2}*(((x2^{b3=1}-1)/{b3}))*(b3 != 0)
    +ln (x2)*(b3==0))
```

(where we set the initial guess for $b3$ to 1). This is much simpler than the implementation in previous versions of Stata. Until now, when faced with a non-linear problem, I switched from Stata to some other program; I can now happily stick with Stata for non-linear least squares.

In the new consistency implemented in V9, (almost) all statistical routines (including nl and maximum likelihood) can now produce any of the various kinds of standard errors for which Stata is famous, including standard errors of marginal effects and other quantities of interest (via linearization, not simulation). A downside of Stata's dominance in this area, for which we can blame ourselves and not Stata, is that it has become unfortunately common for researchers to simply report that they used "robust" standard errors; `‘‘, robust’’` is a subcommand, not a method. While it is good that we will now see more bootstrapped standard errors, will they be used any more knowledgeably than we currently use variants of the Huber based standard errors? Stata is quite good at describing what it is doing; one wishes researchers could be so clear.

The other big change in Stata is a new matrix language, Mata. Unlike Stata's older (and still existing) matrix language, Mata feels like a fast C-like matrix language. While obviously there are a few differences in notation and operators, Mata feels a lot like R (and GAUSS). I have not done any large simulations in Stata, but looping over some big matrix inversions did show it to be fast. I have always avoided the older Stata matrix language, finding it cluncky. Mata lets one work naturally, and has all the appropriate indexing, looping and such that make R and GAUSS powerful. To give but one (critical) example, OLS is now one easy–to–read line

```
b = invsym(X'X)*(X'y)
```

which is similar to what one would do in R. This is a great addition to Stata. Instructors in the first graduate regression course no longer have an excuse not to have students generate all standard regression results using only standard matrix commands. (Stata makes it easy to import real data into these matrices, so that one can then compare the re-

sult of the matrix operations with the output of the canned regression routine.)

But either no good deed goes unpunished, or Stata has dropped the ball well before crossing the goal line. While it is perfectly nice to be able to reproduce regression commands in matrix form, and while the Mata language has a nice structure for doing simulations and such (to see the sampling properties of the estimators), Mata (in V9) lacks a maximizer (the equivalent of R's `optim` command). My students can nicely write any code they like to do any maximum likelihood routine, but, alas, they cannot actually then use this code to find the maximum and relevant derivatives. Thus, for my purposes, Mata is useless for both teaching the maximum likelihood course. Of course one can combine Mata with Stata's `ml` routine to actually produce maximum likelihood estimators, but this does not make pedagogic sense to me. I am hopeful that V10 will remedy this problem.

My bottom line is that this is a major upgrade to Stata, the biggest since the survival and longitudinal modules were put in V6 (though time-series analysts may disagree). The new hierarchical model is a really big thing, and I wish that in our discipline the measurement model would be considered a big thing. Clearly any general lab will want to upgrade to V9. I would think that any researcher who uses Stata as their primary package would also want to upgrade; there are so many new features of V9 that almost everyone (even those who simply use Stata to run regressions) will find some big improvement in V9 that is worth the upgrade price. Given Stata's very generous "grad plan," a new copy of Intercooled Stata Version 9 is about $100. (Hard copy documentation is additional. All pricing details are on Stata's very clear web site, `www.stata.com`.) Clearly existing Stata users will be even happier with V9. Users of other packages may find that the hierarchical or multivariate modules may cause them to now prefer Stata. I have always though that Stata was a great statistical package; it is now considerably better. Obviously there are some specialized packages that implement their specialities better than does Stata. Hence I still prefer EVIEWS to Stata for some time-series estimations and LIMDEP still does a few things that Stata does not do. This said, V9 has dramatically cut whatever advantage other packages may have had, while still keeping all the advantages of Stata.

For readers of TPM, the only real competition for Stata as a general package is R. As packages, each does the vast majority of things that the vast majority of users actually do. At the margins, each shows its heritage, so that Stata does more econometric and "social science things" while R is better at implementing what statisticians do. Each has a different look and feel, so that R users are very unlikely to migrate to Stata and vice versa. Stata's command structure probably makes it easier to do the things that Stata implements directly, but at the cost that going beyond what Stata has implemented is more difficult (but far from rocket science). R's object oriented approach integrates the flexibility in directly (so there is no distinction between "commands" and matrix operations), at a cost that some simple things are *slightly* harder to do. Thus, even when I use R, I prepare the data in Stata. In the end the argument about R vs. Stata is a bit like arguing about which three star restaurant is better. Both are excellent (better than anyone could ever rightfully expect), preferences between them are most likely idiosyncratic, and in the end the right thing to do is to eat at both, with the only interesting question being "which one tonight?" While one might think that one might prefer the free three star restaurant, the pecuniary costs of both Stata and R are at best a small portion of the real costs of using either (and Stata, while not free, is hardly expensive).

A more interesting issue for me is whether we should use Stata or R in our graduate methods courses. I have always found that students have an easier time in Stata, allowing them to concentrate on the methods and not the package. The advantage of R was that students could code all the routines and convince themselves that nothing being done was magic (and that they could anything they needed themselves). Thus in the linear models class I like to have students reproduce all results using matrix calculations, and in the maximum likelihood class I like to have them write their own maximum likelihood routines. While either of these could be done in Stata, I never found that either the programming or matrix language in V8 made it easy for students to understand the structure of estimation. Mata in V9 changes that for linear models; students using Stata can now use Mata to easily see what is going on in any linear routine. But, as noted above, the lack of an optimizer in Mata means that the same cannot be done for maximum likelihood routines. Thus I will continue to use a combination of Stata and R in my maximum likelihood class. The first part of the class, on general maximum likelihood estimation will use R; the second part, applications in such areas as limited dependent variables and event history, will use Stata. I hope by V10 that I can totally switch the class to Stata.

I stress that the lack of an optimizer in Mata is not quite "but other than that, how was the play, Mrs. Lincoln?" Version 9 is a major upgrade to Stata. Users of V8 will find more than enough in V9 to make the upgrade worthwhile; some will find the upgrade enormous, while others will just find it very good. And it is nice to know that V9 should not make anyone less happy.

References

Alvarez, R. Michael and Jonathan Nagler. 1998. When Politics and Models Collide: Estimating Models of

Multiparty Elections. *American Journal of Political Science* 42: 55-96.

Glasgow, Garrett. 2001. Mixed Logit Models for Multiparty Elections. *Political Analysis* 9: 116-36.

Kedar, Orit and Phillip Shively. 2005. Introduction to the Special Issue. *Political Analysis* 13: 1-4.

Pinheiro, José C. and Douglas M. Bates. 2000. *Mixed Effects Models in S and S-Plus*. New York: Springer.

Quantian: A Comprehensive Statistical Computing Environment

Christopher N. Lawrence
Duke University & Debian Project
lawrencc@debian.org

Dirk Eddelbuettel
Debian Project
edd@debian.org

While political methodologists and other quantitative social scientists are the most obvious beneficiaries of the widespread availability of personal computers and associated software, virtually all social scientists rely on computers for a variety of research-related tasks, including but not limited to empirical analyses; word processing and typesetting; communicating and publishing via email and the web; electronic library and publication searches; and various administrative tasks.

