The Stata Journal

The *Stata Journal* publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go "beyond the Stata manual" in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those who teach, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

For more information on the *Stata Journal*, including information for authors, see the web page

http://www.stata-journal.com

The *Stata Journal* is indexed and abstracted in the following:

- CompuMath Citation Index®
- Current Contents/Social and Behavioral Sciences®
- RePEc: Research Papers in Economics
- Science Citation Index Expanded (also known as SciSearch®)
- Social Sciences Citation Index®

The *Stata Journal*, electronic version (ISSN 1536-8734) is a publication of Stata Press. Stata and Mata are registered trademarks of StataCorp LP.

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates

The listed subscription rates include both a printed and an electronic copy unless otherwise mentioned.

Subscriptions mailed to U.S. and Canadian addresses:

3-year subscription	$195
2-year subscription	$135
1-year subscription	$ 69
1-year student subscription	$ 42
1-year university library subscription	$ 89
1-year institutional subscription	$195

Subscriptions mailed to other countries:

3-year subscription	$285
2-year subscription	$195
1-year subscription	$ 99
3-year subscription (electronic only)	$185
1-year student subscription	$ 69
1-year university library subscription	$119
1-year institutional subscription	$225

Back issues of the *Stata Journal* may be ordered online at

http://www.stata.com/bookstore/sjj.html

Individual articles three or more years old may be accessed online without charge. More recent articles may be ordered online.

http://www.stata-journal.com/archives.html

The *Stata Journal* is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the *Stata Journal*, StataCorp, 4905 Lakeway Drive, College Station, TX 77845, USA, or emailed to sj@stata.com.

Volume 10 Number 1 2010

The Stata Journal

The Stata Journal (2010)
10, Number 1, pp. 1–2

Editorial announcements

1 Birthdays 2010

The year 2010 marks two notable birthdays for the Stata community. StataCorp itself was founded 25 years ago as Computing Resource Center. The *Stata Journal* starts its 10th volume with this issue. So, happy birthday to them and happy birthday to us!

As computing-oriented people, we should perhaps have made more fuss at Stata's 16th birthday in 2001. Somehow that was overlooked, and it is a while to wait until Stata is 32, so at the *Stata Journal*, we are choosing to do something a little special at each major birthday, modulo 5. The 20th anniversary of Stata was marked by a special issue with a historical theme, accessible to subscribers and nonsubscribers alike at

http://www.stata-journal.com/sj5-1.html

For the 25th anniversary, we have decided to commemorate the Stata community by running interviews with various people who have played major parts in making Stata what it is over the last quarter-century, just as our 20th anniversary issue featured an interview with William Gould, president of StataCorp. The first interview in 2010 is with Christopher F. Baum and is included in this issue.

The year 2010 is, however, a 16th birthday too. The 16th London Users Group meeting will be held in September. We are sad to announce the December 2009 death of Ana Timberlake, who did so much for London Users Group meetings between 1995 and 2009, and also Users Group meetings in Spain, Ireland, and Poland. An appreciation of a much loved member of the community follows these announcements.

2 Indexing widened

External recognition of quality is always welcome to both producers and consumers of any academic journal. We are delighted to announce that starting with this issue, the *Stata Journal* will be indexed in two further citation indexes of Thomson Scientific: the Social Sciences Citation Index and Current Contents/Social and Behavioral Sciences. These join the Citation Index Expanded and the CompuMath Citation Index, which began indexing the *Journal* in 2005.

3 Associate Editors

Much of the burden of providing and organizing reviews of submitted articles inevitably falls on our distinguished team of associate editors. With this issue, we thank most warmly Thomas Lumley, University of Washington, who has served as an associate editor since 2001, for all his contributions; and we welcome most warmly in the same role Peter A. Lachenbruch, Oregon State University.

H. Joseph Newton and Nicholas J. Cox
Editors, *Stata Journal*

The Stata Journal (2010)
10, Number 1, pp. 3–8

A conversation with Kit Baum

Nicholas J. Cox
Department of Geography
Durham University
Durham, UK
n.j.cox@durham.ac.uk

Abstract. As explained in the *Editorial announcements* at the start of this issue, the *Stata Journal* will be featuring various interviews with selected members of the Stata community in 2010, Stata's 25th anniversary year. In this issue, we start with an online interview, dated October 14, 2009, of Christopher F. "Kit" Baum, an economics professor at Boston College. Kit has been highly active as a Stata user since the 1990s, contributing as an author and associate editor to this journal and its predecessor, the *Stata Technical Bulletin*; as a much-downloaded Stata program author; as an author of two notable Stata-based texts; as a frequent participant on Statalist and at Users Group meetings in both the United States and several European countries; and as founder and maintainer for more than a decade of the SSC archive, which now contains many hundred user-written Stata packages. In this interview, he comments on how he got into Stata and adds his own speculations for the future.

Cox: Tell us a bit about your background. How did you get into economics as a student?

Baum: I became interested in economics as a freshman in college, around the time I figured out that I was not cut out to be a physics major. I ended up with an elective course in economics in the third quarter of that academic year, taught by Professor Phil Thomas, who became my senior thesis advisor. Despite an unimpressive start, I never looked back.

Cox: Do you see yourself as an economist or an econometrician?

Baum: I am an economist who spends a lot of time doing applied econometrics and programming, both statistical/econometric and database applications. I used to teach monetary theory and policy, macroeconomics, and financial derivatives regularly, but in the last several years, I have been teaching nothing but various flavors of econometrics to undergraduates and PhD students. Despite the fact that my most highly cited article (by a mile) is Baum, Schaffer, Stillman, *Stata Journal* (SJ) 2003[1], most of my recently published research has been oriented toward the effects of uncertainty on firms' behavior, banks' behavior, and international trade flows.

1. Baum, C. F., M. E. Schaffer, and S. Stillman. 2003. Instrumental variables and GMM: Estimation and testing. *Stata Journal* 3: 1–31.

Cox: How did you get started in computing? What hardware and software were you using?

Baum: Early in my college career, I took a computer science course—the only course offered in the late 1960s!—and became intimately acquainted (as did Bill Gould) with an IBM 1620 Model II. I learned Fortran II (the only thing beyond assembler and machine language which that machine supported), worked in the computer center as my work-study job, and did quite a bit of programming. My senior thesis in economics involved developing some interactive programs for teaching economic models. As "dumb terminals" were only then becoming available, this was quite impractical, as one had to run the modeling programs on the 1620s console typewriter, but I did get top marks for effort.

When I became a PhD student at Michigan–Ann Arbor, I earned my keep by programming for two years as Professor Bob Stern's assistant, working on various data-management and modeling exercises related to international trade and tariffs. I put those programming skills to good use in my doctoral dissertation, written with Professor Phil Howrey, applying Gregory Chow's techniques for optimal control of discrete-time models to macroeconomic modeling in the presence of parameter uncertainty (a computationally burdensome problem on mainframes of the mid-1970s). That work was all done in Fortran as well. Some estimation was done in mainframe TSP.

Cox: How did you get into Stata? What other econometrics or statistics software were you using at the time?

Baum: Several years after I joined the Boston College economics faculty, one of my colleagues was working on a research project with a coauthor at Harvard or MIT which made heavy use of panel data. He came to me and insisted that we purchase and support this package called Stata, which I had not heard of before. At the time, I was teaching the first-year PhD econometrics course using Ken White's Shazam package, requiring the students to do a number of exercises and an empirical research paper. Shazam had a number of good features, but also was quite clumsy in some ways, so I migrated to the RATS package. It wasn't much more popular with the students but provided tools like vector autoregressions (VARs) and impulse response functions (IRFs) that no other package had at the time. That first version of Stata, which the department purchased for a Unix server, was version 5. I switched to using Stata in the first-year PhD econometrics course. The next time I taught the undergraduate econometrics course, I used Stata. Since that time, I have used Stata exclusively in teaching, and it is used by my colleagues in all sections of undergraduate econometrics. I still turn to RATS occasionally for specialty capabilities such as multivariate GARCH, but it may be that Stata's new commands in version 11 can do what I need in that area as well.

Cox: A very big contribution you have made to the Stata community is setting up and maintaining the SSC archive over several years. Tell us how that came about.

Baum: My involvement here came as a side effect of becoming involved with RePEc, Research Papers in Economics, a volunteer organization that has become the largest col-

lection of economics-related bibliographic information. RePEc was started by Thomas Krichel to enable the sharing of working papers (preprints), which are the predominant mode of rapid communication of research results in economics. Boston College Economics was one of the earliest participants in this effort, which was an outgrowth of earlier efforts in the United Kingdom. It became clear that we could not only document (and provide downloads for) working papers, but we could also document journal articles, monographs—and in a blinding flash, it came to me: why not software?

This was a controversial notion, and one that was somewhat of a hard sell to the rest of the RePEc team: why would we create software archives and commingle them in a sense with working paper and journal article archives? If it were not for Stata, this concept would probably not have emerged. Many other kinds of software have their own repositories (CPAN for Perl, for instance). Some software vendors host archives (e.g., MATLAB routines at the MathWorks or RATS routines at Estima). Stata software, in the form of ado-files and help files, comprised complete routines: if well-written, far more useful than most software circulating for other statistical packages and languages, in that Stata user-written commands are "first-class citizens" once installed on the adopath.

At the time (1998), Stata routines were "published" as electronic supplements to the *Stata Technical Bulletin* (STB) and disseminated on 3.5-inch floppy disks mailed with the hardcopy STB. The latest user-written routines were included as in-line messages to Statalist, but the code was often mangled by line wraps and the like. It seemed that having an archive from which these files could be downloaded would make sense, and the RePEc architecture supported that as soon as the other team members grudgingly accepted my proposals to create a "software template" that could provide bibliographic information about software. That was not limited to Stata, of course; any source code can be housed in the archive, and there is a nontrivial number of components in the SSC archive in other languages (however, binary code in the form of `.exe` is banned for security reasons).

The popularity of the SSC archive as a single source for user-written Stata commands was obviously not what StataCorp intended when it created the `net` commands, but since then the market has spoken. Most user-programmers find it a useful alternative to maintaining their own `net` site. Nick Cox and I wrote `archutil` (STB, 1999,[2] 2000[3]) to implement `net` for the SSC archive, and StataCorp responded with the full-featured `ssc` command. More recently, the development of `adoupdate` has made it as easy to stay up to date with user-written software as `update` does with official Stata. It has been a very fruitful collaboration.

2. Baum, C. F., and N. J. Cox. 1999. ip29: Metadata for user-written contributions to Stata programming language. *Stata Technical Bulletin* 52: 10–12. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 121–124. College Station, TX: Stata Press.
3. Cox, N. J., and C. F. Baum. 2000. ip29.1: Metadata for user-written contributions to Stata programming language: Extensions. *Stata Technical Bulletin* 54: 21–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 124–126. College Station, TX: Stata Press.

Cox: Do you view yourself as a gatekeeper in any sense over SSC?

Baum: I would have reacted to this question "of course not", but on this very day, I found myself looking at the ado-file of a submission and composing an email to the author indicating that without several changes I could not accept the routine (e.g., thou shalt not wantonly strew global macros about the user's workspace). The author kindly complied and provided a new version, dealing with this and a couple of other issues, the same day. But generally, I do not read the code, especially that of well-known authors. It is clearly stated that the material in the archive bears no warranty by virtue of its inclusion. If clear flaws are pointed out, and an author does not respond to a request to make corrections to the routine, I will remove it—but that has very rarely happened.

Cox: You have contributed several much-used Stata commands yourself. Do you have any particular favorites?

Baum: Of course, I am very fond of `ivreg2`, although Mark Schaffer has contributed much more of the inventive programming to that routine as it has become more and more complex. I am proud of the fact that StataCorp was compelled to improve official Stata's capabilities in this area, developing the `ivregress` routine. Between `ivregress` and `ivreg2`, I believe Stata has the best feature set for dealing with instrumental-variables estimation of any available econometric software. Apart from `ivreg2`, my major contributions have been in the time-series area: the first implementation of vector autoregressions (`vecar`, made wholly obsolete by Stata's `var` suite); rolling regressions (`rollreg`, largely superseded by the `rolling:` prefix); `dfgls`, with Richard Sperling, adopted by StataCorp; `kpss`, `modlpr`, and `roblpr` tests for long memory; and several unit-root tests, now available in official Stata 11. I believe that the popularity of several of these routines has had a real influence on the course of development at StataCorp and, in particular, on the strengthening of its capabilities in the time-series domain. That makes Stata much more competitive with those packages and languages which have traditionally been used for time-series work. Being able to do that work while remaining in the Stata environment is very helpful.

Cox: You have presented at many Stata Users Group meetings. Perhaps you could say something to those who have never been about what they are like.

Baum: The Stata Users Group meetings are always enjoyable experiences for new users and experienced users alike. The user presentations are an eclectic mix of topics from a broad set of disciplines, which is very interesting for those of us who only attend economics and finance seminars. The techniques of interest in medical fields, for instance, often provide inspiration for adapting some of that methodology to good advantage in my own research, although I do not routinely peruse the biostatistics literature. Ample time is given in coffee breaks, luncheons, and dinners for participants to chat with speakers, both users and StataCorp staff, about their work. Last summer's DC09 Conference was quite large, with over 100 present each day; but even in that venue, there were many opportunities for fruitful interactions.

Cox: You have recently written two fat books for Stata Press...

Baum: Well, not fat in comparison to other recent Stata Press offerings, but long enough to cover the subject in reasonable detail, I believe. I am exceedingly pleased and humbled by the popularity of *An Introduction to Modern Econometrics Using Stata*.[4] As Austin Nichols' excellent SJ review[5] indicates, it is not a textbook from which one learns econometrics, but a guide for those who know some econometrics and need guidance beyond theory in applying it to their research. Those of us who do empirical research realize that a small fraction of the research effort involves actual estimation, with a sizable set of data-management tasks preceding estimation and analysis of econometric results. Likewise, a research project often entails a lot of work in producing tables, graphs, and the like, and that should be automated. The book stresses how both of those important ancillary activities are important adjuncts to performing the proper estimation and conducting the appropriate diagnostic tests. I look forward to reworking the book someday to include discussion of the new features added in Stata 11, such as factor variables and margins, that further enhance researchers' efforts.

I am also very pleased to see that *An Introduction to Stata Programming*[6] has been quite popular. I have always thought that StataCorp did not make it easy to get into Stata programming, as most users did not have the *Stata Programming Reference Manual* available, and numerous important details were only available in that manual. With the enlightened change in Stata 11, making all the manuals available to every Stata user in very convenient PDF, the user experience has radically changed. Users are now much more able to use do-file programming to good advantage with the help of the manuals and a copy of ISP.

Cox: By the way, we gather that you are a Macintosh user...

Baum: I would say that I am a Unix user. During my 30+ years at Boston College (my one and only academic employer), I have always used Unix machines for serious computing. In the early years, that meant Digital Equipment Corporation's flavor of Unix known as VMS. When we first acquired a "compute server" for academic computing, it was IBM's AIX. Later, we acquired a multiprocessor Sun server, and that provided access to Stata with a useful amount of memory and disk space. When that machine became obsolete, we replaced it with an Apple xServe. In addition to that machine, we have a Mac Pro with four-core Intel Xeon to run Stata/MP 11, and the university's research computing unit has a Stata/MP license for their Linux cluster.

The point here is not that I use Macintosh: the point is that I use Unix tools (including the console version of Stata) heavily, and I find that Mac OS X is a convenient way to use those tools. Linux would be a good choice too, although not quite as well supported in

4. Baum, C. F. 2006. *An Introduction to Modern Econometrics Using Stata*. College Station, TX: Stata Press.
5. Nichols, A. 2007. Review of An Introduction to Modern Econometrics Using Stata by Baum. *Stata Journal* 7: 131–136.
6. Baum, C. F. 2009. *An Introduction to Stata Programming*. College Station, TX: Stata Press.

terms of software. But the innate stability and professional quality of Unix/Linux/Mac OS X operating systems stands in stark contrast to the various versions of Windows. I've heard Bill Gould tell us that Windows Vista is actually a great OS, but his beliefs do not seem to have been shared by the marketplace. Windows 7 may be a great choice, but I'd rather rely on an operating system that has evolved over several decades, whose flaws are found by programmers rather than end-users. But one interesting feature of Apple hardware these days is that it can run any flavor of Windows (and Linux, for that matter) in addition to Mac OS X. Its industrial design is also highly innovative, and many of the features common to all machines these days were pioneered by Apple. So I am quite happy to use Apple hardware and software, and never averse to moving my compute-bound tasks over to Linux if I need more horsepower. Like about half of my colleagues in the economics department, I just don't use Windows.

Cox: How do you see Stata use developing on your side of statistical science?

Baum: I hope that with the aggressive development of new econometric capabilities, evident in the long list of enhancements to Stata 11, more researchers in economics and finance will find Stata to be a very useful platform. Those who use Monte Carlo simulation regularly are often unaware that Stata makes that very straightforward and efficient. My colleagues who use vector autoregressions and related techniques are not fully aware that Stata is very competitive in this area. The addition of state-space modeling, multivariate ARCH, and related time-series techniques should make reliance on Stata more workable for those of us with needs in that area, and the addition of the `gmm` command is an important step forward for our field of research. Nevertheless, I still believe that several important capabilities are lacking: for instance, estimation and forecasting of nonlinear simultaneous models (NL3SLS or FIML), support for contour plots/likelihood profile plots, and support for some forms of 3D graphics. These are areas where one must be apologetic when colleagues ask "can Stata do ...". I hope to see developments in these areas, as well as official support of some capabilities for publication-quality output. On the whole, though, StataCorp has been highly responsive to researchers' expressed needs, and that explains a lot of its growing popularity among researchers in economics, finance, and related disciplines. As an economist, I must point out that the reasonable price tag certainly doesn't hurt. I look forward to further enhancements that will further improve my ability to perform research tasks efficiently and effectively.

Cox: Thank you, Kit, for your time and thoughts.

The Stata Journal (2010)
10, Number 1, pp. 9–10

Ana Isabel Palma Carlos Timberlake (1943–2009)

Ana Timberlake died on 17 December 2009 after a long fight with idiopathic pulmonary fibrosis. A kind and warm-hearted but also very strong and independent woman, she was the founder and chairman of Timberlake Consultants Limited (TCL), a firm specializing in statistical software and consultancy, with particular focus on medical research and econometric modeling applications.

Ana was born in Portugal, the daughter of a civil engineer. Her early childhood was on the Pego do Altar Dam construction site in the Alentejo, where her father, Armando da Palma Carlos, was the resident engineer. Her family were no strangers to strong women: her aunt Elina Guimarães has been described as the first feminist in Portugal.

Ana took her first degree in mathematics at Lisbon University before moving to Britain in 1969 to do a Master's degree in statistics at Southampton University. She was then employed at a small research unit in London, Planning and Transport Research and Computation (PTRC).

One of her early assignments at PTRC was a detailed analysis of the U.S. survey upon which the British breathalyzer test had been based. The original dataset consisted of 8,000 accident records and a further 8,000 controls. Early tabulations suggested that

driving behavior actually improved with the intake of a small amount of alcohol, which was contrary to all former laboratory research. However, after Ana had standardized the data for weather, vehicle age, driving experience, and so forth, it did become clear that there was a degradation in driving ability with alcohol intake.

While working at PTRC, Ana started upon a doctoral degree at Queen Mary College, London, under David Barton, into the use of mathematics by scientists and engineers. This research caused some consternation among certain professions when it showed that the level of mathematical sophistication generally employed by them barely exceeded that gained in a O-level mathematics course, as then attained by many 15- or 16-year-olds.

Ana then joined Control Data Corporation in London as a statistical consultant. Here she started to form her lifelong and worldwide associations with various academics, researchers, and developers of statistical techniques and software. This led Ana to form TCL in 1982 as a firm that specialized in bridging the gap between research and development and the application of statistics and modeling. She somehow managed to juggle her family life with successfully building her firm into a fully international business.

TCL's links with Stata are especially strong. Timberlake and its associate companies distribute Stata in the United Kingdom, Brazil, Ireland, Poland, Portugal, and Spain. Ana had an impact on many Stata users by starting users meetings. The first was held in London in 1995 and annual meetings have been held there ever since. By the end of 2009, 53 meetings had been held worldwide, all of them following essentially the same mix of talks and tutorials as originally devised by Ana, and all ending with "Wishes and grumbles", Ana's happy phrase for a session devoted to user comments.

Ana will be very sadly missed, not only by family and friends but also by many in the wider statistical community who enjoyed their contacts and meetings with her. Those who knew her personally admired her dignity and courage as she battled over the last few years with a cruel condition.

Teresa Timberlake and Nicholas J. Cox

The Stata Journal (2010)
10, Number 1, pp. 11–29

Direct and indirect effects in a logit model

Maarten L. Buis
Department of Sociology
Tübingen University
Tübingen, Germany
maarten.buis@uni-tuebingen.de

Abstract. In this article, I discuss a method by Erikson et al. (2005, *Proceedings of the National Academy of Science* 102: 9730–9733) for decomposing a total effect in a logit model into direct and indirect effects. Moreover, I extend this method in three ways. First, in the original method the variable through which the indirect effect occurs is assumed to be normally distributed. In this article, the method is generalized by allowing this variable to have any distribution. Second, the original method did not provide standard errors for the estimates. In this article, the bootstrap is proposed as a method of providing those. Third, I show how to include control variables in this decomposition, which was not allowed in the original method. The original method and these extensions are implemented in the `ldecomp` command.

Keywords: st0182, ldecomp, mediation, intervening variable, logit, direct effect, indirect effect

In this article, I aim to study direct, indirect, and total effects in a logit model and, in particular, a generalization of a method by Erikson et al. (2005) for computing those effects when the variable whose effect we want to decompose is a categorical variable. Direct, indirect, and total effects are studied to reveal a mechanism through which one variable affects another variable. The idea is illustrated using figure 1: There is a variable X that has an effect on a variable Y, but part of this effect occurs through another variable, Z. This indirect effect occurs because X influences Z, which in turn influences Y. Within this framework, the effect of X on Y while controlling for Z is called the direct effect. The indirect and direct effect together form the total effect of X on Y. The indirect effect is the part of the effect of X on Y that can be explained, while the direct effect is the residual, or unexplained, part of the effect. The aim of such an analysis is to try to explain why a variable X influences a variable Y by specifying a mechanism: the effect occurs through the third variable, Z. Z is called an intervening variable. It differs from a confounding variable in the direction of the effect between X and Z. If Z was a confounding variable, then it would affect X rather than the other way around.

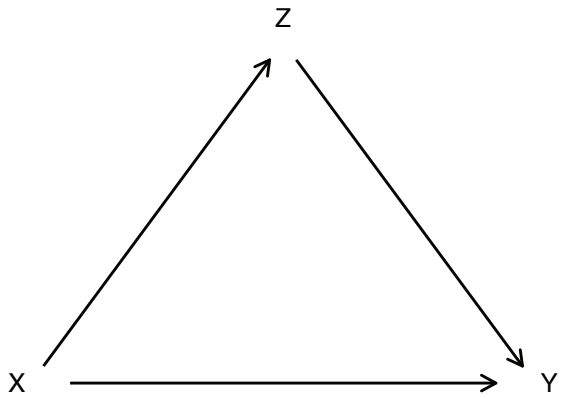

Figure 1. Direct, indirect, and total effects.

X, Y, and Z could be many things. For example, within political science there is literature (e.g., Campbell et al. [1960]) that explains the effect of party identification (X) on voting for a party (Y) in part through how one perceives the issues and the parties (Z). The idea is that someone's party identification is a relatively stable characteristic of a person, almost like a personality trait, which can color/distort the perceptions of the issues and the positions of the candidates or parties,[1] which in turn influences voting behavior. This indirect effect thus represents a mechanism through which party identification influences voting behavior, and we want to know how important that mechanism is relative to the total effect of party identification.

Alternatively, within sociology there is literature (e.g., Boudon [1974]) that looks at the degree to which children from different social backgrounds (X) have different probabilities of attending college (Y) and the part of this effect that can be ascribed to an indirect effect through academic performance in high school (Z): Children from a higher class background do well in high school, and those children who do well in high school are more likely to attend college. Possible explanations for this could be that higher class parents have better genes (nature) or are able to provide a more intellectually stimulating environment for their children (nurture) or both. Again, the aim is to estimate how important this mechanism is relative to the total effect of family background.

1. A striking example of this is reported by Bartels (2002), who found in a survey held in 1988 that more than 50% of the respondents who identified themselves as a "strong Democrat" thought that the inflation got worse or a lot worse during the Reagan presidency, while only 13% of the "strong Republicans" thought that this was the case. In actual fact, the inflation rate in consumer prices fell from 13.5% in 1980 to 4.1% in 1988.

More generally, the common practice of comparing regression coefficients before and after controlling for a set of other covariates is implicitly an attempt at decomposing a total effect into a direct and an indirect effect. Within linear regression ([R] **regress**), this is an easy and convenient way of estimating the direct, indirect and total effect. In this case, the total effect of X is estimated using a regression that does not control for Z. The direct effect of X is the effect of X in a regression with both X and Z as explanatory variables, and the indirect effect is the difference between these two, because the total effect is the sum of the direct and indirect effect. However, this method will not work when dealing with nonlinear models like logistic regression ([R] **logit**). In this article, I will show why this is the case. I will also discuss a method proposed by Erikson et al. (2005) for solving this problem, I will propose a generalization of this method, and I will introduce the `ldecomp` command, which implements these methods in Stata.

1 The problem of indirect effects in a logit model

The key problem when estimating the direct, indirect, and total effects is that the standard method of estimating them—comparing estimates from models that do and do not control for Z—will not work in nonlinear models like logistic regression. The easiest way to see that is in an example where there is no indirect effect.

To illustrate the problem, I create artificial data where I know that there cannot be an indirect effect of X through Z on Y; I show that the naïve method finds a substantial indirect effect. The variable Z is created so that it can take three values (0, 1, and 2), and the variable X is created so that it can take two values (0 and 1). There is no relationship between X and Z (a low X individual is as likely to be a high value on Z as a high X individual). So in this example, there is no indirect effect of X on Y through Z. We start by creating the variables X and Z:

```
. drop _all
. set obs 60000
obs was 0, now 60000
. generate z = ceil(_n/20000) - 1
. bysort z: generate x = ceil(_n/10000) - 1
. tab x z
```

	z			
x	0	1	2	Total
0	10,000	10,000	10,000	30,000
1	10,000	10,000	10,000	30,000
Total	20,000	20,000	20,000	60,000

Next we create the dependent variable Y according to the logistic regression equation $P(y=1)/1-P(y=1) = -4+4X+2Z$, as discussed in Buis (2007b).

```
. set seed 12345
. generate y = runiform() < invlogit(-4 + 4*x + 2*z)
```

Next we compute the naïve estimates of the indirect effect:

```
. quietly logit y x z
. estimates store direct
. local direct = _b[x]
. quietly logit y x
. estimates store total
. local total = _b[x]
. estimates tab direct total
```

Variable	direct	total
x	4.0391332	2.6256242
z	2.026339	
_cons	-4.0452305	-1.3123133

```
. display "naive indirect effect = " `total´ - `direct´
naive indirect effect = -1.413509
```

Someone who did not know that the data were created such that the indirect effect is exactly zero and so used the naïve estimate of the indirect effect would conclude that the indirect effect is about 54% of the total effect; the fact that it has the opposite sign from the total effect would suggest that this indirect effect has a noticeable dampening influence on the effect of X on Y. This is not caused by sampling variation but instead caused by a structural bias in the naïve estimator.

The reason for this bias is that a logistic regression is comparison of proportions that have first been transformed into log odds-ratios;[2] this is illustrated in figure 2. It consists of four vertical lines: on the two outer lines are plotted the probabilities of success on Y, while the equivalent log odds are plotted on the two inner lines. The two left lines represent the high X group, while the two right lines represent the low X group. The open symbols and the solid arrows show how the probabilities are transformed into log odds and how, within each category of Z, the log odds of the high X group are compared with the log odds of the low X group.

This represents what happens when we estimate a `logit` model with both X and Z as explanatory variables. When we leave the variable Z out of the model—for example, because we want to estimate the total effect—we are in effect first computing the average of the proportions and then transforming them into log odds. This is represented by the closed circles and the dashed arrows. However, the more extreme values are less extreme in the probability metric than in the log odds metric; that is, the probabilities close to either 0 or 1 are more bunched together than their log odds counterparts. Therefore, computing the average proportion before transforming the proportions into log odds means that the extreme values are less influential than they would have been if the

2. The odds and the log odds contain exactly the same information as the proportion, just presented differently. The proportions times a hundred tell you how many people out of a hundred are expected to attend college, while the odds tell you how many people are expected to attend college for every person who does not attend college. The odds (O) can be derived from the proportion (p) in the following way: $O = p/1 - p$.

means were computed in the log odds metric, so the average is being pulled toward the less extreme categories. As a consequence, the effect in terms of log odds will be less when Z is left out of the model, even if there is no indirect effect. This problem is very similar to the problems discussed in Bartus (2005) and Buis (2007a).

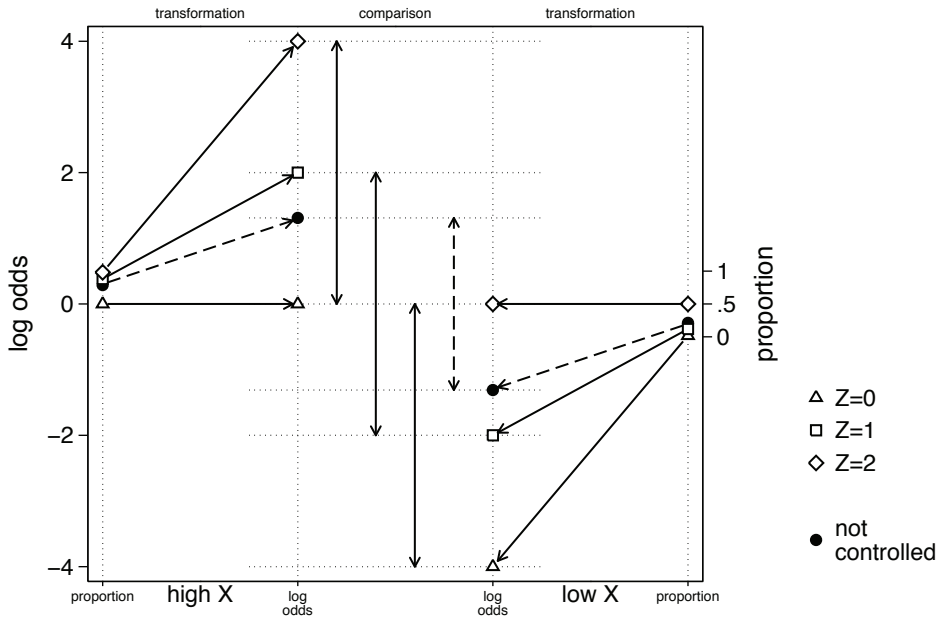

Figure 2. The effect of X with and without controlling for Z.

2 A solution

2.1 Outline

For there to be an indirect effect of X on Y through Z, X needs to have an effect on Z and Z has to have an effect on Y. For example, children from higher classes (X) are likely to do better in high school (Z) than children from lower classes, and those who have done better in high school (Z) are more likely to enter college (Y). In figure 2, there was no indirect effect because the distribution of Z was the same for both categories of X. In figure 3, this has changed to include an indirect effect. The distribution of Z is represented by the size of the symbols. So in this new example, there are within the high X group more high Z individuals than medium Z individuals and more medium Z individuals than low Z individuals. The distribution of Z for the low X group is exactly the opposite. Now there are two reasons why high X individuals are more likely to belong to the high Y group: 1) They are more likely to belong to the high Z group, and

those who belong to the high Z group are more likely to belong to the high Y group. This is the indirect effect of X through Z on Y. Figure 3 shows this in the following way: for the high X group, the high Z group is more influential when computing the average probability because the high Zs are more numerous, while for the low X group, the low Zs are more influential. This leads to a larger difference between the high and low status group. 2) They are more likely to belong to the high Y group even when compared with a low X group that has the same value on Z. Figure 3 shows this in the following way: for each level of Z, the high X group has a higher proportion belonging to the high Y group than does the low X group. This is the direct effect.

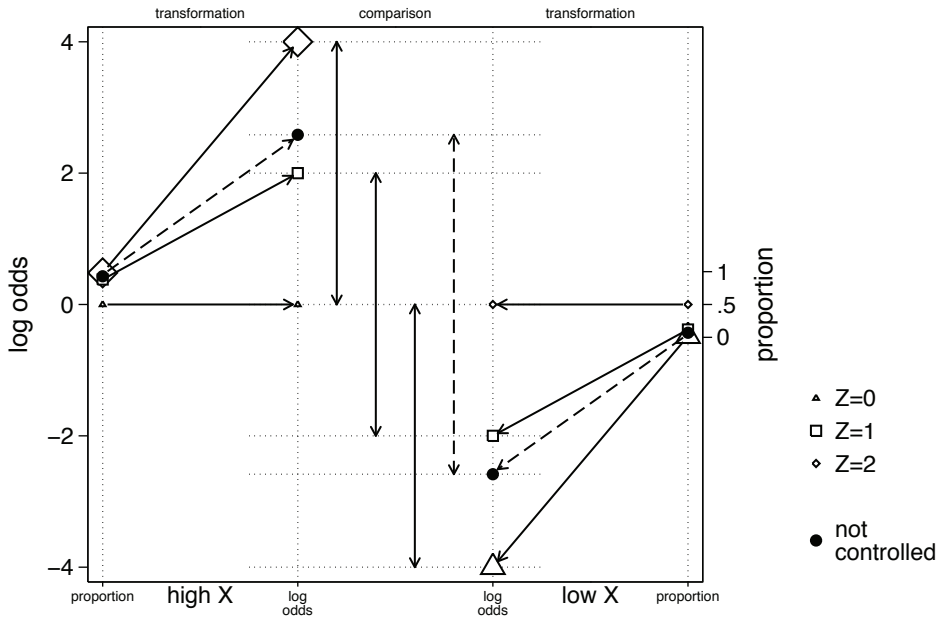

Figure 3. Indirect and direct effects.