In this article, we present a comprehensive computing environment—Quantian[1] (Eddelbuettel 2003)—that we believe fills these needs at a fraction of the monetary cost of commercial software, while not sacrificing the capabilities and ease-of-use associated with such packages. Unlike most commercial software products, Quantian is "free" / "open source" software,[2] which leads to a number of advantages over commercial alternatives, particularly in an academic setting:

Academic freedom. Unlike traditional software, free software allows you to build on others' work by extending and redistributing the results to anyone you choose, without further permission from the original author. For example, the MASS package for the R statistical language and environment (R Development Core Team 2005) includes a function named `polr` (*p*roportional-*o*dds *l*ogistic *r*egression) that estimates the ordered logit model, but until recently there were no functions included in R's standard library collection which estimated the closely-related ordered probit model. With "closed-source" software, the researcher would have to write his or her own ordered probit routine to be allowed to share it with others. By contrast, because MASS is "open source," anyone is free to modify the code of `polr` to estimate the ordered probit model as well, and give that code to anyone else, so long as they give the original authors of the code their due credit.[3] Some free software licenses, most notably the GNU General Public License that R uses, also require the distribution of the source for derived works.

Developed by actual users. Most free software is developed by individuals or groups who use the software in their daily work. R, for example, is developed primarily by a core team of well-known statisticians, further extended by researchers and methodologists from different disciplines, including social sciences, and employed in a variety of cutting-edge research projects. Two prominent examples are the BioConductor project for the analysis and comprehension of genomic data, and the Virtual Data Center software developed at Harvard for sharing data over the Internet which both use R as the underlying computational engine.

Low or zero cost. As the name implies, free software is just that—free. While distributors of free software are allowed to charge as much or as little as they like for it (often in exchange for some end-user support or making the software easy to install and use), the same software can often be found on the Internet at zero cost. The low cost of free software, however, doesn't come at the expense of quality; in fact, some free software packages, like TEX and R, are more robust and have

[1] `http://dirk.eddelbuettel.com/quantian.html`

[2] Free software advocates distinguish two aspects of freedom: price ("free as in beer") and liberty of reuse ("free as in speech"). Free software is always "free" in the political sense; as explained below, it is often "free" in the economic sense of the term as well. This confusion in the English language has led to the advocacy by some of the term "open source" as a replacement.

[3] In recent months, the authors of the MASS package added several other link functions to `polr`, including probit. The optional MCMCpack package also includes an estimator for the ordered probit model.

[4] For example, McCullouch (2003) compared the accuracy of the free spreadsheet Gnumeric (part of the GNOME desktop environment) and

more features than their commercial "closed source" alternatives.[4]

These advantages make free software an ideal platform for all social scientists—and in particular for graduate students in the field and teaching methodology in a lab setting. A department could establish a methods lab by recycling discarded computers using free software at a minimal cost, instead of licensing proprietary software.

Until recently, however, such an endeavor would be difficult for an individual department to accomplish. Gathering the required software and installing it on a lab of computers would be a time-consuming process, and would require ongoing maintenance by someone with a strong background in computing—expertise that is not available at many smaller departments and often dependent on an intake of graduate students or junior faculty.

Today, however, there are several options available. A common approach in the recent years has been to gather all of the software on a freely-distributable CD-ROM or DVD, which can be used to install it on computers that will be used as lab machines. While this approach solves the problem of collecting and distributing the software, it still imposes an installation and maintenance burden.

We instead propose the use of a relatively new technology—the "live" DVD-based operating system—to solve the installation and maintenance problem. This operating system, which is called Quantian, builds on several (related) open source operating systems: clusterKnoppix, a CD-ROM-based operating system for distributed computing (itself based on Knoppix, a CD-ROM/DVD-based operating system created by Klaus Knopper), Debian GNU/Linux, a non-commercial operating system based on the Linux kernel that is the product of a worldwide network of software developers, and, of course, all the individual applications available in Debian, Knoppix and clusterKnoppix that are included in Quantian.

Quantian, like Knoppix, is designed to operate on a PC-compatible computer[5] without installing anything on the computer's hard drive. This approach was common in the days of floppy disks, but it was generally abandoned as computer hard drives became larger, faster than floppy disks, and less expensive. The increased speed of CD-ROM (and now DVD) drives, coupled with the widespread availability of writable CDs and DVDs have made removable-storage-based operating systems like Knoppix and Quantian appealing once again. Instead of needing a lengthy and complex installation process, these operating systems run directly from a CD or DVD drive, providing a complete operating system within a couple of minutes of powering on the computer. This reduces the computing expertise needed to use Quantian to a minimum, as well as allowing potential users to test drive Quantian without going through an installation process. This approach also allows the use of Quantian on borrowed computers, such as loaned machines or in public-use computer labs.

Quantian also builds on the free software packages produced by the Debian project. Debian developers have packaged over 10,000 pieces of free software, including a wide variety of packages for statistical and numerical computing. Quantian takes many of the scientific and mathematical tools from Debian's collection of software, as well as a full desktop environment, providing a rather complete set of free software for statistical computing.[6]

At present, Quantian adds about four gigabytes of scientific software to a basic Knoppix system[7] including a complete TEX and LATEX installation, based on the teTEX distribution, as well as additional style files and utilities; the R statistical computing environment, along with an almost complete selection of R packages from both the Comprehensive R Archive Network (including MCMCpack and Zelig, two packages developed by political methodologists) and from BioConductor; Octave, a numerical computing system similar to Matlab; a number of computer algebra systems; Gnuplot and several other scientific plotting engines; the OpenDX and Mayavi data visualization systems; and the XEmacs editing suite, including the popular AucTEXand ESS (Emacs Speaks Statistics) packages for editing TEXand interacting with statistical software, respectively.[8] Many of these software packages will be familiar to readers of *The Political Methodologist*.

For programmers, Quantian also includes a complete software development environment based on the GNU Compiler Collection (GCC) for programming in C, C++, Fortran, and Java; additionally, the Perl, Python (including the Numeric and Scientific Python modules), Ruby, Lua and Tcl scripting languages are available for less computationally-intensive tasks.[9]

While the Quantian environment is based on Linux,

Microsoft's Excel spreadsheet (part of Microsoft Office), and revealed serious numerical inaccuracies in several functions included in earlier versions of both products. However, Gnumeric's problems were rectified in later versions, while many problems in Excel were either not fixed or given incomplete fixes that led to other inaccuracies in Excel 2003.

[5]Virtually all live CD-ROM/DVD projects target x86-compatible PCs, with a handful of exceptions for Mac and 64-bit PC platforms.

[6]An overview of Debian's thousands of packages is available: `http://packages.debian.org/unstable/`; of particular interest will be packages in the math and science categories: `http://packages.debian.org/unstable/math/`, `http://packages.debian.org/unstable/science/`.

[7]This is measured relative to the CD-ROM-based Knoppix 3.* versions which measure around 2.2 GB uncompressed versus about 6.6 GB for Quantian. However, the recently released Knoppix 4.0 is also DVD-based and contains several applications also found in Quantian.

[8]Veterans of UNIX will also appreciate the inclusion of Vim, a clone of the venerable *vi* editor.

[9]A full listing of included software is available at the Quantian web site: `http://dirk.eddelbuettel.com/quantian.html`.

within Quantian users can access files on USB-memory devices, floppy, Zip or hard drives connected to the computer, as well as networked resources on Windows and Macintosh networks and on the Internet; there is also support for printing to local and network printers. This versatility allows users to load and save data sets to persistent storage, even though the operating system is only transitory.[10]

Quantian, in effect, combines the ease of use of an installed computer system like Windows or Mac OS X with the convenience of having a complete statistical environment that can be used on almost any PC-compatible computer. We envision several use cases for Quantian by political scientists:

Computer labs. Undergraduate and graduate quantitative methods courses often involve lab sessions. The preparation of computer labs is a time-consuming task, made even more difficult if the computers involved are under the jurisdiction of another department or an academic computing office. Quantian solves this problem by allowing the temporary use of a statistical computing environment; all the instructor needs to do is produce one CD (or DVD) for each computer, and when the class begins, each computer can be rebooted into the Quantian environment. When the class is over, the students only need to remove the CDs and reboot the computers back to their "normal" operating system.[11]

Homework. Undergraduate and graduate methods courses also require students to complete homework assignments. With Quantian, students can use the exact environment they use in class for out-of-class assignments on their own personal computers, with no licensing restrictions or lengthy installation process; all the students need is a CD-ROM/DVD containing the Quantian system.

Ad-hoc computing clusters. Quantian includes the openMosix system for parallel computing. This will allow the use of Quantian for ad-hoc computing clusters, which can dramatically increase the speed of computationally-intensive estimation procedures such as numerical integration, Markov chain Monte Carlo (MCMC), multiple imputation, and bootstrapping. With Quantian, a university computing lab could be put to productive use over a weekend to solve computing problems, without disrupting its use during the week for instruction or other purposes and without permanently installing any software in the lab.

Bare or "hand-me-down" machines. Old computers, before they are removed from service or salvaged, often have their entire hard drive wiped clean. Legally, inheritors of these PCs must purchase a new copy of a commercial operating system like Windows to have the right to use it on these machines. Quantian, as a truly free operating system, has no such legal entanglements, and thus is an ideal replacement for Windows on salvaged hardware.[12]

Travel. Faculty and graduate students often travel to conferences or other campuses to conduct and present research. Quantian, as a complete statistical system on one CD-ROM or DVD, can be taken anywhere and used on virtually any PC-compatible computer with a CD-ROM or DVD drive made in the past decade—in hotel business centers, on loaned laptops, and in computer labs at other institutions—without having to travel with a computer.

Saving space. One of the apparent laws of computing is that eventually data will expand to fill all available disk space. Having your operating system on a CD or DVD will free space on your computer for email, data sets, papers, letters and other personal information that needs to be stored.

Virus protection. Using an operating system like Quantian, where the operating system itself is on a non-writable storage medium, will reduce the ability of a virus to infect important system files and leave your system in an unusable state.[13]

Additional information on Quantian, including new releases, is available at the Quantian web site, `http://dirk.eddelbuettel.com/quantian.html`. You can obtain a copy of Quantian on CD or DVD for a small fee from a number of distributors, or produce your own copy for free using a personal computer with a fast Internet connection and a CD or DVD writer. The website also has instructions on how customized variants of Quantian can be created.

We look forward to feedback from users of Quantian, including suggestions for additional software to be included

[10] However, Knoppix-based systems such as Quantian also allow for installation to the hard drive, and are frequently used as a means to installing a Debian-based system. Moreover, a USB-memory device can be used as a 'portable' home directory in which a user can store configuration, application data, or even extensions such as R packages beyond the hundreds already provided by the base system.

[11] Many newer computers have support for network booting using the Intel PXE protocol; this will allow all of the machines in a lab to be booted from a single master computer. For details on how to use this procedure, see `http://dirk.eddelbuettel.com/quantian/howto_netboot.html`.

[12] This is of course true for other free Linux distributions, but not necessarily for many commercial Linux distributions, such as Red Hat Enterprise Linux and Novell's SUSE Linux, which have per-seat licenses.

[13] On the other hand, the read-only nature of the storage medium also implies that security updates cannot be applied. The user will have to wait for (or create) a new release of the CD/DVD with the updated software.

and reports on its usage by readers of *The Political Methodologist*. We also plan to provide reports on some of the above use cases at future academic conferences and to continue evangelizing the use of Quantian and free software by both political methodologists and the broader scientific community.

References

Dirk Eddelbuettel. 2003. "Quantian: A scientific computing environment." In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing* (DSC 2003). URL http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/Eddelbuettel.pdf. ISSN 1609-395X.

Bruce D. McCullouch. 2003. "Fixing statistical errors in spreadsheet software: The cases of gnumeric and excel." Technical report, *Computational Statistics & Data Analysis Statistical Software Newsletter*. URL http://www.csdassn.org/software_reports/gnumeric.pdf.

R Development Core Team. 2005. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005. URL http://www.R-project.org. ISBN 3-900051-07-0.

Data Entry

Using XEmacs Macros to Process ASCII Data Files

Michael C. Herron
Dartmouth College
Michael.C.Herron@Dartmouth.edu

Introduction

Most quantitative researchers will at one point or another be forced to deal with ASCII data files that on account of formatting cannot be read directly into statistical packages. A key task for someone who comes across such an unwieldy file is to coerce it into a format that can be easily digested by a spreadsheet or a statistical package. This task can be very time-consuming if the format of the file does not lend itself to easy manipulations.

A simple and portable format for ASCII data files is comma-delimited. Comma-delimited files are easily read into R with the command `read.csv` and Stata with the command `insheet`. In comma-delimited files data fields are separated by commas; another portable format is tab-delimited, but separation by commas should be considered preferable because commas are inherently visible unlike tabs. Thus, unwieldy file coercion typically involves manipulating badly formatted ASCII data lines so that each line contains a logical set of entries, each line has commas in the right places, and there are no commas in wrong places.

It is important to note that comma-delimited files have their own limitations. In particular, they are problematic if data fields themselves include commas. For instance, suppose that a data field of interest is "last name,first name." In this case, the comma between last name and first name does not separate fields.

An alternative to delimiting fields by commas is to delimit them by pipes ("—") instead. Since pipe characters rarely are used in data files this is usually a safe approach.[1] The bottom line here is that one needs to be careful when choosing a delimiter. This article uses commas but there is nothing that prevents one from modifying what is below to use pipes, number signs, etc.

A common source of unwieldy ASCII files is the PDF document format. PDF files are easily portable (indeed, the "P" in PDF stands for portable) insofar as they can be viewed by publicly available software that runs in all modern operating systems. It is not uncommon for government agencies to publish data in PDF format.

However, despite their benefits, PDF files cannot be read directly into statistical packages. While they often can be converted to ASCII (see below for comments on this), PDF to ASCII conversion often results in a somewhat messy collection of characters and numbers.

For example, consider a set of DuPage (Illinois) County election returns from the 2004 general election.[2]

[1] Both R and Stata allow users to specify field delimiters for reading ASCII files, so there is no risk in a user choosing a delimiter that best suits a given file.

[2] DuPage County election data can be found at http://cms.dupageelections.com. The URL for the file discussed here is http://eh.robis.net/uploads/JID37_2004_11_results_Addison.pdf.

DuPage County is divided into a number of townships, one of which is Addison (other townships are similar and the use of Addison here is without loss of generality). Precinct returns for Addison Township are available from DuPage County in PDF format only. Figure 1 shows presidential elections results the Addison PDF file after it has been run through the standard Unix utility `pdftotext`, version 3.00, using the preserve layout option (other elections, i.e., Illinois U.S. Senator are similar).[3]

To make Figure 1 clear, Addison Township precinct 30001 produced 253 votes for Bush, 395 for Kerry, six for Badnarik, and so forth; precinct 30002 produced 142 votes for Bush, 160 for Kerry, zero for Badnarik, and so forth; and precinct 30015 produced 287 for Bush, 278 for Kerry, four for Badnarik, and so forth.