Erikson et al. (2005) propose two related methods for estimating a direct, indirect, and total effect such that the direct and indirect effects add up to the total effect. The first method is illustrated in figure 4 and consists of the following steps:

1. Estimate a logistic regression using both X and Z, and optionally, the interactions between X and Z.

2. Predict for each individual the log odds of success on Y (this is the linear predictor) and transform these to probabilities.

3. Compute within each group of X the average predicted probability and transform these to log odds. The difference in these log odds between the groups of X represents the total effect.

4. Assume that the low X individuals are actually high X individuals, predict for each low X individual the log odds in this counterfactual scenario, and transform these to probabilities.

5. Compute the average of these probabilities. This is the counterfactual probability of success on Y for high X individuals if they had the distribution of Z of the low X individuals. These are then transformed into log odds.

6. The high X individuals and this counterfactual group differ with respect to the distribution of Z, but the probabilities conditional on X and Z are kept constant. Therefore, comparing these groups gives the effect of X caused by the differences in the distribution of Z, that is, the indirect effect.

7. The low X individuals and the counterfactual group differ with respect to the probabilities conditional on X and Z, but the distribution of Z is kept constant. Therefore, comparing these groups gives the effect of X while controlling for the distribution of Z, that is, the direct effect.

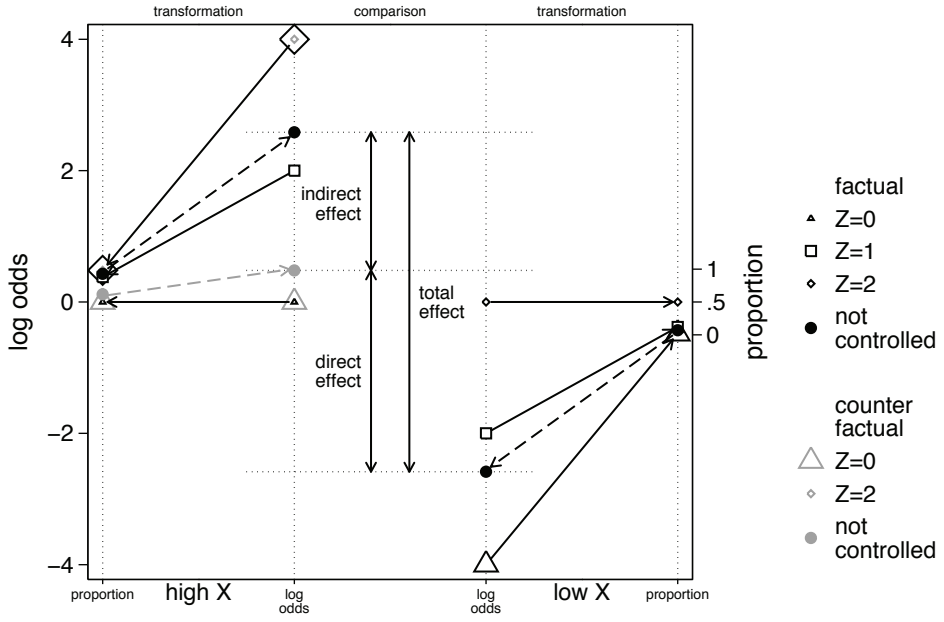

Figure 4. Method 1.

Figure 4 above and (1) on the following page both show that this way the total effect is always the sum of the direct and indirect effect. In (1), the O is the odds of success on Y, the first subscript represents the logistic regression coefficients, and the second subscript represents the distribution of Z.[3] So $O_{x=1,z\,|\,x=1}$ is the odds of success for the high X group, while $O_{x=1,z\,|\,x=0}$ is the counterfactual odds of success for the high X group if it had the distribution of Z of the low X group.

$$\underbrace{\ln(O_{x=1,z\,|\,x=1}) - \ln(O_{x=0,z\,|\,x=0})}_{\text{total}} = \underbrace{\ln(O_{x=0,z\,|\,x=1}) - \ln(O_{x=0,z\,|\,x=0})}_{\text{indirect}} + \underbrace{\ln(O_{x=1,z\,|\,x=1}) - \ln(O_{x=0,z\,|\,x=1})}_{\text{direct}} \tag{1}$$

Using the rule that $\ln(a) - \ln(b) = \ln(a/b)$, it can be shown that these effects are actually log odds-ratios:

$$\underbrace{\ln\left(\frac{O_{x=1,z\,|\,x=1}}{O_{x=0,z\,|\,x=0}}\right)}_{\text{total}} = \underbrace{\ln\left(\frac{O_{x=0,z\,|\,x=1}}{O_{x=0,z\,|\,x=0}}\right)}_{\text{indirect}} + \underbrace{\ln\left(\frac{O_{x=1,z\,|\,x=1}}{O_{x=0,z\,|\,x=1}}\right)}_{\text{direct}} \tag{2}$$

This means that this decomposition can also be presented in terms of odds ratios, by exponentiating both sides of (2). Because of the rule that $\exp(a+b) = \exp(a) \times \exp(b)$, the total effect is now the product of the direct and indirect effects:

$$\underbrace{\frac{O_{x=1,z\,|\,x=1}}{O_{x=0,z\,|\,x=0}}}_{\text{total}} = \underbrace{\frac{O_{x=0,z\,|\,x=1}}{O_{x=0,z\,|\,x=0}}}_{\text{indirect}} \times \underbrace{\frac{O_{x=1,z\,|\,x=1}}{O_{x=0,z\,|\,x=1}}}_{\text{direct}} \tag{3}$$

The second method of decomposing the total effect into a direct and indirect effect proposed by Erikson et al. (2005) is exactly the same as the first method, except that it uses the counterfactual probability of success on Y for the low X individuals assuming that they have the distribution of Z of the high X individuals, as is illustrated in figure 5. The logic behind these two methods is exactly the same, but they do not have to result in exactly the same estimates for the direct and indirect effects, though they are often very close. Jackson et al. (2007) propose to solve this problem by computing the size of the indirect effect relative to the total effect using both methods and report the average of the two.

3. In fact, as can be seen in the description of this technique, these are averaged estimated log odds, so some people would prefer to incorporate that in the notation by adding hats and bars. However, I think that this notation is complicated enough as is, and hats and bars are not crucial for the points that these equations make, so I chose not to include them.

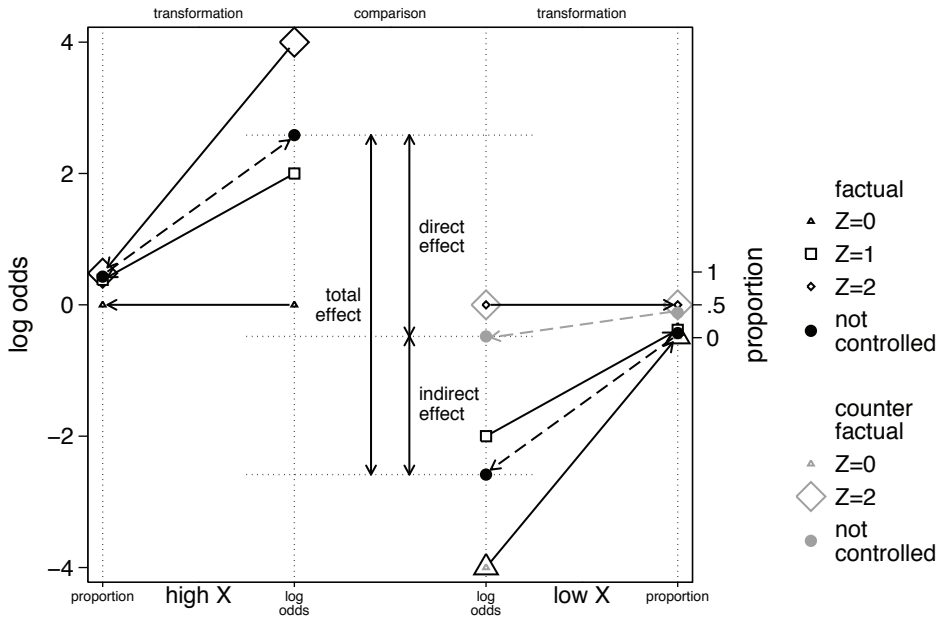

Figure 5. Method 2.

Another difficulty with this method is that the variable whose effect we want to decompose (X) must be categorical and the number of effects that must be decomposed increases very quickly with the number of categories in X. The reason is that the decomposition will be different for all pairwise comparisons of categories. Normally, a categorical explanatory variable consisting of C categories can be entered as $C-1$ dummies, with each dummy representing a comparison between the reference category and one of the other categories, leading to $C-1$ effects. All other pairwise comparisons can be directly derived from those effects. Consider a logit model with a categorical variable with three categories—low ($x=0$), middle ($x=1$), and high ($x=2$)—and the low category was chosen to be the reference category. This will result in two odds ratios: the odds ratio comparing the medium and the low group, and the odds ratio comparing the high and the low group, or $O_{x=1}/O_{x=0}$ and $O_{x=2}/O_{x=0}$, respectfully. This excludes the third possible comparison: the medium versus the high group. This comparison can be derived directly from the other two odds ratios by dividing the two odds ratios:

$$\frac{O_{x=1}/O_{x=0}}{O_{x=2}/O_{x=0}} = \frac{O_{x=1}}{O_{x=0}} \times \frac{O_{x=0}}{O_{x=2}} = \frac{O_{x=1}}{O_{x=2}}$$

This same reasoning, however, will not work for the decomposition. Consider the three indirect effects we get when using the first method: $O_{x=0,z\,|\,x=1}/O_{x=0,z\,|\,x=0}$, $O_{x=0,z\,|\,x=2}/O_{x=0,z\,|\,x=0}$, and $O_{x=1,z\,|\,x=2}/O_{x=1,z\,|\,x=1}$.

$$\frac{O_{x=0,z\,|\,x=2}/O_{x=0,z\,|\,x=0}}{O_{x=0,z\,|\,x=1}/O_{x=0,z\,|\,x=0}} \neq \frac{O_{x=1,z\,|\,x=2}}{O_{x=1,z\,|\,x=1}}$$

So with this decomposition, we cannot get away with only presenting all comparisons with a single reference category (leading to $C-1$ effects), but we will have to display the decomposition for all pairwise comparisons (leading to $\binom{C}{2}$ effects).

2.2 Generalization

In the original formulation of this method, the variable through which the indirect effect occurs (Z) has to be normally distributed. This is because of the way Erikson et al. (2005) propose to compute the average predicted probabilities. As discussed in steps 3 and 5 in the previous section, these averaged predicted probabilities play a key role in this decomposition. Erikson et al. (2005) compute these averages by assuming that Z follows a normal distribution, and then they integrate over this distribution. This method is generalized in this article to allow Z to follow any distribution by computing the average predicted probability by using the arithmetic mean of the predicted and counterfactual probabilities, thereby integrating over the empirical distribution of Z instead of over a normal distribution. As an added bonus, this method is also easier to compute because the integration over the normal distribution has to be done numerically since there is no closed-form solution for this integral. For these reasons, the generalized method is the default method in the `ldecomp` command, which implements both decompositions in Stata.

2.3 Standard errors

To get an idea about the degree of uncertainty around these estimates, one would usually also estimate standard errors. These are not provided by Erikson et al. (2005) and Jackson et al. (2007), but can be easily computed using the bootstrap (Efron and Tibshirani 1993). The bootstrap uses the idea that the standard error is the result of the following thought experiment: Assume we could draw many samples from the population and compute a statistic in each of these samples. Because these samples will slightly differ from one another, so will the estimates of the statistic. The standard error is the standard deviation of these different estimates and indicates how much variation one can expect due to the estimate being based on a random sample. Drawing many random samples from the population is not practical, but we do have a good estimate of the population—the sample—and we can without any difficulty draw (with replacement) many samples from this "estimate of the population". So when using the bootstrap, many samples are drawn with replacement from the observed sample, the statistic is computed within each sample, and the estimate of the standard error is the

standard deviation of these statistics. Within Stata, this process has been automated with the `bootstrap` command; see [R] **bootstrap**.

2.4 Control variables

Control variables can be included in this method by estimating the logit model from step 1 (in section 2.1) including these control variables, but afterward, at steps 2 and 4, fixing their value at one specific value, for example their mean.

3 Implementation in Stata

Both the original method proposed by Erikson et al. (2005) and the generalizations proposed in this article have been implemented in Stata as the `ldecomp` command.

3.1 Syntax

`ldecomp` *depvar* [*control_var1* [...]] [*if*] [*in*] [*weight*], `direct(`*varname*`)`
`indirect(`*varlist*`)` [`at(`*control_var1* # [; *control_var2* #] [...] `)` `obspr`
`predpr` `predodds` `or` `rindirect` `normal` `range(`# #`)` `nip(`#`)` `interactions`
`nolegend` `nodecomp` `nobootstrap` *bootstrap_options*]

`fweights`, `pweights`, and `iweights` are allowed when the `nobootstrap` option is specified.

3.2 Options

`direct(`*varname*`)` specifies the variable whose direct effect we want to decompose into an indirect and total effect. This has to be a categorical variable, and each value of *varname* is assumed to represent a group.

`indirect(`*varlist*`)` specifies the variable(s) through which the indirect effect occurs. By default, multiple variables are allowed and these can follow any distribution. If the `normal` option is specified, only one variable may be entered, and this variable is assumed to be normally distributed.

`at(`*control_var1* # [; *control_var2* #] [...] `)` specifies the values at which the control variables are to be fixed. The default is to fix the control variables at their mean value.

`obspr` specifies that a table of the observed proportions be displayed.

`predpr` specifies that a table of predicted and counterfactual proportions be displayed. If the `normal` option is not specified and there are no control variables, then the diagonal elements of this table will be exactly the same as the observed proportions.

`predodds` specifies that a table of predicted and counterfactual odds be displayed.

`or` specifies that the decomposition be displayed in terms of odds ratios instead of log odds-ratios.

`rindirect` specifies that the relative contributions of the indirect effects to the total effect (in terms of log odds-ratios) be displayed.

`normal` specifies that the predicted and counterfactual proportions be computed according to the method specified by Erikson et al. (2005). This means that the variable specified in `indirect()` is assumed to be normally distributed. This option was primarily added for compatibility with Erikson et al. (2005). By default, `ldecomp` uses the more flexible method described in this article.

`range(# #)` specifies the range over which the numerical integration of the variable specified in `indirect()` is to be performed. The default is the minimum of that variable minus 10% of the range of the variable and the maximum of the variable plus 10% of the range of the variable. This option may only be specified with the `normal` option because in the default method there is no need for numerical integration.

`nip(#)` specifies the number of integration points used in the numerical integration of the variable specified in `indirect()`. The default is `nip(1000)`. This option may only be specified with the `normal` option because in the default method there is no need for numerical integration.

`interactions` specifies that interactions between the categories of the variable specified in `direct()` and the variable(s) specified in `indirect()` be included. In other words, the effects on the dependent variable of the variables specified in `indirect()` are allowed to differ from one another for each category of the variable specified in `direct()`. This option was added primarily for compatibility with Erikson et al. (2005).

`nolegend` suppresses the legend that is by default displayed at the bottom of the main table.

`nodecomp` prevents `ldecomp` from displaying the table of decompositions, which can be useful in combination with the `obspr`, `predpr`, or `predodds` options.

`nobootstrap` prevents `ldecomp` from using `bootstrap` to calculate standard errors.

bootstrap_options are allowable options of `bootstrap`. The following options are allowed: `reps(#)`, `strata(`*varlist*`)`, `size(#)`, `cluster(`*varlist*`)`, `idcluster(`*newvar*`)`, `bca`, `saving(`*filename*`[ , `*suboptions*`])`, `jackknifeopts(`*jkopts*`)`, `mse`, `seed(#)`, `nodots`, and `level(#)`. See [R] **bootstrap**.

3.3 Example

The use of `ldecomp` is illustrated using data from the Wisconsin Longitudinal Study (Hauser and Sewell 1957–1977), which contain data on a random sample of 10,317 men and women who graduated from Wisconsin high schools in 1957. In this example, we want to study the part of the effect of social class on the probability of entering college that can be explained by performance during high school. The dependent variable is whether a respondent ever attended college (`college`). Class is measured by the type of occupation of the father (`ocf57`). The original data contained five classes, which would lead to $\binom{5}{2} = 10$ effects that are to be decomposed. To keep the number of effects manageable, the number of classes has been reduced to three: a lower class (unskilled workers and farmers), a middle class (skilled workers), and a higher class (white collar workers, professionals, and executives). The performance during high school is measured with the percentile rank of high school grades (`hsrankq`). This means that computing the counterfactual proportions using the method proposed by Erikson et al. (2005) will be problematic, because percentile rank scores will follow a uniform distribution instead of a normal distribution. However, the default method can be used without problem because it does not make any assumption about the distribution of performance. Relevant descriptive statistics are shown below:

```
. use wisconsin, clear
(Wisconsin Longitudinal Study)
. recode ocf57 2=1 3=2 4=3 5=3
(ocf57: 6196 changes made)
. label define ocf57 1 "lower" 2 "middle" 3 "higher", modify
. label value ocf57 ocf57
. table ocf57 if !missing(hsrankq, college),
> contents(mean college mean hsrankq freq) format(%9.3g) stubwidth(15)
```

occupation of r father in 1957	mean(college)	mean(hsrankq)	Freq.
lower	.284	48.2	5218
middle	.38	50.6	868
higher	.619	56.2	2837

There are big differences between the classes in the proportion of respondents that attend college. Moreover, those classes with a low proportion attending college also tend to have done worse during high school. So part of the differences in proportions attending college could be due to differences in performance. `ldecomp` is used to estimate these direct, indirect, and total effects. In the example below, the effects are presented as odds ratios, so the total effect is the product of the indirect and direct effect. Consider the decomposition according to the first method of the difference between high class (denoted as 3) and low class (denoted as 1) students, that is, the first three coefficients of the panel labeled "`3/1`". Overall, the odds of attending college for high class students is 4.10 times as large as the odds for low class students (the total effect). Low class students would have 1.23 times higher odds of attending college if they had the same performance as high class students (indirect effect according to method 1), while the

high class students would have 3.34 times higher odds of attending college than low class students when we keep the performance constant at the level of high class students (direct effect according to method 1).

```
. ldecomp college, direct(ocf57) indirect(hsrankq) or
(running _ldecomp on estimation sample)
Bootstrap replications (50)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
Bootstrap results                               Number of obs      =      8923
                                                Replications       =        50
```

	Observed Odds Ratio	Bootstrap Std. Err.	z	P>|z|	Normal-based [95% Conf.	Interval]
2/1						
total	1.547746	.1087768	6.22	0.000	1.34858	1.776327
indirect1	1.061166	.0297615	2.12	0.034	1.004408	1.12113
direct1	1.458534	.0889065	6.19	0.000	1.294287	1.643624
indirect2	1.060416	.0292343	2.13	0.033	1.004638	1.11929
direct2	1.459565	.0889226	6.21	0.000	1.295284	1.644683
3/1						
total	4.098896	.1563335	36.99	0.000	3.80366	4.417047
indirect1	1.228616	.0205526	12.31	0.000	1.188987	1.269566
direct1	3.33619	.1231675	32.63	0.000	3.103313	3.586542
indirect2	1.22293	.0212813	11.56	0.000	1.181922	1.26536
direct2	3.351702	.1256571	32.26	0.000	3.114249	3.607259
3/2						
total	2.6483	.1803249	14.30	0.000	2.317438	3.026399
indirect1	1.157325	.03452	4.90	0.000	1.091606	1.226999
direct1	2.288295	.1361168	13.92	0.000	2.036475	2.571253
indirect2	1.153977	.0331169	4.99	0.000	1.090861	1.220745
direct2	2.294933	.1384071	13.77	0.000	2.039079	2.582889

```
in equation i/j (comparing groups i and j)
let the fist subscript of Odds be the distribution of the the indirect variable
let the second subscript of Odds be the conditional probabilities
Method 1: Indirect effect = Odds_ij/Odds_jj
          Direct effect = Odds_ii/Odds_ij
Method 2: Indirect effect = Odds_ii/Odds_ji
          Direct effect = Odds_ji/Odds_jj

value labels
   1 lower
   2 middle
   3 higher
```

To get an idea of the relative importance of the indirect effect compared with the total effect, one can add the `rindirect` option, like in the example below. This means that the sizes of the indirect effects relative to total effects are shown at the bottom of the decomposition table. These are labeled `#/#r`. So if we look again at the comparison between children from higher and lower class fathers (the rows labeled `3/1` and `3/1r`), we see that according to method 1 the indirect effect is $0.206/1.41 \times 100\% = 14.6\%$ of the total effect. According to method 2, this is $0.201/1.41 \times 100\% = 14.3\%$. So, on average, the indirect effect is 14.5% of the total effect. In general, this table shows

that the size of the indirect effect is about 14 percent of the total effects. Additionally, the example below illustrates that by leaving out the `or` option, `ldecomp` will show the direct, indirect, and total effects in terms of log odds-ratios, which means that the total effect is now the sum of the direct and indirect effects.

```
. ldecomp college, direct(ocf57) indirect(hsrankq) rind nolegend
(running _ldecomp on estimation sample)
Bootstrap replications (50)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
Bootstrap results                                 Number of obs      =      8923
                                                  Replications       =        50
```

	Observed Coef.	Bootstrap Std. Err.	z	P>|z|	Normal-based [95% Conf.	Interval]
2/1						
total	.4367997	.0681297	6.41	0.000	.303268	.5703314
indirect1	.0593679	.0244276	2.43	0.015	.0114906	.1072451
direct1	.3774319	.0684413	5.51	0.000	.2432894	.5115743
indirect2	.0586611	.0240686	2.44	0.015	.0114875	.1058347
direct2	.3781386	.0684292	5.53	0.000	.2440198	.5122575
3/1						
total	1.410718	.0421018	33.51	0.000	1.3282	1.493236
indirect1	.2058881	.014635	14.07	0.000	.177204	.2345723
direct1	1.204829	.0390007	30.89	0.000	1.12839	1.281269
indirect2	.2012494	.014978	13.44	0.000	.1718931	.2306057
direct2	1.209468	.0386461	31.30	0.000	1.133723	1.285213
3/2						
total	.9739179	.075011	12.98	0.000	.8268989	1.120937
indirect1	.1461109	.0234312	6.24	0.000	.1001865	.1920353
direct1	.8278069	.0759825	10.89	0.000	.678884	.9767298
indirect2	.1432144	.0237856	6.02	0.000	.0965955	.1898332
direct2	.8307035	.0760533	10.92	0.000	.6816418	.9797652
2/1r						
method1	.1359155	.0574128	2.37	0.018	.0233885	.2484425
method2	.1342975	.0568445	2.36	0.018	.0228844	.2457105
average	.1351065	.0571076	2.37	0.018	.0231776	.2470354
3/1r						
method1	.1459457	.0096034	15.20	0.000	.1271234	.164768
method2	.1426575	.0097455	14.64	0.000	.1235567	.1617583
average	.1443016	.0095878	15.05	0.000	.1255099	.1630933
3/2r						
method1	.1500239	.0256887	5.84	0.000	.0996749	.2003729
method2	.1470497	.0258689	5.68	0.000	.0963475	.1977519
average	.1485368	.0256919	5.78	0.000	.0981816	.198892

Notice that the size of the indirect effect relative to the total effect can be larger than 100%, negative, or both. This might puzzle people who think of these relative effects as the proportion of the total effect that can be explained by the indirect effect. However, there is no reason why the direct and indirect effect cannot have opposite signs, and in

those cases, these "weird" proportions can occur. Thinking of these numbers as the size of the indirect effect relative to the size of the total effect can help avoid confusion.

In addition to the decomposition itself, `ldecomp` can produce a number of tables that together illustrate step by step how it builds this decomposition. The first of these tables is the table of the predicted and counterfactual proportions, shown below.

```
. ldecomp college, direct(ocf57) indirect(hsrankq) predpr nodecomp
predicted and counterfactual proportions
```

	association		
distribution	lower	middle	higher
lower	.284	.366	.571
middle	.296	.38	.585
higher	.327	.415	.619

The diagonal elements in this table represent the predicted proportions, both factual distribution of performance (the rows) and factual conditional probabilities (the columns), while the off-diagonal elements represent the counterfactual proportions. For example, 28.4% of the children from lower class fathers will attend college. If these children had the same performance as the children of higher class fathers, then 32.7% of them would attend college. If they had the same conditional probabilities as the children of higher class fathers, then 57.1% would attend college. This indicates that the direct effect is stronger than the indirect effect. The next step in the computation is to transform these proportions into odds:

```
. ldecomp college, direct(ocf57) indirect(hsrankq) predodds nodecomp
predicted and counterfactual odds
```

	association		
distribution	lower	middle	higher
lower	.396	.578	1.33
middle	.421	.613	1.41
higher	.487	.71	1.62

The proportions are transformed into odds by dividing the proportion by one minus the proportion, so the odds of attending college for children from lower class fathers is $0.284/1 - 0.284 = 0.397$. The difference between this number and the number in the table is caused by rounding. This odds is interpreted as follows: for every child of a lower class father who does not go to college, there are only 0.397 children of lower class fathers who do go to college.

The results that were obtained at the beginning of this example can be computed using the predicted and counterfactual odds from the previous table. For example, if we return to the decomposition of the total effect of having a higher class father rather than a father from the lower class using method 1, we can compute it by filling in (3) with the predicted and counterfactual odds from the previous table:

$$\underbrace{\frac{1.62}{0.396}}_{\text{total}} = \underbrace{\frac{0.487}{0.396}}_{\text{indirect}} \times \underbrace{\frac{1.62}{0.487}}_{\text{direct}}$$

$$\underbrace{4.09}_{\text{total}} = \underbrace{1.23}_{\text{indirect}} \times \underbrace{3.33}_{\text{direct}}$$

This example also makes it possible to see if the naïve method is really as bad as I claim, and if the generalization that I proposed in this article is really an improvement. We know that `hsrankq` deviates considerably from a normal/Gaussian distribution,[4] so if the generalized method is an improvement on the original method, then in this example it should yield noticeably different estimates. The estimates using the original method by Erikson et al. (2005) are obtained by specifying the `normal` option in `ldecomp`.

The naïve estimate was computed as follows: First, a `logit` model of `college` on `ocf57` was estimated. The effect of `ocf57` is the naïve estimate of the total effect. Then a `logit` model of `college` on `ocf57` and `hsrankq` was estimated. The effect of `ocf57` is the naïve estimate of the direct effect. The naïve estimate of the indirect effect relative to the total effect is (total effect − direct effect) / total effect. The results are shown in table 1. This table clearly illustrates the major underestimation of the indirect effect when the naïve method is used. Moreover, the method by Erikson et al. (2005) also leads to a considerable underestimation of about a quarter of the effect obtained using the generalized method. This underestimation is the result of the incorrect assumption made by Erikson et al. (2005) that `hsrankq` is normally distributed. Because the generalized method makes no such assumption, it seems to be the safest method of the three for computing the decomposition of total effects into direct and indirect effects after a logistic regression model.

Table 1. Comparing different estimates of the size of the (average) indirect effect relative to the size of the total effect.

	generalization	Erikson et al. (2005)	naïve
middle versus low	0.135	0.110	0.009
high versus low	0.144	0.105	0.014
high versus middle	0.149	0.102	0.017

4 Summary and discussion

In this article, I began by showing why getting estimates of the direct and indirect effects in a logistic regression is hard. I then presented a method by Erikson et al. (2005) to

4. As noted before, it is a percentile rank score, so it follows a uniform distribution.

estimate these effects, and I proposed a generalization of this method. The idea is that a categorical variable X can have a direct effect on a variable Y and an indirect effect through Z. One can find the direct effect of X by comparing the log odds of successes in one category of X with the counterfactual log odds of successes of another category of X given that they have the distribution of Z of the first category. This way, the factual and counterfactual group only differ with respect to X and not with respect to the distribution of Z, thus controlling for the distribution of Z. Similarly, the indirect effect can be found by comparing the log odds of success within one category of X with the counterfactual log odds of success within that same category of X with a distribution of Z of another category of X. This way, the factual and counterfactual groups differ only with respect to the distribution of Z. In its original form, this method assumed that the variable through which the indirect effect occurs has a normal distribution. In this article, this method was generalized by allowing the variable to have any distribution. Moreover, the use of the bootstrap is proposed to estimate standard errors, and I added the possibility to include control variables.

This is a relatively new methodology, and so there are still some loose ends to tie up. First, the fact that there are two different estimates of the direct effects and two different estimates of the indirect effects is less than elegant. Second, the fact that a separate decomposition needs to be estimated for all pairwise class comparisons instead of all comparisons with a single reference category can quickly lead to a very large number of estimates. Third, this method is not the only way of attaining estimates of direct and indirect effects; there are, for instance, the methods proposed by Gomulka and Stern (1990), Even and Macpherson (1990), Fairlie (2005), Yun (2004), and Bauer and Sinning (2008). Two of these have been implemented in Stata: the method by Fairlie (2005) in the `fairlie` command (Jann 2006), and the method by Bauer and Sinning (2008) in the `nldecompose` command (Sinning, Hahn, and Bauer 2008). How these alternatives compare with the method discussed in this article needs to be explored.

5 Acknowledgments

I thank the members of the Comparative Stratification Research Seminar at the Vrije Universiteit Amsterdam, Michelle Jackson, William Gould, and an anonymous referee for their helpful comments.

6 References

Bartels, L. M. 2002. Beyond the running tally: Partisan bias in political perceptions. *Political Behavior* 24: 117–150.

Bartus, T. 2005. Estimation of marginal effects using margeff. *Stata Journal* 5: 309–329.

Bauer, T. K., and M. Sinning. 2008. An extension of the Blinder–Oaxaca decomposition to nonlinear models. *Advances in Statistical Analysis* 92: 197–206.

Boudon, R. 1974. *Education, Opportunity and Social Inequality Prospects in Western Society*. New York: Wiley.

Buis, M. L. 2007a. Predict and adjust with logistic regression. *Stata Journal* 7: 221–226.

———. 2007b. Stata tip 48: Discrete uses for uniform(). *Stata Journal* 7: 434–435.

Campbell, A., P. E. Converse, W. E. Miller, and D. E. Stokes. 1960. *The American Voter*. New York: Wiley.

Efron, B., and R. J. Tibshirani. 1993. *An Introduction to the Bootstrap*. New York: Chapman & Hall/CRC.

Erikson, R., J. H. Goldthorpe, M. Jackson, M. Yaish, and D. R. Cox. 2005. On class differentials in educational attainment. *Proceedings of the National Academy of Science* 102: 9730–9733.

Even, W. E., and D. A. Macpherson. 1990. Plant size and the decline of unionism. *Economics Letters* 32: 393–398.

Fairlie, R. W. 2005. An extension of the Blinder–Oaxaca decomposition technique to logit and probit models. *Journal of Economic and Social Measurement* 30: 305–316.

Gomulka, J., and N. Stern. 1990. The employment of married women in the United Kingdom 1970–83. *Economica* 57: 171–199.

Hauser, R. M., and W. H. Sewell. 1957–1977. *Wisconsin Longitudinal Study (*WLS*) [graduates and siblings]: 1957–1977*. Madison, WI: University of Wisconsin–Madison. http://www.ssc.wisc.edu/∼wls/documentation/.

Jackson, M., R. Erikson, J. H. Goldthorpe, and M. Yaish. 2007. Primary and secondary effects in class differentials in educational attainment: The transition to A-level courses in England and Wales. *Acta Sociologica* 50: 211–229.

Jann, B. 2006. fairlie: Stata module to generate nonlinear decomposition of binary outcome differentials. http://ideas.repec.org/c/boc/bocode/s456727.html.

Sinning, M., M. Hahn, and T. K. Bauer. 2008. The Blinder–Oaxaca decomposition for nonlinear regression models. *Stata Journal* 8: 480–492.

Yun, M.-S. 2004. Decomposing differences in the first moment. *Economics Letters* 82: 275–280.

About the author

Maarten L. Buis is affiliated with the Department of Sociology at the University Tübingen, where he studies the interaction between demographic processes and inequality of educational opportunity. He is also a frequent contributor to Statalist.

The Stata Journal (2010)
10, Number 1, pp. 30–45

Using the world development indicators database for statistical analysis in Stata

P. Wilner Jeanty
Department of Agricultural, Environmental, and Development Economics
The Ohio State University
Columbus, OH
jeanty.1@osu.edu

Abstract. The World Bank's world development indicators (WDI) compilation is a rich and widely used database about the development of most economies in the world. However, after insheeting a WDI dataset, some data management is required prior to performing statistical analysis. In this article, I propose a new Stata command, `wdireshape`, for automating this data management. While reshaping a WDI dataset into structures amenable to panel data, seeming unrelated regression, or cross-sectional modeling, `wdireshape` renames the series and places the series descriptors into variable labels.