The fields in the vote return lines (e.g., "253 395 6...") are not separated by commas, there are nuisance lines (e.g., "REP") between vote return lines, and the data in the lines needs to be transposed so that each row is a precinct rather than a candidate. In other words, converted DuPage data needs pre-processing before it can be read by statistical software.

PDF files that consist of scanned images cannot in general be converted to ASCII with a utility like `pdftotext`. When faced with an image embedded in a PDF file one approach is to use optical character recognition (OCR) software to convert the image to characters. OCR software, if it is able to recognize the characters in the PDF image, can in theory produce ASCII text akin to what `pdftotext` produces.

One way to coerce unwieldy ASCII data files into usable formats is with a text editor like XEmacs. XEmacs, unlike WYSIWYG word processors like Microsoft Word and Apple's Pages, is designed to process ASCII text. XEmacs is open source software, it runs under Linux, Macintosh OS X, and Windows, and it can be downloaded from `http://www.xemacs.org`, among many other places. Moreover, XEmacs can be easily programmed, and in this brief article I explain how user-designed XEmacs keyboard macros can be created to deal with data files that, for lack of a better description, look like a big mess.

I will assume throughout this article that readers have a basic understanding of how XEmacs works and are running XEmacs in a windowing environment like X11.[4] For those who are not familiar with XEmacs, documentation can be found at `http://www.xemacs.org/Documentation`. XEmacs is very similar to the veritable Emacs editor; command keystrokes sometimes differ between the two, but the macro functionality I describe here is also available in Emacs.

Modern text editors are very powerful tools that can be harnessed for data processing, not to mention programming, document creation, and many other tasks. Beyond XEmacs and Emacs, Windows users have access to WinEDT,[5] and OS X users can use the freely-available TextWrangler or its commercial cousin BBEdit.[6]

The Basics of XEmacs Macros

An XEmacs macro is useful when a command sequence needs to be repeated, either in the preparation of a single file or across multiple files. For instance, suppose that one has a list of names as follows:

John,Smith5
Richard,Roe4
Jane,Doe7

where the fields above are first name and last name elided with a single-digit number.

If a list had hundreds of names like the ones above, it would not be efficient to delete each trailing number by hand. Nonetheless, note the following. From the beginning of each line, the XEmacs command "`C-e`" (meaning Control-e) moves the cursor to the end of the line and then a single backspace removes the offending number. So, if one could program a macro to start from the beginning of a line, move to the end, delete one character, and then shift to the next line, the problem of getting rid of trailing numbers would be simple.

In general, XEmacs macro definition works as follows. One tells XEmacs to start definition of a macro; one then types the actual key sequence desired for the macro; finally, one tells XEmacs that macro definition is complete.

The way to create a macro for the deletion of trailing digits is as follows. First, move one's cursor to the top line of the file (although, as it will be clear, this is not necessary). Second, type "`C-x (`" (meaning `Control-x` followed by a left parenthesis) to begin defining a macro. Third, carry out the sequence of commands necessary to parse a single line. That is, type "`C-e [BACKSPACE] [DOWNARROW] C-a`". Fourth, type "`C-x )`" to complete the macro definition.

Note the presence of "`[DOWNARROW] C-a`" at the end of the macro. This ensures that the macro finishes its operation at the beginning of a line, i.e., the same place it started. This is essential because repeated macro calls will not work as intended if each macro call ends in a difference place structurally compared to where it started.

[3] The syntax for this is `pdftotext -layout Addison.pdf`, which will produce Addison.txt.

[4] Running XEmacs in a terminal is certainly feasible but this has its own set of problems, one of which is that the interpretation of Control-s can get complicated.

[5] Shareware available from `http://www.winedt.com`

[6] Text Wrangler and BBEdit are published by Bare Bones Software at `http://www.barebones.com`.

```
DUPAGE COUNTY, ILLINOIS
GENERAL ELECTION NOVEMBER 2, 2004
TOWNSHIP Precinct data Canvass A
OFFICIAL RESULTS
30001 to 30015
ADDISON Precincts
PRESIDENT AND VICE-PRESIDENT OF THE UNITED STATES
30001 30002 30003 30004 30005 30006 30007 30008 30009 30010 30011 30012 30013 30014 30015
CANDIDATE
253 142 432 289 180 268 130 276 170 141 227 214 254 163 287
George W. Bush
REP
395 160 265 215 150 318 188 263 200 233 246 281 184 265 278
John F. Kerry
DEM
6 0 9 9 3 2 2 3 0 5 2 4 2 6 4
Michael Badnarik
LIB
```

Figure 1: Presidential election results from the Addison PDF file converted using `pdftotext`

If you make a mistake while entering in a macro, just type "`C-g`" and restart. You must completely restart when breaking a macro definition this way, i.e., "`C-x (`" to start macro definition is necessary.

At this point XEmacs has a defined macro that, when called, deletes the last character in a line and skips to the next line. Then, one need only type "`M-x call-last-kbd-macro`" to call the macro (recall that `M-x` means Meta followed by x; in most cases this is equivalent to `Escape-x`). This is an awkward command to type repeatedly, so place the following text in an XEmacs initialization file: "`(global-set-key 'f11 'call-last-kbd-macro).`" This defines the F11 function key to call whatever macro XEmacs has in memory; F11 can then be pressed as many times as needed. If one had a file with, say, 500 lines that needed to be parsed in accordance with this macro, one could type "`C-u 500 F11`" which tells XEmacs to call F11 500 times, i.e., call the current macro in memory 500 times. Note that "`C-u`" is a generic command for calling a command repeatedly. For example,
"`C-u 200 ,`" will generate 200 commas.

As an aside, another useful definition for one's XEmacs initialization file is a shortcut for search and replace:

```
;;; set C-xrp as replace-string
(global-set-key "\C-xrp" 'replace-string)
```

Henceforth I will assume that "`C-x r p`" is defined this way.

What if, rather than deleting the trailing number in a line, one wanted to make the number a separate field, i.e., change "John,Smith5" to "John,Smith,5" and so on? From the beginning of a line, here is the macro sequence that would accomplish this:"`C-x ( C-e [LEFTARROW] , [DOWNARROW] C-a C-x ).`" The only difference between this sequence of commands and the previous sequence is that in the latter the addition of a comma has replaced deletion.

Note that the XEmacs command `call-last-kbd-macro` calls only the last macro defined and not, say, the macro defined before a second macro was defined. Below I discuss how to name macros and save them so that new macros do not displace old ones.

If one makes a error when defining a macro, there are two ways to handle it. One, as noted above, is to press "`C-g`" which is the XEmacs way of stopping something, in this case macro definition. If macro definition is stopped with "`C-g`" then no macro is defined and, if one had been in memory, it is not deleted. Another way is to simply continue the macro definition in a way that corrects for the mistake. This second approach is not always possible. Nonetheless, consider the following example.

Suppose that, as in the above example, one intended to type "`C-x ( C-e [BACKSPACE] [DOWNARROW] C-a C-x )`" but accidentally pressed "`z`" after entering "`C-e`" to move the cursor to the end of a line. In this simple case, one need only delete the "`z`" with a back space which will remove the error and then continue with the macro definition. In this case, the macro will have been defined by "`C-x ( C-e z [BACKSPACE] [BACKSPACE] [DOWNARROW] C-a C-x ),`" which will work nicely. The first back space deletes the accidental "`z`" and the second back space deletes the comma as originally designed.