Keywords: dm0045, wdireshape, paverage, world development indicators, reshape, panel data, seeming unrelated regression

1 Introduction

The World Bank's world development indicators (WDI) compilation is a rich and widely used database about the development of most countries in the world, with time series going back to 1960. The 2009 WDI compilation consists of more than 800 indicators in over 90 tables organized in 6 sections, including world view, people, environment, economy, states and markets, and global links (The World Bank Group, 2009). The WDI database is available online for a paid subscription or on a CD-ROM as a purchase. However, after insheeting a WDI dataset, some data management is required prior to performing statistical analysis. Further, while the World Bank has made great strides in rendering WDI in several forms for download, seemingly unrelated regressions analysis, for example, cannot be carried out using any of such forms. The new Stata command `wdireshape`, presented in this article, automates the data management required to get the data ready for statistical analysis. In particular, `wdireshape` allows you to obtain data structures amenable to panel data, seeming unrelated regression, or cross-sectional modeling.

The next section presents the `wdireshape` command, and section 3 expands on data preparation and how to get WDI data into Stata. Section 4 outlines `wdireshape` syntax and options, while examples are presented in section 5. Section 6 concludes the article.

2 The wdireshape command

`wdireshape` reshapes a Stata dataset obtained by insheeting a text (`.csv`) file downloaded from the WDI World Bank web site or extracted from the WDI CD-ROM. Depending on the option specified with `wdireshape`, the new dataset has a structure suitable for panel-data analysis, seemingly unrelated regression (SUR), or cross-sectional modeling. The panel-data structure is known as long form, and the SUR and cross-sectional structures are known as wide form. During the reshaping process, `wdireshape` places the WDI series descriptors into Stata variable labels and enables users to supply, in a varlist, names of their devising for the WDI series.

3 Data preparation

Before extracting a `.csv` file from the WDI web site or a recent CD release, users must choose a data orientation with series or country in rows and time in columns. The `.csv` file can be imported into Stata by typing

```
. insheet using filename.csv, names clear
```

`wdireshape` works on this insheeted dataset. Older CD releases, such as the WDI-2005 CD-ROM, produce `.csv` files that must be managed prior to insheeting. In particular, the years must be prepended with a letter, which can be done in a spreadsheet or by using the procedure suggested in Baum and Cox (2007).

4 Commands

4.1 Syntax for wdireshape

The syntax to check the number of indicators and their order of appearance in a WDI dataset is

`wdireshape, sername(`*varname*`)`

The syntax to reshape the dataset is

`wdireshape` *newvarlist*`, prepend(`*letter(s)*`) ctyname(`*varname*`) sername(`*varname*`)`
`ctycode(`*varname*`) sercode(`*varname*`) [byper(#) startyr(#) endyr(#)`
`byvar sur cros nstring(#)]`

(*Continued on next page*)

4.2 Options for wdireshape

Required options

`prepend(`*letter(s)*`)` specifies the letters with which the years are prepended. One or two letters from `a` to `z` should be used. When insheeting a WDI dataset downloaded from the World Bank's web site, the years are prepended with "`yr`".

`ctyname(`*varname*`)` specifies the variable containing the country names.

`sername(`*varname*`)` specifies the variable holding the series names.

`ctycode(`*varname*`)` specifies the variable containing the country code elements.

`sercode(`*varname*`)` specifies the variable containing the series code elements.

With these five required options specified, `wdireshape` will attempt to reshape the entire dataset at once, which is the default. Due to memory issues, reshaping large datasets at once may not be successful. In such a case, Stata will prompt the user to specify the `byvar` or `byper(#)` option, or to increase the amount of memory allocated to Stata. You can reset the size of memory only if you are using Stata/MP, Stata/SE, or Stata/IC.

Optional options

`byper(#)` requires `wdireshape` to reshape the dataset 1 year, 5 years, or 10 years at a time, as long as the time span contains no gaps. One of these three values should be used with the `byper(#)` option. If either 5 or 10 is specified, `wdireshape` will account for the fact that the last subperiod may not be of 5 or 10 years. Also, Stata will check whether the current memory size is enough to reshape the data 5 or 10 years at a time.

`startyr(#)` specifies the first year of the time period.

`endyr(#)` specifies the last year of the time period.

Note 1: The `byper(#)`, `startyr(#)`, and `endyr(#)` options must be combined.

`byvar` specifies that the dataset be reshaped one variable at a time, as proposed by Kossinets (2006). The `byvar` option may not be combined with `byper(#)`, `startyr(#)`, and `endyr(#)`.

`sur` requests a wide form suitable for (SUR) analysis (see [R] **sureg**). By default, the dataset is reshaped in long form for panel-data analysis (see [XT] **xtreg**). When this option is specified, in the reshaped dataset, the country names are postfixed to the user-supplied variable names and are represented by `c1`, `c2`, etc. `describe` the reshaped dataset if you want to know which countries `c1`, `c2`, ..., `c`*n* represent. In Stata 10 or higher, you can just look at the variable labels in the variable window. The SUR-reshaped structure displays the years in rows and the variables, for each country, in columns.

`cros` requests a wide form suitable for cross-sectional analysis. The `cros`-reshaped structure displays the country names in rows and the variables, for each year, in columns. Obviously, the `cros` option may not be combined with `sur`.

Note 2: When the `sur` or `cros` option is specified, Stata will complain if the resulting number of variables exceeds its limits, which are 99 for Small Stata, 2,047 for Stata/IC, and 32,767 for Stata/MP and Stata/SE.

`nstring(#)` indicates that the dataset contains the WDI missing-value symbols, the double dots (`..`), and that they should be removed from the reshaped dataset. # represents the number of identifier variables in the dataset. For example, `nstring(4)` must be specified when the dataset includes names and code elements for both countries and series as identifier variables. When the `nstring(#)` option is specified, if an error occurs for any reason, the dataset to be reshaped needs to be reloaded before running `wdireshape` again. Otherwise, Stata will abort with a type-mismatch error.

Note 3: In the case of large datasets, reshaping 10 years at a time—as long as the time period is at least 10 years and there is enough memory—would be much faster than reshaping variable by variable. However, when the time period contains gaps, `byper(#)` will not work.

4.3 Syntax for paverage

The syntax to calculate *p*-year averages of the variables in a panel dataset is

`paverage` *varlist*`, p(#)` `indiv(`*varname*`)` `yr(`*varname*`)`

4.4 Description of paverage

`paverage` (pronounced p-average) calculates series of averages in a panel dataset. In the process, the labels of the original variables, if present, are attached to the average variables. The time period must be a multiple of the subperiod over which averages need to be calculated. When analyzing a panel dataset with a long time period, using series of averages is a common way to reduce business-cycle effects and measurement error.

4.5 Options for paverage

`p(#)` indicates the number of years for which averages need to be calculated. # ranges from 2 to 10. For example, specifying `p(5)` will create a five-year average dataset.

`indiv(`*varname*`)` specifies the variable name containing the country names, individuals, or firms.

`yr(`*varname*`)` specifies the variable name holding the years.

5 Examples

This section illustrates `wdireshape`, and all examples use data downloaded from the WDI web site.

5.1 Example 1: Reshape everything at once for panel-data analysis

To import the data, type

```
. set memory 20m
(20480k)
. insheet using wdi_exmpl1.csv, names clear
(46 vars, 577 obs)
```

I take a look at the raw dataset by listing a few observations for three countries and two years of the time period, 1961 and 2002.

```
. list countryname seriesname yr1961 yr2002 in 1/13, sepby(countryname)
> string(30) noobs
```

countryname	seriesname	yr1961	yr2002
Afghanistan	Foreign direct investment, net..	..	..
Afghanistan	GDP (current US$)	1240000000	4040000000
Afghanistan	Trade (% of GDP)	12.55061	..
Algeria	Foreign direct investment, net..	..	1.904728
Algeria	Foreign direct investment, net..	..	1070000000
Algeria	Foreign direct investment, net..	..	..
Algeria	GDP (current US$)	2430000000	55900000000
Algeria	Trade (% of GDP)	113.7483	61.70864
Angola	Foreign direct investment, net..	..	15.43182
Angola	Foreign direct investment, net..	..	1670000000
Angola	Foreign direct investment, net..	..	0.2648883
Angola	GDP (current US$)	..	10800000000
Angola	Trade (% of GDP)	..	143.2107

Prior to reshaping the data, I use the first syntax of `wdireshape` to rigorously check the number of variables (series/indicators) and their order of appearance. I recommend doing this check before reshaping the data.

```
. wdireshape, sername(seriesname)
 The current dataset contains 5 variables in the following order:
 1) Foreign direct investment, net inflows (% of GDP)
 2) Foreign direct investment, net inflows (BoP, current US$)
 3) Foreign direct investment, net outflows (% of GDP)
 4) GDP (current US$)
 5) Trade (% of GDP)
```

Supplying five new variable names that befit the series descriptors, I now use the second syntax of `wdireshape` to reshape the data. I specify `nstring(4)`[1] to remove the WDI missing-value symbols, the double dots (`..`), from the reshaped dataset. Removing the double dots is paramount to perform numerical operations on the reshaped dataset: Stata will regard as string data any columns containing them.

```
. wdireshape fdingdp fdincur fdiout gdp trade,  prepend(yr) ctyname(countryname)
> sername(seriesname) sercode(seriescode) ctycode(country_code) nstring(4)
Reshaping your dataset (everything at once), please wait
Your dataset has been reshaped and is ready for panel data analysis
```

As output, Stata first announces that the entire dataset is being reshaped at once. When the conversion process is complete, another message asserts that the dataset has been reshaped and is ready for panel-data analysis, which is the default.

I use `describe` (see [D] **describe**) to get an overview of the reshaped dataset:

```
. describe
Contains data
  obs:         5,166
 vars:             9
 size:       428,778 (98.0% of memory free)
```

variable name	storage type	display format	value label	variable label
cid	float	%9.0g		Country ID
countrycode	str3	%9s		Country code
countryname	str30	%30s		Country name
year	int	%9.0g		
fdingdp	double	%12.0g		Foreign direct investment, net inflows (% of GDP)
fdincur	double	%12.0g		Foreign direct investment, net inflows (BoP, current US$)
fdiout	double	%12.0g		Foreign direct investment, net outflows (% of GDP)
gdp	double	%12.0g		GDP (current US$)
trade	double	%12.0g		Trade (% of GDP)

```
Sorted by:  cid  year
     Note:  dataset has changed since last saved
```

1. If the `nstring(`$\#$`)` option is specified and an error occurs for any reason (e.g., invalid syntax), the raw dataset to be reshaped needs to be reloaded before running `wdireshape` again. Otherwise, Stata will abort with a type-mismatch error.

Another look is provided by `list`ing the first few observations:

```
. list countryname year fdingdp fdincur fdiout gdp trade in 1/10
```

	countryname	year	fdingdp	fdincur	fdiout	gdp	trade
1.	Afghanistan	1961	.	.	.	1.240e+09	12.55061
2.	Afghanistan	1962	.	.	.	1.230e+09	14.22764
3.	Afghanistan	1963	.	.	.	9.610e+08	26.03551
4.	Afghanistan	1964	.	.	.	8.000e+08	26.94445
5.	Afghanistan	1965	.	.	.	1.010e+09	32.67108
6.	Afghanistan	1966	.	.	.	1.400e+09	27.14286
7.	Afghanistan	1967	.	.	.	1.670e+09	20.98273
8.	Afghanistan	1968	.	.	.	1.370e+09	24.11003
9.	Afghanistan	1969	.	.	.	1.410e+09	25.07887
10.	Afghanistan	1970	.	.	.	1.750e+09	21.72811

Panel-data analysis

I now use the Stata `xtsum` command to show that the data are xt ready (see [XT] **xt**).

```
. xtsum fdingdp fdincur fdiout gdp trade
```

Variable		Mean	Std. Dev.	Min	Max	Observations
fdingdp	overall	1.858811	4.390006	-82.81054	72.32173	N = 3136
	between		2.099348	-.0070139	10.36776	n = 110
	within		3.924819	-85.64466	69.48761	T-bar = 28.5091
fdincur	overall	4.19e+08	2.61e+09	-4.55e+09	4.93e+10	N = 3642
	between		1.48e+09	-4375758	1.30e+10	n = 111
	within		2.15e+09	-1.26e+10	3.67e+10	T-bar = 32.8108
fdiout	overall	.1266303	.5728872	-7.195931	7.155485	N = 2474
	between		.2665292	-.0696363	2.175378	n = 114
	within		.4976684	-7.423015	6.91092	T-bar = 21.7018
gdp	overall	2.41e+10	8.02e+10	1.16e+07	1.45e+12	N = 4171
	between		5.29e+10	4.35e+07	3.64e+11	n = 121
	within		5.63e+10	-2.94e+11	1.11e+12	T-bar = 34.4711
trade	overall	67.14495	38.96681	1.530677	228.8752	N = 3985
	between		35.73635	15.20354	163.0827	n = 121
	within		17.71395	-18.7896	174.2657	T-bar = 32.9339

`xtsum` provides much more information than `summarize`. For example, the values for n in the "Observations" column indicate the number of countries for which data are available on each variable. Because the data are xt ready, panel-data models can be fit using the Stata `xtreg` command (see [XT] **xtreg**).

Time series of averages across countries

Time series of averages across countries can be obtained using the Stata `collapse` command.

```
. collapse fdingdp fdincur fdiout gdp trade, by(year) cw // mean is the default
```

Now suppose that I want to graph the time series of averages for investment (net inflows and outflows). I type the following lines of code to produce figure 1. Here I use two y axes because the two variables are of different scales.[2]

```
. set scheme sj
. label variable fdingdp "Investment, net inflows (% of GDP)"
. label variable fdiout "investment, net outflows (% of GDP)"
. scatter fdingdp year, s(t) c(l) yaxis(1)  ||
> scatter fdiout year, s(D) c(l) yaxis(2) legend(cols(1)) xlab(1970(5)2002)
```

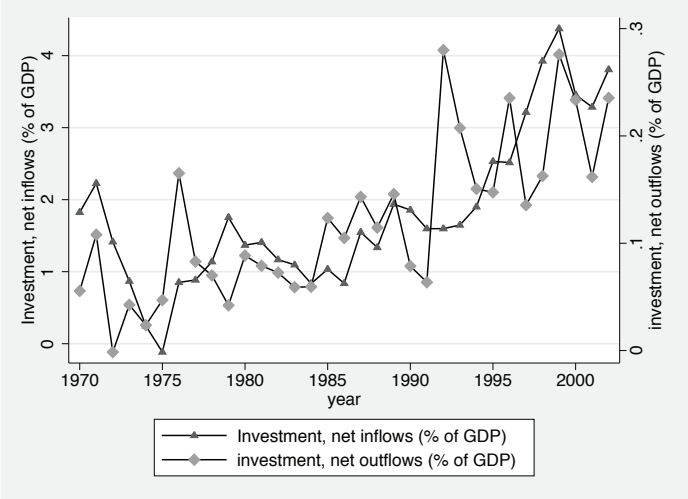

Figure 1. Time series of averages of investment (net inflows and outflows)

Country averages across years

To obtain country averages across years, I load and reshape the dataset one more time. But this time, I elect to reshape one variable at a time by specifying the `byvar` option.

2. Issuing the `tsline fdingdp fdiout` command would do the job, but scales are not taken into account.

```
. insheet using wdi_exmpl1.csv, names clear
(46 vars, 577 obs)
. wdireshape fdingdp fdincur fdiout gdp trade, prepend(yr) ctyname(countryname)
> sername(seriesname) sercode(seriescode) ctycode(countrycode) byvar nstring(4)
Reshaping your dataset one variable at a time, please wait
Your dataset has been reshaped and is ready for panel data analysis
```

I now run the Stata `collapse` command to obtain country averages across years.

```
. collapse  fdingdp fdincur fdiout gdp trade, by(countryname) cw
```

Now suppose that I want to know which countries have their annual average net outflow investment between 0.1 and 0.5 percent, given their annual average net inflow investment. The names of these countries are displayed in figure 2, which I obtained by typing the following:

```
. label var fdingdp "Investment, net inflows (% of GDP)"
. label var fdiout "investment, net outflows (% of GDP)"
. generate pos=3
. replace pos=6 if countryname== "South Africa"
(1 real change made)
. replace pos=9 if countryname=="Niger"
(1 real change made)
. scatter fdingdp fdiout  if fdiout>=.1 & fdiout<=.5, mlabel(countryname)
> xscale(log) xscale(range(.6)) mlabv(pos) yscale(range(-1))
```

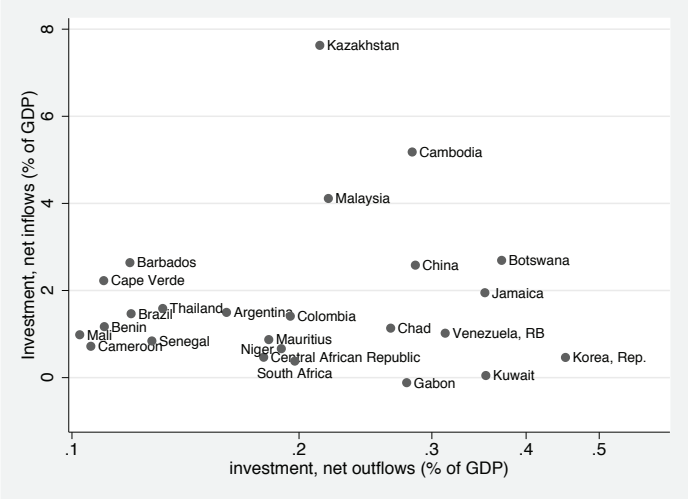

Figure 2. Countries with annual average net outflow investment between 0.1 and 0.5 percent of gross domestic product (GDP)

If the time period is a multiple of 5 or 10, 5-year or 10-year averages can be calculated using the `paverage` command described in section 4.3. `paverage` is available from the

Statistical Software Components archive. Baum (2006) also illustrates some tools for operating on and manipulating panel data.