Multiple editing tasks can be defined in a single macro. Suppose that one wanted to coerce "John3,Smith5" to "John,3,Smith,5" and so on across number lines. A macro to do what is needed is as

follows: "C-x (C-s , [LEFTARROW] [LEFTARROW] , C-e [LEFTARROW] , DOWNARROW C-a C-x)." Note the presence of "C-s ," which makes XEmacs search for the first comma after the cursor. This is useful because the number of spaces before the first comma may not constant across lines, i.e., one may want to convert "John3,Smith5" as well as "Meredith5,Smith1."

Similarly, suppose that one wanted to change lines that read as follows:

john,smith
richard,roe
Jane,doe

so that the first letters of first and last names were capitalized.

The trick to this operation is noticing that the second letter to be capitalized in each line follows the first (and only) comma in the line. So, the way to write a macro to do what is needed is as follows: "C-x (C-[SPACE] [RIGHTARROW] M-u C-s , [RETURN] C-[SPACE] [RIGHTARROW] M-u [DOWNARROW] C-a."

This works as follows. "C-x (" as we have seen before starts macro definition, assuming that the cursor is on the first character of the line. Then, "C-[SPACE] RIGHTARROW" highlights the character under the cursor and "M-u" changes this character to uppercase. Next, "C-s , [RETURN]" searches for the next comma and then ends the search (this is the purpose of "RETURN" after the comma). Finally, "C-[SPACE] [RETURN] M-u" works as before, and "DOWNARROW C-a" returns to the first character of the next line.

The point here is that XEmacs macros are very good at tasks like repeated character insertion, deletion, and modification. The trick necessary for defining a macro is figuring out what sequence of insertions, deletions, and modifications will change a messy data line or lines into something that is useful.

Naming and Saving Keyboard Macros

After generating a keyboard macro one might want to save it for use later. Some macros are highly idiosyncratic to a given file and may not merit saving. But, others can be very useful, i.e., a macro that capitalizes certain fields in a line.

Saving an XEmacs macro is a two-step process. First, one must name the macro with the command "M-x name-last-kbd-macro" to which one supplies a name. This command will give a name to the last keyboard macro defined. Then, to save this definition in a file so that it can be read later, one navigates to the file of interest (i.e., "C-x C-f /.xemacs/mymacros.el") and types "M-x insert-kbd-macro [RETURN] NAMEOFMACRO [RETURN]" where "NAMEOFMACRO" is the name of the macro defined in the first step of this procedure. At this point XEmacs will insert Lisp code for the macro at the cursor, which is somewhere in, possibly the end of, the "mymacros.el" file (or wherever one wants). One need not know any Lisp syntax to use the macro! Nonetheless, after inserting the macro code save the file with the standard XEmacs file save command ("C-x C-s") and resume work.

To load the macro definitions in a file type "M-x load-file /.xemacs/mymacros.el" assuming that the file to be loaded is named "mymacros.el" and it is in a naturally placed ".xemacs" directory. The "load-file" command in XEmacs is akin to sourcing a file.

Multiple macro files can be sourced during an XEmacs session, and macros can be deleted from files with careful editing. The Lisp syntax for a macro is self-contained as a perusal of a file that contains macro definitions will show. For instance, the Lisp code for the first macro discussed here "C-x (C-e [LEFTARROW] , [DOWNARROW] C-a C-x)" is as follows (the macro was named "fixit"):

```
(defalias 'fixit
   (read-kbd-macro "C-e <left> , <down> C-a"))
```

Macros for DuPage Data

I will now provide a few macros that can be used to process the DuPage data discussed in the introduction. Those readers interested in learning how these macros work should download the complete DuPage PDF file, convert it to ASCII, and then follow along. Note that the DuPage file has many sections that mirror the excerpt above.

The first thing I would do when faced with something like the DuPage file is use macros to remove unnecessary lines. For example, any line with the words "GENERAL ELECTION" in it needs to be removed (again, there are many such lines even though the excerpt above only contains one). A macro to do this is as follows: "C-x (C-s GENERAL ELECTION [RETURN] C-a C-k C-k C-x)." This macro works by searching ("C-s") for the words "GENERAL ELECTION," moving to the beginning of the line in which "GENERAL ELECTION" was found ("C-a"), and then deleting the line ("C-k C-k"). It is not necessary in this case to end the macro with a second "C-a" since the cursor, when the macro completes, will already be in the beginning of a line. To use the macro move the cursor to the beginning of the document ("ESC <") and then press F11 assuming that one has defined F11 appropriately. One can hold F11 repeatedly until XEmacs reports that "GENERAL ELECTION" cannot be found.

I would then repeat this process with the phrases "DUPAGE COUNTY," "Precinct data," "OFFICIAL RESULTS," "PRESIDENT," and "PRECINCTS." I would also do the same with the string of dashes "------" to

get rid of lines that contain only dash dividers. I would also delete lines based on the word "to" but be warned that deleting lines based on short words or fragments can be risky. In this case I would ensure that my macro deletes based on " to " as opposed to deleting based on "to" without surrounding spaces: "`C-x ( C-s [SPACE] to [SPACE] [RETURN] C-a C-k C-k C-x ).`"

Finally, we need to insert commas between vote return fields. I would do this by defining a macro as follows: "`C-x ( C-s Bush [RETURN] C-a [UPARROW] C-[RIGHTARROW] , C-[RIGHTARROW] , C-[RIGHTARROW] , ...C-x )`" where "`C-[RIGHTARROW] ,`" is repeated as many times as necessary. This macro works as follows. It first searches on the word "Bush" and then returns ("`C-a`") to the beginning of the line that contains this word. The macro then moves the cursor up a line ("`[UPARROW]`"). Next, "`C-[RIGHTARROW]`" calls an XEmacs command called "forward-word" that moves the cursor to the end of the word that follows the cursor. In this case, the end of the word is the space after 253, which is right where a comma is needed. Further instances of "`C-[RIGHTARROW] ,`" fill in the line with commas. After the macro is complete, it would be called with `F11`, which will deal with every line that begins with Bush. The same process can be repeated with "Kerry" substituting for Bush and so forth.

This is a brute force approach that is not very efficient: one has to write a macro for each candidate name. Another approach is as follows.

If one searches on the word "Bush," (note that Bush is the first candidate), then deleting the Bush line ("`C-k C-k`") and the following line with "REP" in it ("`C-k C-k`" again) leaves only Bush vote returns in the Bush lines, broadly defined. If one then executes "`[DOWNARROW]`" the cursor ends up in Kerry lines where four "`C-k`" commands delete Kerry and "DEM." This process can be repeated for each candidate. So, a resulting macro that deletes a number of unnecessary lines is as follows: "`C-x ( C-s Bush C-a C-u 2 C-k [DOWNARROW] C-u 2 C-k [DOWNARROW] C-u 2 C-k ...C-x ).`"

This macro works on a block of candidate returns where each block starts with the candidate Bush and then has the same number of candidates afterward. Once the macro has been executed, the next step is to write a macro to insert commas in numerous lines. This can be done with a macro calling a macro (i.e., write one macro which inserts commas in a line, name the macro, say, "putincommas" and write a second macro that calls "putincommas") or a single macro.

After candidate names are cleared out one thing remaining is a collection of lines that contain registered voter counts. These lines are for the most part redundant and easy to eliminate using a macro: "`C-x ( C-s registered C-a [UPARROW] C-u 4 C-k C-x ).`" This macro not only removes registered voter lines, but it also removes total vote numbers as well. It works by searching for the word "registered," moving the cursor to the beginning of the preceding line, and then deleting registered voter and total vote counts.