5.2 Example 2: Reshape five years at a time for SUR analysis

To import the data and check the number of indicators and their order of appearance, I type

```
. insheet using wdi_country_time.csv, names clear
(50 vars, 1157 obs)

. wdireshape, sername(seriesname)

 The current dataset contains 13 variables in the following order:

 1) Agricultural land (% of land area)
 2) Agricultural machinery, tractors per 100 sq. km of arable land
 3) Fertilizer consumption (100 grams per hectare of arable land)
 4) GDP (constant 2000 US$)
 5) GDP (current US$)
 6) GDP per capita growth (annual %)
 7) Irrigated land (% of cropland)
 8) Permanent cropland (% of land area)
 9) Population density (people per sq. km)
 10) Population growth (annual %)
 11) Rural population density (rural population per sq. km of arable land)
 12) Trade (% of GDP)
 13) Urban population growth (annual %)
```

I choose to reshape the data five years at a time and invoke the `sur` option to request a structure amenable to SUR analysis. Although a memory size of four megabytes is enough to load the data, a memory error message would have occurred had I not specified `byp(5)`.

```
. wdireshape agland tractsk fertilha gdpcnst gdpcur gdppg irrigpct croplnd
> popdens popg ruraldens trade urbpg, prepend(yr) ctyname(countryname)
> sername(seriesname) sercode(seriescode) ctycode(countrycode) startyr(1961)
> endyr(2006) byper(5) sur nstring(4)

Reshaping your dataset 5 years at a time
Now reshaping period  1961 - 1965
Now reshaping period  1966 - 1970
Now reshaping period  1971 - 1975
Now reshaping period  1976 - 1980
Now reshaping period  1981 - 1985
Now reshaping period  1986 - 1990
Now reshaping period  1991 - 1995
Now reshaping period  1996 - 2000
Now reshaping period  2001 - 2006

Your dataset has been reshaped and is ready for SUREG analysis
```

The dataset reshaped for SUR contains 1,158 variables, because the raw dataset had 13 variables and 89 countries. As mentioned above, this structure displays the years in rows and the variables, for each country, in columns. I now display a few observations on the variables `year` and GDP per capita growth pertaining to four Latin American (LA) countries.

```
. list year gdppgc11 gdppgc9 gdppgc32 gdppgc60 in 1/5
```

	year	gdppgc11	gdppgc9	gdppgc32	gdppgc60
1.	1961	7.0261211	-.15207509	1.5345942	4.0316689
2.	1962	2.0897761	3.2527942	.78809956	8.1733652
3.	1963	-2.107187	4.0718175	6.630273	6.466808
4.	1964	.49288856	2.464101	1.852948	7.5981292
5.	1965	.17106316	3.2549077	1.5911081	5.5632783

As should be emphasized, the country names are concatenated to the user-supplied variable names and are represented by `c1`, `c2`, `c3`, and so on. You can decipher them by typing `describe` or by looking at the variable labels from the variable window if you are using Stata 10 or higher. I now `describe` the variable GDP per capita growth for the four LA countries.

```
. describe gdppgc11 gdppgc9 gdppgc32 gdppgc60
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
gdppgc11        double %10.0g                 GDP per capita growth (annual %)
                                                - Brazil
gdppgc9         double %10.0g                 GDP per capita growth (annual %)
                                                - Bolivia
gdppgc32        double %10.0g                 GDP per capita growth (annual %)
                                                - Guatemala
gdppgc60        double %10.0g                 GDP per capita growth (annual %)
                                                - Nicaragua
```

To show that the reshaped dataset is ready for SUR analysis, I estimate four SURs, one for each LA country. Given the data at hand, I want to investigate whether the agricultural sector contributes to the GDP per capita growth in these countries. First, I `describe` for the first country, which is Brazil, the other variables used in the analysis. Then I estimate the regressions with the Stata `sureg` command. Here I specify the `dfk` and `corr` options to request a small-sample adjustment and the Breusch–Pagan test for independent equations.

```
. describe tractskc11 fertilhac11 croplndc11 popgc11
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
tractskc11      double %10.0g                 Agricultural machinery, tractors
                                                per 100 sq. km of arable land -
                                                Brazil
fertilhac11     double %10.0g                 Fertilizer consumption (100 grams
                                                per hectare of arable land) -
                                                Brazil
croplndc11      double %10.0g                 Permanent cropland (% of land
                                                area) - Brazil
popgc11         double %10.0g                 Population growth (annual %) -
                                                Brazil
```

```
. foreach num of numlist  11 9 32 60 {
  2.    local eqn "`eqn´ (gdppgc`num´ L.gdppgc`num´ tractskc`num´ fertilhac`num´
> croplndc`num´ popgc`num´) "
  3. }
. sureg `eqn´, dfk corr
Seemingly unrelated regression
```

Equation	Obs	Parms	RMSE	"R-sq"	chi2	P
gdppgc11	41	5	3.083649	0.4231	24.85	0.0001
gdppgc9	41	5	3.174336	0.2522	16.53	0.0055
gdppgc32	41	5	1.719363	0.5391	41.80	0.0000
gdppgc60	41	5	5.245668	0.3524	23.91	0.0002

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
gdppgc11						
gdppgc11						
L1.	.2014618	.1446664	1.39	0.164	-.0820791	.4850028
tractskc11	-.0842038	.0301914	-2.79	0.005	-.1433779	-.0250297
fertilhac11	.009484	.0040085	2.37	0.018	.0016275	.0173405
croplndc11	-20.48911	10.52829	-1.95	0.052	-41.12418	.1459484
popgc11	-1.500581	2.382403	-0.63	0.529	-6.170006	3.168843
_cons	24.21315	11.5525	2.10	0.036	1.570678	46.85563
gdppgc9						
gdppgc9						
L1.	.0428409	.1508581	0.28	0.776	-.2528357	.3385174
tractskc9	-.1529211	.1650501	-0.93	0.354	-.4764134	.1705712
fertilhac9	-.0086263	.0561642	-0.15	0.878	-.1187061	.1014536
croplndc9	78.50515	30.84602	2.55	0.011	18.04806	138.9622
popgc9	17.4214	5.810168	3.00	0.003	6.033678	28.80912
_cons	-46.62524	16.28789	-2.86	0.004	-78.54892	-14.70157
gdppgc32						
gdppgc32						
L1.	.3834769	.1242297	3.09	0.002	.1399912	.6269626
tractskc32	-.318205	.1216817	-2.62	0.009	-.5566968	-.0797131
fertilhac32	.0112325	.0029539	3.80	0.000	.005443	.017022
croplndc32	-6.652823	1.92693	-3.45	0.001	-10.42954	-2.876109
popgc32	11.52029	4.294384	2.68	0.007	3.103453	19.93713
_cons	3.227747	11.92218	0.27	0.787	-20.1393	26.59479
gdppgc60						
gdppgc60						
L1.	-.1455299	.1551685	-0.94	0.348	-.4496546	.1585947
tractskc60	-.9258023	.2191462	-4.22	0.000	-1.355321	-.4962838
fertilhac60	.0129039	.0078384	1.65	0.100	-.0024592	.0282669
croplndc60	36.16019	12.29522	2.94	0.003	12.06201	60.25838
popgc60	7.464455	4.188569	1.78	0.075	-.7449899	15.6739
_cons	-67.79043	29.24217	-2.32	0.020	-125.104	-10.47684

(*Continued on next page*)

```
Correlation matrix of residuals:

           gdppgc11   gdppgc9  gdppgc32  gdppgc60
gdppgc11    1.0000
 gdppgc9   -0.3330    1.0000
gdppgc32    0.0860   -0.0576    1.0000
gdppgc60   -0.2389    0.1804   -0.2081    1.0000

Breusch-Pagan test of independence: chi2(6) =    10.436, Pr = 0.1074
```

As the results indicate, the null hypothesis that the coefficients in each respective regression are zero is rejected at the one percent significance level. However, the Breusch–Pagan test fails to reject at the conventional significance level the null hypothesis that the correlation of the residuals across equations is zero.

5.3 Example 3: Reshape 10 years at a time for cross-sectional analysis

In this example, the same dataset used in example 2 is reshaped 10 years at a time. I specify the `cros` option to request a structure appropriate for cross-sectional analysis.

```
. drop _all

. quietly set memory 2m

. insheet using wdi_country_time.csv, names
(50 vars, 1157 obs)

. wdireshape agland tractsk fertilha gdpcnst gdpcur gdppg irrigpct croplnd
> popdens popg ruraldens trade urbpg, prepend(yr) ctyname(countryname)
> sername(seriesname) sercode(seriescode) ctycode(countrycode) start(1961)
> end(2006) byp(10) cros nstring(4)

Reshaping your dataset 10 years at a time
Now reshaping period  1961 - 1970
Now reshaping period  1971 - 1980
Now reshaping period  1981 - 1990
Now reshaping period  1991 - 2000
Now reshaping period  2001 - 2006

Your dataset has been reshaped and is ready for cross sectional or change analysis
```

As can be seen, `wdireshape` accounts for the fact that the subperiod 2001–2006 is not of 10 years. Because the raw dataset consists of 13 variables observed on a period of 46 years, the reshaped dataset contains at least a total of 598 variables. Recall that the `cros`-reshaped structure places the country names in rows and the variables, for each year, in columns. For the years 1961 and 2006, I `list` a few observations on the variables population density (`popdens`) and urban population growth (`urbpg`):

```
. list countryname popdens1961 popdens2006 urbpg1961 urbpg2006 in 1/10
```

	countryname	popd~1961	popd~2006	urbpg1961	urbpg2006
1.	Afghanistan	15.623018	.	5.6376534	5.2102131
2.	Algeria	4.6212609	14.001507	6.4459141	2.4988708
3.	Angola	4.0977862	13.147816	5.8804117	4.0143405
4.	Argentina	7.6554608	14.294807	2.3680321	1.1362566
5.	Bangladesh	402.52815	1108.8965	6.4879591	3.5009976
6.	Barbados	538.07209	628.00706	.60466256	1.3126057
7.	Belize	4.1316061	13.030856	2.8929328	2.2994995
8.	Benin	21.300515	78.586803	8.3754272	3.9628651
9.	Bolivia	3.1597392	8.617589	3.0227506	2.470662
10.	Botswana	1.0341344	3.1018031	6.8385594	.9156059

I now `describe` these variables:

```
. describe countryname popdens1961 popdens2006 urbpg1961 urbpg2006
```

variable name	storage type	display format	value label	variable label
countryname	str24	%24s		Country name
popdens1961	double	%10.0g		1961 - Population density (people per sq. km)
popdens2006	double	%10.0g		2006 - Population density (people per sq. km)
urbpg1961	double	%10.0g		1961 - Urban population growth (annual %)
urbpg2006	double	%10.0g		2006 - Urban population growth (annual %)

Now if I want to calculate change in population density from 1961 to 2006, I type

```
. generate popdens_ch = popdens2006 - popdens1961
(2 missing values generated)
```

In summary, `wdireshape` helps reduce data-management tasks when WDI users need to

- Conduct a panel-data analysis.

  Run `wdireshape` without specifying the `sur` or `cros` option.

- Analyze a time series of averages across countries.

  First, run `wdireshape`, and then run the Stata `collapse` command by the variable containing the years.

- Analyze a series of averages across years (pure cross-sections).

  First, run `wdireshape`, and then run the Stata `collapse` command by the variable holding the country names.

- Analyze a series of p-year averages (with $p = 2, 3, \ldots, 10$).

  First, run `wdireshape`, and then run `paverage`, available from the Statistical Software Components archive.

- Conduct a SUR analysis.

  Run `wdireshape` with the `sur` option.

- Perform a change or cross-sectional analysis.

  Run `wdireshape` with the `cros` option.

- Operate on and manipulate WDI as a panel dataset of countries.

  Run `wdireshape`, and then apply Baum's 2006 suggestions provided in chapter 3.

6 Conclusions

In this article, I introduced a new Stata command, `wdireshape`, enabling Stata users to efficiently manage WDI datasets. While allowing users to supply variable names of their choosing for the series, `wdireshape` reshapes the data for panel data, SUR, or cross-sectional modeling. In the process, the WDI series descriptors are placed into Stata variable labels.

7 Acknowledgments

I thank Christopher Baum, the participants of the 2008 Summer North American Stata Users Group meeting in Chicago, and an anonymous reviewer for useful comments and suggestions. This work was partly funded by the grant "Conflict, Poverty, and Environmental and Food Security" from the Mershon Center at The Ohio State University. Support from the Swank Program is also greatly acknowledged.

Inspiration for the `byvar` option comes from Stata code posted on the web by Kossinets (2006).

8 References

Baum, C. F. 2006. *An Introduction to Modern Econometrics Using Stata*. College Station, TX: Stata Press.

Baum, C. F., and N. J. Cox. 2007. Stata tip 45: Getting those data into shape. *Stata Journal* 7: 268–271.

Kossinets, G. 2006. Code to reshape a WDI dataset in Stata.
http://www.columbia.edu/acis/eds/data_tools/tips/reshape_manyvar.do.

The World Bank Group. 2009. World Development Indicators (WDI) Online.
http://publications.worldbank.org/WDI/.

About the author

P. Wilner Jeanty is a postdoctoral scholar in the Department of Agricultural, Environmental, and Development Economics at The Ohio State University.

The Stata Journal (2010)
10, Number 1, pp. 46–60

Tabulating SPost results using estout and esttab

Ben Jann
ETH Zürich
Zürich, Switzerland
jann@soz.gess.ethz.ch

J. Scott Long
Indiana University
Bloomington, IN
jslong@indiana.edu

Abstract. The `SPost` user package (Long and Freese, 2006, *Regression Models for Categorical Dependent Variables Using Stata* [Stata Press]) is a suite of postestimation commands to compute additional tests and effects representations for a variety of regression models. To facilitate and automate the task of tabulating results from `SPost` commands for inclusion in reports, publications, and presentations, we introduce tools to integrate `SPost` with the `estout` user package (Jann, 2005, *Stata Journal* 5: 288–308; 2007, *Stata Journal* 7: 227–244). The `estadd` command can retrieve results computed by the `SPost` commands `brant`, `fitstat`, `listcoef`, `mlogtest`, `prchange`, `prvalue`, and `asprvalue`. These results can then be tabulated by `esttab` or `estout`.

Keywords: st0183, SPost, regression table, estadd, estout, esttab, brant, fitstat, listcoef, mlogtest, prchange, prvalue, asprvalue

1 Introduction

The detailed interpretation of regression models often requires the incorporation of information that goes beyond the standard regression coefficients and reported tests. For example, the interpretation of an ordered logit model might include odds ratios, standardized odds coefficients, predicted probabilities for each outcome, and the results of the Brant test. Similarly, the selection among a series of count models might involve comparison of Bayesian information criterion statistics from competing models. While official Stata and user-written commands provide tables, these tables are generic and are rarely in the specific form an analyst wants for presentations. In this article, we demonstrate extensions of the `estout` and `SPost` packages that allow you to use `estout` and `esttab` to produce professional tables that combine the results from estimation and postestimation commands.

In this article, we assume that the reader is familiar with the `SPost` and `estout` packages. While readers who are unfamiliar with these packages should be able to follow the examples we provide, to fully utilize these new features we suggest that you read Jann (2005, 2007) and Long and Freese (2006). We also suggest consulting the `estout` web site, http://repec.org/bocode/e/estout—which contains dozens of examples that combine `SPost` commands with `estout` and `esttab`—and the `SPost` web site, http://www.indiana.edu/~jslsoc/spost.htm. For a more theoretical discussion of many of the models and tests used, see Long (1997).