The set of macro described above of the DuPage file is not complete insofar as processing the file with them will not by itself result in a comma-delimited file that can be easily read into a statistical package. However, the macros constitute a solid step in this direction. Based on the macro-development principles elucidated above, it is straightforward to write additional macros to finish up the DuPage file.

Macros for Key Biscayne Data

Another example of a somewhat complicated ASCII data file is the excerpt from November 16, 2004 election results in Key Biscayne, Florida in Miami-Dade County shown in Figure 2.[7]

The lines in Figure 2 describe the votes cast in Precinct 49 in a village council race. The difficulty of working with data lines formatted in this way stems from a number of things: the lines for Precinct 49 contain many unnecessary entries like percentages, blank lines, and other irrelevant details; they have different numbers of fields per line; and, some have embedded quotes. Ideally one would want a comma-delimited line for Precinct 49 as follows: 49, 315, 104, 72, 197, 195, 96, 94, 0. This line contains a precinct identifier ("49") and describes precinct vote returns, i.e., 315 total ballots, 104 votes for Akerlund-Tarajano, 72 votes for Davey, and so forth.

The complete Key Biscayne file from the November 16 election lists results from four precincts. This number is small, of course, so coercing Key Biscayne data into a usable format can be done by hand in a short short amount of time. This is not a satisfactory approach in general though, because precinct data files can easily have results from hundreds of different precincts.

The first thing I would do if confronted with the Key Biscayne file would be to eliminate nuisance lines. Getting rid of the "TOTAL VOTES" should be straightforward for readers who followed the DuPage example: "`C-x ( C-s TOTAL VOTES C-a C-u 1 C-k C-x ).`" Then repeat this macro as often as necessary. One could also do this for lines that contain "(VOTE FOR) 3."

The next step (or, a next step insofar as one can approach data file coercion from many difference angles) is converting the candidate fields to comma-separated lines. Here is how I would begin to handle this. To get rid

[7] Source of the Key Biscayne file is `http://elections.miamidade.gov/ele111604/kb-d.pdf`.

```
0001 PRECINCT 49
TOTAL VOTES % IVOTRONICS ABSENTEE
BALLOTS CAST - TOTAL.  .  .  .  .  .  .  . 315 315 0
BALLOTS CAST - CITY:KEY BISCAYNE.  .  .  . 315 315 0

VILLAGE COUNCIL CITY:KEY BISCAYNE

(Vote for )  3
Annika Akerlund-Tarajano . . . . . 104 12.12 104 0
Michael W. Davey . . . . . 72 8.39 72 0
Enrique Garcia. . . . . 197 22.96 197 0
Steve Liedman . . . . . 195 22.73 195 0
Federico ‘‘Fredy’’ Padovan  . . . . . 96 11.19 96 0
Thomas Thornton  . . . . . 194 22.61 194 0
Over Votes . . . . . 0 0 0
Under Votes . . . . . 87 87 0
```

Figure 2: Excerpt from November 2004 election results in Key Biscayne, Florida in Miami-Dade County

of extraneous periods, first go to the beginning of the document ("`ESC <`") and then replace all instances of " . " with blanks: "`ESC < C-x r p [SPACE] . [SPACE] [RETURN] [RETURN].`" Note that importance of spaces surrounding periods: you do not want to turn 12.12 into 1212!

Next, all instances of three consecutive spaces should be turned to commas: "`ESC < C-x r p [SPACE] [SPACE] [SPACE] [RETURN] , [RETURN].`" Now, because there are doubled-up commas, create a macro to get rid of them: "`C-x ( ESC W C-x r p , , [RETURN] , [RETURN].`" This macro, when executed, will change all instances of ",," to "," and then quit. One would call this macro repeatedly until there are no doubled-up commas. Then do the same for ", " with the following: "`C-x ( ESC < C-x r p , [SPACE] [RETURN] , [RETURN] C-x )`" which can be called as often as necessary.

Aside: changing comma-space combinations to single and plain commas is a very common task in ASCII data file coercion. I suggest writing and saving for future use several macros that take these combinations and fix them; see the macro naming and saving directions above.

Finally there is the problem of trailing zeroes. These are easy to fix with a search-and-replace call that removes trailing zeroes: "`C-x r p 0 C-q C-j [RETURN] C-q C-j.`" This call is straightforward except for one feature: it uses the quote prefix "`C-q`" in a search. The reason for this is as follows. To remove trailing zeroes one has to search for zeroes that occur right before a new line starts. A new line in XEmacs is preceded by a control character, namely "`C-j.`" To tell XEmacs to search for a control character one must quote the character so that it is not executed. That is, a new line can be started in a file by pressing "`C-j.`" However, to search for zero followed by a new line character, one searches for "`0 C-q C-j`" where "`C-q`" inserts a control character.

Again, if you use XEmacs to process data files you will regularly find yourself getting rid of things like trailing zeroes and trailing commas. To automate this I recommend writing and saving several macros that do this task. Then, whenever you run into the need to remove, say, trailing commas, you can just go to the beginning of your file and type "`C-u 500 byetrailingcommas`" if you named your macro "byetrailingcommas." Calling a macro like this 500 times is guaranteed to work if your file has 500 or fewer lines. Since the macro has a search embedded in it, the macro will quit on your, and the 500 calls to the macro will also stop, if a search fails because there are no trailing commas to be found.

These macros do not completely resolve the formatting issues in the Key Biscayne data, but they address most of them. It is straightforward to deal with additional nuisance lines.

Conclusion

To summarize, XEmacs macros, augmented with search-and-replace calls, provide a very powerful way to manipulate ASCII data files. Those interested in learned how XEmacs can be harnessed to process data files should consider the following steps.

First, install XEmacs on one's own computer. For those who use linux, XEmacs is probably already installed. For Mac OS X and Windows XP users, installing XEmacs is straightforward (OS X users should install an XEmacs that runs under X11 or a Carbon version of Emacs like yaced, available at `http://yaced.sourceforge.net`.) Second, learn the basics of XEmacs editing by actively maintaining files in XEmacs. Third, begin to experiment with macros. After several months of use XEmacs macros will become second nature and all ASCII files will inevitably be looked at with an eye toward the question, "What sort of macro is appropriate for working this sort of file?"

Data Entry: Going Pro

Seth Masket
University of Denver
smasket@du.edu

Suppose you have a large quantity of data in an unusable format. It could be hard copies of campaign finance reports, printed records of roll call votes, or eighteenth century election returns. Whatever it is, it's of no real use to you unless it's inputted into a computer database, but that will require literally hundreds, if not thousands, of work hours.

You could approach this problem by hiring a team of student researchers. Or you might just sit down in front of your computer with a thermos full of coffee and prepare for a long, hard slog. Before going these routes, though, you might consider hiring a professional data entry firm.

Professional firms offer many advantages over other data entry methods. For one, they are highly accurate in their work, which is particularly helpful with large datasets, for which it is nearly impossible to check every inputted item. Professional firms are also considerably faster than most students or researchers. After all, this is what they get paid to do, while students and researchers do such data entry in their spare time. Finally, professional firms, particularly ones based outside the United States, usually offer quite reasonable rates.

A co-author and I recently hired a data entry firm in India to input more than a century of roll call votes from a state legislature's journals into a database. Through a brief e-mail chat, we were able to establish the parameters of the job and our expectations, and we easily negotiated a reasonable price that was far below those of comparable domestic firms. Despite the size of the job (more than 70,000 votes cast by over 4,000 legislators), the firm was able to meet our expectations quickly and with a very low error rate. Indeed, I am now working on a second state legislature and have hired the same firm to help me.