Our extensions to the estout and SPost packages add features that allow moving computed results from the SPost commands brant, fitstat, listcoef, mlogtest, prchange, prvalue, and asprvalue to estout or esttab tables. Supported regression models include asmprobit, clogit, cloglog, cnreg, intreg, logit, mlogit, mprobit, nbreg, ologit, oprobit, poisson, probit, regress, rologit, slogit, tobit, zinb, zip, ztnb, and ztp, although not all commands are applicable for all models (see Long and Freese [2006] for details on these models).[1]

To use the new features, you need to install the latest versions of the estout and SPost packages. estout can be obtained from the Statistical Software Components Archive at Boston College. Type

```
. ssc install estout, replace
```

to install the package. The SPost software is available from J. Scott Long's web site. To locate and install the package, type

```
. findit spost9_ado
```

and follow the prompts that you will be given. Alternatively, type

```
. net install spost9_ado, from(http://www.indiana.edu/~jslsoc/stata) replace
```

The new features are only available with SPost for Stata 9 and later (spost9_ado). SPost for Stata 8 (spostado) is not supported.

2 Syntax and examples

The general procedure to tabulate results from an SPost command in estout or esttab is to fit a model with a Stata regression command and then run the SPost command to obtain additional postestimation results. estadd combines the SPost command's results with the model's e() returns. After that, the results are accessible to estout or esttab via the cells() or the stats() option, depending on whether the results are added in a matrix or as scalars. Alternatively, the main(), aux(), and scalars() options can be used in esttab to access the results, although cells() and stats() are more general and flexible, albeit more complex. We begin with two examples that illustrate the potential of combining SPost results with estout. We then provide details on the estadd syntax for each of the supported SPost commands and provide further examples. An extensive set of additional examples can be found at estout's web site (see http://repec.org/bocode/e/estout/spost.html).

As a simple example, suppose that you want to tabulate information measures computed by fitstat for a linear regression model fit by regress. You could type

1. Stata 11 factor variables are not (yet) supported by the SPost commands.

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/regjob3
(Long´s data on academic jobs of biochemists \ 2009-03-13)
. regress job fem phd ment fel art cit
  (output omitted)
. estadd fitstat, bic
AIC:                            2.580   AIC*n:                       1052.793
BIC:                       -1371.725   BIC´:                         -60.162
BIC used by Stata:          1080.872   AIC used by Stata:           1052.793
added scalars:
               e(aic0) =  2.5803757
              e(aic_n) =  1052.7933
               e(bic0) =  -1371.7248
              e(bic_p) =  -60.162312
          e(statabic) =  1080.8722
          e(stataaic) =  1052.7933
. esttab, cells(none) scalars(aic0 aic_n bic0 bic_p)
──────────────────────────
                      (1)
                      job
──────────────────────────
N                     408
aic0                2.580
aic_n              1052.8
bic0              -1371.7
bic_p              -60.16
──────────────────────────
```

Notice that we used the `cells(none)` option to suppress the regression coefficients. To customize the labels, you can revise the `esttab` command, for example, as follows:

```
. esttab, cells(none)
>     scalars("aic0 AIC" "aic_n AIC*n" "bic0 BIC" "bic_p BIC´")
──────────────────────────
                      (1)
                      job
──────────────────────────
N                     408
AIC                 2.580
AIC*n              1052.8
BIC               -1371.7
BIC´               -60.16
──────────────────────────
```

If you are working with multiple models, you can either add results to each model individually after estimation as above or first fit and store a set of models and then apply `estadd` to all of them in one call by using the colon syntax. Here is an example of the latter, using `eststo` (which is also part of the `estout` package) to store the models:

```
. eststo: quietly regress job fem phd ment
(est1 stored)
. eststo: quietly regress job fem phd ment fel art cit
(est2 stored)
. estadd fitstat: *   // compute fitstat for all models
. esttab, cells(none)
>     scalars("n_rhs # RHS vars" "aic0 AIC" "aic_n AIC*n"
```

```
>             "bic0 BIC" "bic_p BIC´")
>     nonumbers mtitles("Model 1" "Model 2")
```

	Model 1	Model 2
N	408	408
# RHS vars	3	6
AIC	2.639	2.580
AIC*n	1076.7	1052.8
BIC	-1359.9	-1371.7
BIC´	-48.30	-60.16

```
. eststo clear  // drop the stored estimates
```

A difference between the two approaches is that with the first method, output from `estadd fitstat` is displayed, whereas execution with the colon syntax is silent. We turned off the model numbers in the table by using the `nonumber` option and added model labels by using `mtitles()`. Furthermore, after using `esttab` to create a table for a given set of results, you may need to clear memory of results that have been stored so that they do not interfere with later tables. This can be done by using the `eststo clear` command.

2.1 Common syntax

Common to all featured commands is the basic syntax

`estadd` *cmd* [*...* `,` `replace` `prefix(`*string*`)` `quietly` *options*] [`:` *namelist*]

where *cmd* is the name of the `SPost` command in question and *namelist* is an optional list of stored estimation sets to which the command will be applied. `replace` permits `estadd` to overwrite existing `e()` returns, `prefix()` specifies a prefix for the names of the added results, and `quietly` suppresses the output of the `SPost` command. *options* are additional options specific to the `SPost` command. `replace`, `prefix()`, `quietly`, and *namelist* are global options and will not be repeated in the syntax diagrams below. Each of the supported `SPost` commands is now considered in alphabetical order.

2.2 estadd brant

The `brant` command tests the parallel regression assumption after an ordered logit or ordered probit model (`ologit` or `oprobit`). The syntax to add results from `brant` is

`estadd brant` [`,` *brant_options*]

where *brant_options* are as described in Long and Freese (2006, 452–454) or in `help brant`. `estadd brant` adds the results of the overall test to the scalars `e(brant_chi2)` (value of test statistic), `e(brant_df)` (degrees of freedom), and `e(brant_p)` (p-value), and adds matrix `e(brant)` containing χ^2 statistics for the individual regressors in the

first row and the corresponding p-values in the second row. The rows of `e(brant)` can be addressed in `estout`'s `cells()` option as `brant[chi2]` and `brant[p>chi2]`. Alternatively, type `brant[#]`, where # is the row number. For example,

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/ordwarm3
(GSS data on attitudes about working women for 1977 & 1989  \ 2009-03-13)
. ologit warm yr89 male white age ed prst
  (output omitted)
. estadd brant, quietly
added scalars:
         e(brant_chi2) =  49.181219
           e(brant_df) =  12
            e(brant_p) =  1.944e-06
added matrix:
              e(brant) :  2 x 6      (chi2, p>chi2)
. esttab, cells("brant[chi2](fmt(1)) brant[p>chi2](fmt(3))" )
>     stats(brant_chi2 brant_p, fmt(1 3) layout("@ @") label("Overall"))
>     nomtitles nonumbers
```

	chi2	p>chi2
yr89	13.0	0.001
male	22.2	0.000
white	1.3	0.531
age	7.4	0.025
ed	4.3	0.116
prst	4.3	0.115
Overall	49.2	0.000

We use the `fmt(#)` option to specify the number of decimal digits to report and the `label("Overall")` option to set the name for the omnibus Brant test results. The `layout("@ @")` option specifies that the `brant_chi2` and `brant_p` statistics be placed in the same row.

2.3 estadd fitstat

The `fitstat` command computes numerous measures of fit for many kinds of regression models. The syntax to add results from `fitstat` is

`estadd fitstat` [, *fitstat_options*]

where *fitstat_options* are as described in Long and Freese (2006, 452–454) or in `help fitstat`. The list of added statistics depends on model and options. For example, researchers frequently want to include at the base of a table the sample size along with fit measures and statistical tests. This can be done as follows, where we also illustrate how you can add information on the provenance of the table to the footer.

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/binlfp3
(Mroz´s 1976 data on labor force participation of women \ 2009-03-13)
```

```
. logit lfp k5 k618 age wc hc lwg inc, nolog
  (output omitted)
. estadd fitstat
  (output omitted)
. eststo logit
. probit lfp k5 k618 age wc hc lwg inc, nolog
  (output omitted)
. estadd fitstat
  (output omitted)
. eststo probit
. local lrlbl "LRX2(`e(lrx2_df)´)"
. local date : di %dCY-N-D d(`c(current_date)´) // get date
. local notes addnotes("t statistics in parentheses"
>     "Two-tailed tests: * p<0.05, ** p<0.01, *** p<0.001"
>     "Source: bjsl04-fitstat.do `date´ Scott Long.")
. esttab,
>     scalars("r2_mf R2_McFadden" "bic BIC" "lrx2 `lrlbl´" "lrx2_p Prob")
>     wide mtitles title(Comparing logit and probit on lfp) nonotes `notes´
Comparing logit and probit on lfp
```

	(1) logit		(2) probit	
k5	-1.463***	(-7.43)	-0.875***	(-7.70)
k618	-0.0646	(-0.95)	-0.0386	(-0.95)
age	-0.0629***	(-4.92)	-0.0378***	(-4.97)
wc	0.807***	(3.51)	0.488***	(3.60)
hc	0.112	(0.54)	0.0572	(0.46)
lwg	0.605***	(4.01)	0.366***	(4.17)
inc	-0.0344***	(-4.20)	-0.0205***	(-4.30)
_cons	3.182***	(4.94)	1.918***	(5.04)
N	753		753	
R2_McFadden	0.121		0.121	
BIC	958.3		958.4	
LRX2(7)	124.5		124.4	
Prob	8.92e-24		9.47e-24	

```
t statistics in parentheses
Two-tailed tests: * p<0.05, ** p<0.01, *** p<0.001
Source: bjsl04-fitstat.do 2009-10-14 Scott Long.
. eststo clear  // drop the stored estimates
```

The commands

```
logit lfp k5 k618 age wc hc lwg inc, nolog
estadd fitstat
eststo logit
```

fit the logit model, compute fit statistics, and store the model estimates along with the postestimation statistics using the reference name `logit`. Similarly, the following commands do the same things for the probit model:

```
probit lfp k5 k618 age wc hc lwg inc, nolog
estadd fitstat
eststo probit
```

To label the likelihood-ratio χ^2 statistic `LRX2`, we want to incorporate the degrees of freedom into the label. We do this by constructing a local with the label in which the degrees of freedom are retrieved from the stored `e(lrx2_df)`:

```
local lrlbl "LRX2(`e(lrx2_df)´)"
```

Within the `scalars()` option of `esttab`, `"lrx2 ‘lrlbl’"` indicates that the statistic named `lrx2` be given the label saved in the local `lrlbl`. We also want to customize the footer of the table. To do this, we create a local `notes` as follows, where each line of the footer is included in double quotes:

```
local date : di %dCY-N-D d(`c(current_date)´)
local notes addnotes("t statistics in parentheses"       ///
    "Two-tailed tests: * p<0.05, ** p<0.01, *** p<0.001" ///
    "Source: bjsl04-fitstat.do `date´ Scott Long.")
```

Finally, in the `esttab` command, the `nonotes` option suppresses the footer that is created by default, and our customized footer is created with the `addnotes()` option contained in the local `notes`.

2.4 estadd listcoef

The `listcoef` command lists transformations of the estimated coefficients for a variety of regression models. The syntax to add results from `listcoef` is

`estadd listcoef` [*varlist*] [`, nosd` *listcoef_options*]

where *listcoef_options* are as described in Long and Freese (2006, 464–467) or in `help listcoef`. Furthermore, the `nosd` option suppresses adding the standard deviations of the variables in `e(b_sdx)`.

Depending on the estimation command and options, `estadd listcoef` adds several vectors containing statistics such as standardized coefficients or factor change coefficients. See `estadd`'s online documentation for details. As a simple example, consider tabulating standardized coefficients for a `logit` model:

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/binlfp3
(Mroz´s 1976 data on labor force participation of women \ 2009-03-13)
. logit lfp k5 k618 age wc hc lwg inc, nolog
  (output omitted)
. estadd listcoef, std quietly
added matrices:
               e(b_xs) :  1 x 7      (bStdX)
               e(b_ys) :  1 x 7      (bStdY)
              e(b_std) :  1 x 7      (bStdXY)
              e(b_sdx) :  1 x 7      (SDofX)
```

```
. estadd fitstat
  (output omitted)
. local lrlbl "LRX2(`e(lrx2_df)´)"
. esttab, cells((b(lab(b-unstd) f(3))     t(lab(t-value) f(2))
>                b_xs(lab(b-xstd) f(3))   b_ys(lab(b-ystd) f(3))
>                b_std(lab(b-std) f(3))   b_sdx(lab(sd(x)) f(3))
>               ))
>     scalars("bic0 BIC" "lrx2 `lrlbl´" "lrx2_p Prob")
>     nomtitles nonumbers compress
>     title(Logit on lfp)
Logit on lfp
```

	b-unstd	t-value	b-xstd	b-ystd	b-std	sd(x)
k5	-1.463	-7.43	-0.767	-0.714	-0.374	0.524
k618	-0.065	-0.95	-0.085	-0.031	-0.042	1.320
age	-0.063	-4.92	-0.508	-0.031	-0.248	8.073
wc	0.807	3.51	0.363	0.394	0.177	0.450
hc	0.112	0.54	0.055	0.055	0.027	0.488
lwg	0.605	4.01	0.355	0.295	0.173	0.588
inc	-0.034	-4.20	-0.401	-0.017	-0.195	11.635
_cons	3.182	4.94				
N	753					
BIC	-4029.7					
LRX2(7)	124.5					
Prob	8.92e-24					

The `f(#)` option within `cells()` controls the number of decimal digits that are reported.

As another example, consider the tabulation of results for specific contrasts in a multinomial logit model. Use `listcoef`'s `lt`, `gt`, and `adjacent` options to determine the contrasts for which results are to be computed. For example,

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/nomocc3
(GSS data on career outcomes for 1982 \ 2009-03-13)
. mlogit occ white ed exper, nolog
  (output omitted)
. estadd listcoef, gt adjacent quietly
added matrices:
              e(b_raw) :  1 x 12     (b)
               e(b_se) :  1 x 12     (se)
                e(b_z) :  1 x 12     (z)
                e(b_p) :  1 x 12     (P>|z|)
             e(b_fact) :  1 x 12     (e^b)
            e(b_facts) :  1 x 12     (e^bStdX)
              e(b_sdx) :  1 x 12     (SDofX)
```

(*Continued on next page*)

```
. esttab, main(b_facts) unstack not nostar nonote modelwidth(14)
```

	(1) occ			
	BlueCol-Menial	Craft-BlueCol	WhiteCol-Craft	Prof-WhiteCol
white	1.407	0.810	1.355	1.058
ed	0.746	1.767	2.147	3.505
exper	1.068	1.378	1.101	1.015
N	337			

The raw coefficients for the requested contrasts are added in `e(b_raw)` (along with additional vectors containing standard errors, z statistics, and p-values).

2.5 estadd mlogtest

The `mlogtest` command computes various tests for multinomial logit models (`mlogit`). The syntax to add results from `mlogtest` is

`estadd mlogtest` [*varlist*] [, *mlogtest_options*]

where *mlogtest_options* are as described in Long and Freese (2006, 473–476) or in `help mlogtest`. `estadd mlogtest` adds a variety of results depending on the specified options (see the online documentation). For example, to compute the likelihood-ratio test that all the coefficients associated with a given independent variable are simultaneously equal to zero, you can use the command `mlogtest, lr`. To place these results in a table, type

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/nomocc3
(GSS data on career outcomes for 1982 \ 2009-03-13)
. mlogit occ white ed exper, nolog
  (output omitted)
. estadd mlogtest, lr
**** Likelihood-ratio tests for independent variables (N=337)
 Ho: All coefficients associated with given variable(s) are 0.
```

	chi2	df	P>chi2
white	8.095	4	0.088
ed	156.937	4	0.000
exper	8.561	4	0.073

```
added matrices:
             e(lrtest) :  3 x 3      (chi2, df, p)
```

```
. esttab, cell(( lrtest[chi2](label(LRX2))
>                lrtest[df]
>                lrtest[p](label(p-value))
>             ))
>     mlabel(none) nonumber noobs
```

	LRX2	df	p-value
white	8.095408	4	.0881451
ed	156.9372	4	6.63e-33
exper	8.560953	4	.073061

2.6 estadd prchange

The `prchange` command computes discrete and marginal changes in predictions for regression models. The syntax to add results from `prchange` is

`estadd prchange` [*varlist*] [*if*] [*in*] [`, pattern(`*typepattern*`) binary(`*type*`) continuous(`*type*`)` [`no`]`avg split` [`(`*prefix*`)`] *prchange_options*]

where *prchange_options* are as described in Long and Freese (2006, 478–485) or in `help prchange`. For example, the `outcome()` option may be used with models for count, ordered, or nominal variables to request results for a specific outcome. Further options are the following:

`pattern(`*typepattern*`)`, `binary(`*type*`)`, and `continuous(`*type*`)` determine which types of discrete change effects are added as the main results. The default is to add the 0 to 1 change effect for binary variables and the standard deviation change effect for continuous variables. Use `binary(`*type*`)` and `continuous(`*type*`)` to change these defaults. Available types are

type	Description
`minmax`	minimum to maximum change effect
`01`	0 to 1 change effect
`delta`	`delta()` change effect
`sd`	standard deviation change effect
`margefct`	marginal effect (only some models)

Use `pattern(`*typepattern*`)` if you want to determine the type of the added effects individually for each regressor. For example, `pattern(minmax sd delta)` would add `minmax` for the first regressor, `sd` for the second, and `delta` for the third, and then proceed using the defaults for the remaining variables.