The fact that the vendor was based outside the U.S. proved not to be a problem. The vendor's employees were proficient in English and highly technically skilled. Shipping can be an issue when working with foreign vendors - it is not cheap to mail hundreds or thousands of photocopied pages overseas. In this case, it proved not to be an issue since the state legislature we were studying makes their journals available on the web. However, when it is an issue, there are ways to mitigate such costs. For example, many universities now have digital senders - devices that rapidly scan pages into PDF format and e-mail them as file attachments. This way, raw pages can be burned onto a CD-ROM, which is much cheaper to mail than reams of paper. The pages can also be uploaded to a server so they can be accessed by anyone in the world with a computer and an internet connection.

One way to improve the accuracy and efficiency of any data entry project is to develop a web-based interface. That is, instead of having a firm directly type your data into a spreadsheet, you can design an web-based program to simplify the data entry. The user's input can be limited to pull-down menus and radio buttons, substantially reducing the potential for error. A web-based interface has numerous other virtues, as well, in that it can be programmed to produce a dataset precisely as you want it and you can keep track of the project as it is being completed.

A web-based interface can be designed by anyone with a reasonable degree of skill with web-based programming. If you don't feel up to this task, it can be quite inexpensive to hire someone to do it for you. Students in computer science departments may be willing to take on the task for a reasonable fee or even course credit. Many professional data entry firms actually have software developers on staff who can help you with such a project. For a recent data entry project, I found a web programmer through the freelance programmer clearinghouse (for example, `http://www.rentacoder.com`). Most such project shouldn't cost much more than $100 to $200.

If you decide to hire a professional data entry firm, here are some tips before you close the deal:

Check references. Many firms have plenty of experience with business clients, but not as much with academic clients. It's helpful if they're familiar with academic researchers' needs and expectations.

Be specific about the output you're expecting. Send some sample material and even mock up some results, if necessary, to show the vendor what you need.

Check the output before it's completed. This is very easy if you have a web-based interface. If not, ask the vendor to send you some results after a dozen or so cases have been entered. Make sure they're doing it correctly.

Figure out your costs ahead of time. You'll need to pay for the data entry. You may need to pay to ship the raw data. You may need to pay for web programming and scanning. Estimate all of this ahead of time

to determine if hiring a professional firm is the best use of your research money.

The last point is not a trivial one. Costs multiply as the project grows, so it's good to know what you're getting into. If shipping costs are going to be high, for example, and if the data entry method itself is at least somewhat resistant to errors, hiring graduate or undergraduate research assistants may be the way to go.

However, the adage that you get what you pay for holds true even in academic research. Even if the costs of a professional firm seem high relative to those of student research assistants, the speed and accuracy of the work may make it worthwhile. At the very least, I'd recommend getting a quote or two from professional firms and looking at some of their previous work. Getting more research money usually isn't impossible; fixing improperly-entered data usually is.

Book Review

Review of Janet M. Box-Steffensmeier and Bradford S. Jones' Event History Modeling: A Guide for Social Scientists

Tze Kwang Teo
University of Illinois at Urbana-Champaign
tzeteo@uiuc.edu

Event History Modeling: A Guide for Social Scientists. Janet M. Box-Steffensmeier & Bradford S. Jones. Cambridge University Press, New York, 2004, 232 pages. $24.00, ISBN 0-521-54673-7 (paperback); $65.00, ISBN 0-521-83767-7 (hardcover).

Quantitative political scientists have come to embrace the advantages of analyzing longitudinal over cross-sectional data. There are myriad social and political phenomena where the time-to-event occurrence is of substantive interest, but may be censored due to a variety of (quasi-)experimental conditions. Event history/survival analysis is well-suited for modeling such phenomena, but until last year books on this technique—be they introductory, intermediate, or advanced level—were written by scholars in other fields such as biostatistics (e.g., Collett 2003; Hosmer & Lemeshow 1999) and sociology (e.g., Yamaguchi 1991). Box-Steffensmeier and Jones (B-SJ) can thus lay claim to two "firsts". *Event History Modeling* (*EHM*) is the first in Cambridge University Press' *Analytical Methods for Social Research* series, and more importantly, the first book-length introduction to survival analysis written by political scientists.

B-SJ have written *EHM* with the applied researcher in mind, thus the book's level of difficulty is similar to the above-cited introductory texts. Prior knowledge of statistical distributions and maximum likelihood estimation will be helpful, but is not required. Examples of applications in American, comparative and world politics are favored over extensive mathematical derivations and proofs, and discussed throughout the text, in order to facilitate conceptual understanding of, and familiarity with event history models and data structures. The book is less suitable for those seeking a text that teaches data analysis using commands for a particular statistical package (e.g., Blossfeld & Rohwer 2002; Cleves, Gould & Gutierrez 2004; Tableman & Kim 2004), but note that S-PLUS/Stata code and data for the key examples covered are available online.[1]

The book can be divided more or less into four parts: conceptual underpinnings and basic mathematical functions (chapters 1–2); regression models (chapters 3–5); model selection, diagnostics, and extensions for inclusion of time-varying covariates (TVCs) (chapters 6–8); and models for "complications" of unit heterogeneity or multiple events (chapters 9–10). The concluding chapter (chapter 11) covers a broad range of issues that can crop up in social science settings. I also note that the unfortunate problem of space constraints has led to the omission of nonparametric descriptive methods,[2] and a shorter-than-desired discussion of how event history analysis relates to causal modeling.[3]

I have two minor issues with the book. First, I would have liked, particularly in the parametric models chapter, more elaboration on the differences between (log-relative)

[1] `http://www.u.arizona.edu/ bsjones/eventhistory.html`

[2] The webpage does host a handout on the Kaplan-Meier survivor function. Readers who want to learn more about these methods may consult texts like Collett (ibid.) and Hosmer & Lemeshow (ibid.).

[3] Blossfeld & Rohwer (ibid., ch. 1) discuss this with respect to cross-sectional, panel, and event history data structures.

hazard and (log-expected) duration metric regression models.[4] The parameterization of the scale parameter λ differs between the two classes of models by only a minus sign, but neglecting to keep this crucial difference in mind can turn out to be disastrous when one is estimating models, interpreting parameter estimates and discussing covariate effects, or coding likelihood functions. Whether to refer to the estimated model as a "hazard" or "duration model" turns out to be more than just an issue of semantics.

Second, I think there is more than meets the eye to when and why the *underlying* baseline hazard of a political phenomena appears to be more "nuisance" than "substance". B-SJ tend to favor, under most conditions, the semiparametric Cox model over parametric alternatives. Their stance is admittedly well-justified—they wisely note, for example, that parametric models may, depending on both included and omitted covariates, impose forms of temporal dependence on the *estimated* baseline hazard that are empirically incorrect. Yet the shape of the hazard may also be affected by aspects of *quasi*-experimental research designs, such as designation of when "analysis time" begins, pooling of multiple units of analysis from different time periods (e.g., Congressional careers, civil wars, etc., all begin at different time points), and inability on the part of the researcher, unlike in experiments, to impose a common right-censoring endpoint on the units "still remaining in the 'study'" (the researcher, surely, has no control over when political careers or wars actually terminate). Survival models have been developed primarily for experimental applications in medicine, epidemiology, and industry—the question of how quasi-experimental, social science settings relate to the models' own underlying assumptions is a topic that merits further exploration.

These quibbles aside, there are *lots* of reasons to like *EHM*. "Valued-addedness" is present in several parts of the book; here are some that caught my attention: an introduction to Royston and Parmar's recent flexible, spline-based modeling approach (which even Collett, surprisingly, does not cover); the excellent use of graphs to depict baseline hazard functions, and illustrate residual-based diagnostics; an introduction to the "counting process" approach; discussion of issues that may arise in models incorporating TVCs; and certainly worth mentioning again, the highly appropriate use of applications from American, comparative, and world politics for pedagogical purposes.

My favorite chapters are undoubtedly 9, 10, and 11. The section on "cure" models is especially praiseworthy, considering that Collett devotes only half a page to this, while Blossfeld & Rohwer (2002), Hosmer & Lemeshow (1999), and Tableman & Kim (2003) omit this topic entirely. The chapter on multiple events—repeatable events of the same kind, or multiple kinds of failure (i.e., "competing risks")—is also detailed. And B-SJ's discussion of the range of conceptual, substantive, and research design issues that can crop up in social science settings, as well as their recommendations on thinking about and dealing with these issues, makes the concluding chapter a definite must-read.

In conclusion, Box-Steffensmeier and Jones have written a highly accessible and incredibly thoughtful introduction to survival analysis for the social scientist. I find *Event History Modeling* to be well suited for adoption both as a graduate and self study text. Highly recommended.

References

Blossfeld, Hans-Peter, and Götz Rohwer. 2002. *Techniques of Event History Modeling: New Approaches to Causal Analysis.* Second Edition. Mahwah, NJ: Lawrence Erlbaum Associates.

Cleves, Mario, William W. Gould, and Roberto G. Gutierrez. 2004. *An Introduction to Survival Analysis Using Stata.* Revised Edition. College Station, TX: Stata Press.

Collett, David. 2003. *Modelling Survival Data in Medical Research.* Second Edition. Boca Raton, FL: Chapman & Hall/CRC.

Hosmer, David W., and Stanley Lemeshow. 1999. *Applied Survival Analysis: Regression Modeling of Time to Event Data.* New York: John Wiley.

Tableman, Mara, and Jong Sung Kim. 2004. *Survival Analysis Using S: Analysis of Time-to-Event Data.* Boca Raton, FL: Chapman & Hall/CRC.

Yamaguchi, Kazuo. 1991. *Event History Analysis.* Newbury Park, CA: Sage.

[4] Better known as "proportional hazards" and "accelerated failure-time" models, respectively.

Section Activities

A note from our Section President

I want to start by thanking all those who have graciously agreed to serve on committees and hold officer positions for our section. We are the second largest section, hold an annual meeting, have an active listserve and web site, and publish both a newsletter and journal, so we have a lot of business and constituents to serve. A full list of the committees, their current membership, and officers is listed on the POLMETH website.

Two new committees that I want to direct your attention to are the Long Range Planning Committee and Diversity Committee. The Long Range Planning Committee is charged with planning for the growth and institutionalization of the section. The committee is charged with looking 5, 10, and 20 years ahead to see how we can continue our reputation of innovating for the betterment of the section. Please direct all suggestions to James Granato, chair of the committee (jgranato@mail.la.utexas.edu). The Diversity Committee's mission is: a) to promote and facilitate diversity within the section, especially with regard to women and other minorities; b) to improve the professional visibility of women and other minorities within the political methodology field; and c) to monitor and provide oversight with respect to these goals. Caroline Tolbert is chairing the committee and can be reached at (ctolber1@kent.edu).

Please note the call for participants in the Annual Summer Conference of the Society for Political Methodology. This is the 23rd meeting and is still one of the most valuable meetings in the discipline. We are appreciative of Jeff Gill, his colleagues, department, and university for graciously hosting us this July 20-23rd. The deadline for application is March 1st. The section is proud to call for proposals for the first annual John T. Williams Dissertation Prize. The deadline for applications is March 1st. The Prize is given in recognition of John T. Williams' contributions to graduate training and is for the best dissertation proposal in the area of political methodology. Proposals using quantitative or qualitative methods are welcome. National Science Foundation format guidelines should be followed, including a length restriction of ten pages. Proposals should be sent to John Aldrich (aldrich@duke.edu). I thank John, as well as the rest of the committee, composed of Virginia Gray, Patrick Brandt, and Burt Monroe for their service to the section. The winner for the John T. Williams Dissertation Award will be selected by June 1st.

Jan Box-Steffensmeier
The Ohio State University

Call for Participation: the 2006 Society for Political Methodology Summer Meeting

The 23rd Annual Summer Methodology Conference will be held July 20–22, 2006, on the campus of the University of California, Davis. These meetings are sponsored by the Society for Political Methodology and the APSA Political Methodology Organized Section. Further support comes from the UC Davis Department of Political Science and the National Science Foundation.

Attendance at the Summer meeting is by invitation. To apply to attend the meetings fill out the application form at the Society's webpage: http://polmeth.wustl.edu/conference.php. Applicants stipulate their willingness to give a paper, act as discussant, present a poster (all graduate students), or simply attend. The deadline for applications is March 1 this year and formal invitations will be sent out April 1. Notice that this is a bit earlier than in previous years.

Graduate students are always encouraged to apply. Graduate students who are accepted for the conference will have their expenses (airfare and conference registration) paid by a grant from the National Science Foundation. Some faculty have been able to fund the attendance of their students, which is extremely helpful to the Society.

The registration is $200 for each faculty member, which must be paid in advance of the conference. UC Davis is providing housing, meals, and entertainment for the conference participants in addition to sponsoring the conference facilities. Faculty are expected to obtain support from their home institutions to cover their conference expenses. However, reimbursement of airfares for untenured faculty who are unable to obtain support from their home institutions is available.

Davis is located between San Francisco and Sacramento next to the Napa Valley (http://psblade.ucdavis.edu/methods.contact.info.html). Nearby activities and attractions include hiking, white water rafting, wine country tours, the Mondavi Center for the Performing Arts, Lake Tahoe, the Yolo Basin wildlife preserve, and the Triple-A Affiliate of the Oakland A's.

Send any questions to jgill@ucdavis.edu. We look forward to seeing you in Davis this Summer!

Jeff Gill
University of California, Davis

THE POLITICAL METHODOLOGIST
Department of Political Science
Massachusetts Institute of Technology
Cambridge, MA 02139

The Political Methodologist is the newsletter of the Political Methodology Section of the American Political Science Association. The support of the MIT Department of Political Science in helping to defray the editorial and production costs of the newsletter is gratefully acknowledged.

Subscriptions to *TPM* are free to members of the APSA's Methodology Section. Please contact APSA (202 483-2512, `http://www.apsanet.org/about/membership-form-1.cfm`) to join the section. Dues are $25.00 per year and include a free subscription to *Political Analysis*, the quarterly journal of the section.

Submissions to *TPM* are always welcome. Articles should be sent to the editor by e-mail (`berinsky@mit.edu`) if possible. Alternatively, submissions can be made on diskette as plain ascii files sent to Adam J. Berinsky, MIT Department of Political Science, 77 Massachusetts Avenue, Cambridge, MA 02139 E53-459. LaTeX format files are especially encouraged. See the *TPM* web-site, `http://polmeth.wustl.edu/tpm.html`, for the latest information and for downloadable versions of previous issues of *The Political Methodologist*.

TPM was produced using LaTeX on a PC running MikTex and WinEdt.