`avg` requests that only the average results over all outcomes are added if applied to ordered or nominal models (`ologit`, `oprobit`, `slogit`, `mlogit`, and `mprobit`). The default is to add the average results as well as the individual results for the different

outcomes (unless `prchange`'s `outcome()` option is specified, in which case only results for the indicated outcome are added). Furthermore, specify `noavg` to suppress the average results and add only the outcome-specific results. `avg` may not be combined with `split` or `outcome()`.

`split`[`(`*prefix*`)`] saves each outcome's results in a separate estimation set if applied to ordered or nominal models (`ologit`, `oprobit`, `slogit`, `mlogit`, and `mprobit`). This can be useful if you want to tabulate the results for the different outcomes in separate columns beside one another. The estimation sets are named *prefix*#, where # is the value of the outcome at hand. If no *prefix* is provided, the name of the estimation set followed by an underscore is used as the prefix. If the estimation set has no name (because it has not yet been stored), then the name of the estimation command followed by an underscore is used as the prefix (e.g., `ologit_`). The estimation sets stored by the `split` option are intended for tabulation only and should not be used with other postestimation commands.

`estadd prchange` returns the discrete change effects in matrix `e(dc)`. The first row of the matrix contains the main results as determined by `pattern()`, `binary()`, and `continuous()`. The second and following rows contain the separate results for each type of effect, using the labels provided by `prchange` as row names. Type `dc[#]` or `dc[`*label*`]` to address the rows in `estout`'s `cells()` option, where # is the row number or *label* is the row name. For example, type `dc[-+sd/2]` to address the centered standard-deviation change effects. To tabulate the main results (first row), simply type `dc`. See the online documentation for further details on added results.

Space constraints do not permit giving detailed examples for all the variants. We illustrate only the application of the `split` option with a stereotype logistic regression:

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/ordwarm3
(GSS data on attitudes about working women for 1977 & 1989  \ 2009-03-13)
. slogit warm yr89 male white age ed prst, nolog
  (output omitted)
. estadd prchange, split quietly
added scalars:
            e(predval) =  .11714775
            e(outcome) =  1
              e(delta) =  1
          e(centered) =  1
added matrices:
                 e(dc) :  5 x 6      (main, Min->Max, 0->1, -+1/2, -+sd/2)
            e(pattern) :  1 x 6
                  e(X) :  4 x 6      (X, SD, Min, Max)
first row in e(dc) contains:
  01 change for binary variables
  sd change for continuous variables
results for outcome 1 stored as slogit_1
results for outcome 2 stored as slogit_2
results for outcome 3 stored as slogit_3
results for outcome 4 stored as slogit_4
```

```
. esttab, main(dc 3) not nostar scalars("predval Pr(y|x)") noobs
>     mtitles nonote title(Discrete changes in outcome probabilities.)
>     addnotes("Note: Change from 0 to 1 for binary variable,"
>              "      else a standard deviation change.")
Discrete changes in outcome probabilities.
```

	(1) 1SD	(2) 2D	(3) 3A	(4) 4SA
yr89	-0.055	-0.081	0.058	0.078
male	0.077	0.105	-0.082	-0.100
white	0.036	0.056	-0.036	-0.055
age	0.039	0.055	-0.042	-0.052
ed	-0.021	-0.030	0.022	0.028
prst	-0.010	-0.014	0.011	0.013
Pr(y|x)	0.117	0.323	0.392	0.167

```
Note: Change from 0 to 1 for binary variable,
      else a standard deviation change.
. eststo clear
```

2.7 estadd prvalue

The `prvalue` command computes model predictions at specified values of the independent variables for regression models for categorical and count variables. The procedure to add results from `prvalue` slightly differs from that for the other commands. First, results have to be collected by repeated calls to `estadd prvalue` by using the syntax

`estadd prvalue` [*if*] [*in*] [`,` `label(`*string*`)` *prvalue_options*]

where *prvalue_options* are as described in Long and Freese (2006, 493–497) or in `help prvalue`. For example, use `x()` and `rest()` to set the values of the independent variables. Furthermore, use `label()` to label the single calls. `pred`# is used as the label if `label()` is omitted, where # is the number of the call. Labels may contain spaces, but they will be trimmed to a maximum length of 30 characters, and some special characters (`:` and `.` and `"`) will be replaced by an underscore. The results from the single calls are collected in some intermediary matrices. Specify `replace` to drop results from previous calls.

Second, after collecting results, `estadd prvalue post` is used to post the predictions and their standard errors (if available) in `e(b)` and `e(se)` so that they can be tabulated (see the online help for information on additional returns). The syntax for posting the predictions is

`estadd prvalue post` [*name*] [`,` `title(`*string*`)` `swap`]

The default for `estadd prvalue post` is to replace the current model with the posted results. However, if *name* is specified, the posted results are stored and the

current model remains active. Use `title()` to specify a title for the stored results. Use `swap` to determine how the results are arranged in `e(b)`. The default is to group predictions by outcomes (i.e., outcome labels are used as equations). However, if `swap` is specified, predictions are grouped by calls (i.e., prediction labels are used as equations).

As an example, consider tabulating the predicted probabilities of labor force participation depending on whether a woman attended college:

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/binlfp3
(Mroz´s 1976 data on labor force participation of women \ 2009-03-13)
. logit lfp k5 k618 age wc hc lwg inc, nolog
  (output omitted)
. estadd prvalue, x(wc=1) label(Wife attended college)
  (output omitted)
. estadd prvalue, x(wc=0) label(Wife did not go to college) save
  (output omitted)
. estadd prvalue, x(wc=1) label(Difference) dif
  (output omitted)
. estadd prvalue post
scalars:
                  e(N) =  753
macros:
             e(depvar) : "lfp"
                e(cmd) : "estadd_prvalue"
              e(model) : "logit"
         e(properties) : "b"
matrices:
                  e(b) :  1 x 6      (predictions)
                 e(se) :  1 x 6      (standard errors)
                 e(LB) :  1 x 6      (lower CI bounds)
                 e(UB) :  1 x 6      (upper CI bounds)
           e(Category) :  1 x 6      (outcome values)
                  e(X) :  7 x 3      (k5, k618, age, wc, hc, lwg, inc)
. esttab, ci(3) wide nostar noobs nonumber nomtitle nonote
>     varwidth(30) modelwidth(11 16) collabels("Pr(in LF)" "95% CI")
>     eqlabels(none) keep(1InLF:)
>     title(Change in probability if wife attends college)
>     note(Note: All other variables held at their mean)
Change in probability if wife attends college
------------------------------------------------------------
                                Pr(in LF)           95% CI
------------------------------------------------------------
Wife attended college               0.710    [0.633,0.786]
Wife did not go to college          0.522    [0.473,0.570]
Difference                          0.188    [0.090,0.286]
------------------------------------------------------------
Note: All other variables held at their mean
```

The three `prvalue` commands compute predictions that are used in three rows of the created table. Notice that the first `prvalue` computes the predictions with `wc=1` and the second, with `wc=0`, while the third `prvalue` computes the difference between the two prior predictions. `estadd prvalue post` then saves the predictions. The `esttab`

command uses options already introduced, although there are a few things to note. `varwidth()` specifies the width of the left stub of the table containing the labels. `modelwidth()` specifies the widths of the models' columns.

2.8 estadd asprvalue

The `asprvalue` command computes predicted values for models with alternative-specific variables. Adding results from `asprvalue` is analogous to adding results from `prvalue`. That is, first, a series of predictions is collected by repeated calls by using the syntax

`estadd asprvalue` [*if*] [*in*] [`, label(`*string*`)` *asprvalue_options*]

where *asprvalue_options* are as described in Long and Freese (2006, 450–452) or in `help asprvalue`. Second, the collected results are posted by using the syntax

`estadd asprvalue post` [*name*] [`, title(`*string*`) swap`]

See section 2.7 for details. Examples can be found on `estout`'s web site.

3 Conclusions

Tables are critical to effectively convey substantive results. But creating tables can be an extremely tedious and error-prone process. The `estout` package makes it very simple to construct basic tables and export them to your word processor. The enhancements to the `estout` and `SPost` packages make it simple to create complex tables that contain both coefficients and postestimation results. We hope that this not only makes life simpler for users but also encourages researchers to create tables containing more detailed information.

4 References

Jann, B. 2005. Making regression tables from stored estimates. *Stata Journal* 5: 288–308.

———. 2007. Making regression tables simplified. *Stata Journal* 7: 227–244.

Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.

Long, J. S., and J. Freese. 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.

(*Continued on next page*)

About the authors

Ben Jann is a sociologist at ETH Zürich in Zürich, Switzerland.

J. Scott Long is Chancellor's Professor of Sociology and Statistics at Indiana University. He is the author of *Regression Models for Categorical and Limited Dependent Variables*, coauthor (with Jeremy Freese) of *Regression Models for Categorical Dependent Variables Using Stata*, and author of the recently published *The Workflow of Data Analysis Using Stata*.

The Stata Journal (2010)
10, Number 1, pp. 61–68

riskplot: A graphical aid to investigate the effect of multiple categorical risk factors

Milena Falcaro
Biostatistics Group
University of Manchester, UK
milena.falcaro@manchester.ac.uk

Andrew Pickles
Biostatistics Group
University of Manchester, UK
andrew.pickles@manchester.ac.uk

Abstract. In this article, we describe a command, `riskplot`, aiming to provide a visual aid to assess the strength, importance, and consistency of risk factor effects. The plotted form is a dendrogram that branches out as it moves from left to right. It displays the mean of some score or the absolute risk of some outcome for a sample that is progressively disaggregated by a sequence of categorical risk factors. Examples of the application of the new command are drawn from the analysis of depression and fluid intelligence in a sample of elderly men and women.

Keywords: gr0044, riskplot, pathways, risk factors, graphs

1 Introduction

Getting a rounded picture of the contributions of individual risk factors and their combination and interaction on the risk of some disease is not easy. While we may have moved on from a narrow focus on significance levels to present confidence intervals for effect estimates obtained under particular models, we rarely progress to isolate the breadth of the sample evidence base for each effect or fully describe the likely importance of the effect on population risk.

Pickles and colleagues (Quinton et al. 1993; Hill et al. 2001) have used manually constructed diagrams to illustrate the impact of sequential risks and risk pathways to study outcome. They considered the effect of a "turning point", the gaining of a supportive partner, on reducing the risk of antisocial behavior (Quinton et al. 1993), and also how such supportive love relationships may be able to moderate the effects of childhood sexual abuse on the risk of depression in adulthood (Hill et al. 2001). The latter extended the diagram in several ways by 1) plotting estimates of risk and risk factor prevalences that were obtained after adjustment for the sample design by the use of weights and 2) attaching to each risk factor combination a label that can contain additional quantitative information, such as the outcome risk under some model or the population attributable fraction for that pathway. In this article, we describe a new command, `riskplot`, that generates plots such as these in a straightforward way.

The `riskplot` command is not statistically sophisticated but is a visual aid intended to give an audience access to the data to assess the strength, importance, and consistency of risk factor effects. It displays the mean of some score or the risk of some outcome, commonly the proportion affected, for a sample that is progressively disaggregated by a

sequence of categorical risk factors. If the risk factors are considered in a chronological order of exposure, then it becomes natural to think of the progressive disaggregation as being a risk factor pathway and the plot can be used to highlight particular pathways that are especially common or lead to unusually high or low outcome risks. As such, they would seem a natural visual aid in the fields of development and lifecourse epidemiology, highlighting the impact of accumulating risk factors.

2 The riskplot command

2.1 Description

The `riskplot` command draws a plot that shows the effect of a set of categorical risk factors on some outcome of interest, Y. The plotted form is a dendrogram that branches out progressively as it moves from left to right, and in which the y axis is the expected value of the outcome and the x axis is a sequence of categorical risk factors, with the vertical width of the branches representing the relative frequency of the risk factor combination.

On the right-hand side next to each branch end, the user can decide to display additional information such as the observed and expected frequencies and the "pathway labels" (composite strings obtained by concatenating the numeric values of the risk factor levels).

For binary risks, the plot is easily interpretable for up to three or four risk factors, though the setting of a threshold frequency such that the rare risk combinations are trimmed from the sample and omitted from the plot often helps. The `riskplot` command also offers scope for the use of color to distinguish, for example, high from low risk pathways. This also helps highlight circumstances where interaction and moderation take place, such that the same level of one risk factor may either increase or decrease risk depending upon the level of a prior risk factor. The use of weights is also allowed and permits, for example, adjustment via probability weights for stratified sampling and attrition. However, when the computation of expected frequencies is required, only frequency weights may be specified.

2.2 Syntax

`riskplot` *depvar* [*xvars*] [*if*] [*in*] [*weight*] [`, all path observed expected`
`c(`*colorstyle*`) thick(#) trim(#) xextended(#)` *twoway_options*]

depvar represents the outcome variable. *xvars* is a list of categorical risk factors whose values must be integers between 0 and 9. No more than 20 risk-factor–level combinations may be displayed in the same plot. Observations that have missing values for either *depvar* or any of the *xvars* are disregarded. Weights are allowed, but when the `all` or `expected` option is specified, only `fweight` may be used. The user is responsible for ensuring the validity of what is produced when `aweight`s or `iweight`s are used.

2.3 Options

`all` displays pathway labels as well as observed and expected frequencies and percentages. This is equivalent to simultaneously specifying the `path`, `observed`, and `expected` options.

`path` displays the pathway labels at the end of each branch.

`observed` displays observed frequencies and percentages.

`expected` displays expected frequencies and percentages (i.e., frequencies and percentages that we would observe if the risk factors were independent).

`c(`*colorstyle*`)` specifies the colors of the branches. *colorstyle* is a list of strings defining the colors to be used for each level of the risk factors. The first string refers to 0s, the second to 1s, and so on. In general, the ith string ($i \geq 1$) in `c()` represents the color for value $i - 1$ of the risk factors. The user may also specify a dot when the default color (i.e., black) is required. For example,

`c(red green blue blue)` implies 0 = red, 1 = green, 2 = blue, 3 = blue, others = black

`c(red . blue)` implies 0 = red, 1 = black, 2 = blue, others = black

`c(red)` implies 0 = red, others = black

`thick(#)` increases the line thickness of the branches. The number specified must be between 0 and 25, where 20 is twice as thick as 10. The default is `thick(10)`.

`trim(#)` prevents the display of branches with relative frequency smaller than the percentage specified. # must be a number between 0 and 100.

`xextended(#)` provides additional space for labels on the right-hand side of the graph.

twoway_options allow control of graph titles, legends, additional lines, text, etc.

2.4 Refining and repositioning graph labels

When using `riskplot`, some plotted pathways may end up being very close to each other, and the corresponding information displayed on the right side of the graph (e.g., path labels and frequencies) may easily overlap. The Graph Editor allows the user to change the appearance of the graph and, among other things, to add and move text. We recommend using this new graphical device to solve overlapping text problems. To use the Graph Editor, right-click on the graph and select "Start Graph Editor". For more details, see the *Stata 11 Graphics Reference Manual* (StataCorp 2009) and *A Visual Guide to Stata Graphics* (Mitchell 2004).

3 Examples

Here we illustrate the application of `riskplot` using data from the Steel-Dyne Cognitive Ageing cohort of elderly men and women. The study was established in 1982, and it involves the follow-up of over 6,000 normal healthy individuals aged 50 years and over (for more details, see Rabbitt et al. [2004]). In this article, we focus on the subsample of subjects who were assessed for depression in 1991 and in 1995. We graphically explore the effect of a set of risk factors on cognitive decline and depression as measured in 1995 by the AH4 intelligence test (Heim 1970) and the Yesavage geriatric depression scale test (Yesavage et al. 1982), respectively. More specifically, the AH4 test consists of two 65-item parts and yields a total score that is used as a scale for grading fluid intelligence, which is the ability to reason abstractly and to solve new problems. The Yesavage geriatric depression scale test is a screening instrument of 30 items with a yes/no format and a scale ranging between 0 (no depression) and 30 (severe depression).

The risk factors we focus on are `sex` (0 = male, 1 = female), social class (`sclass`: 0 = high, 1 = low), and depression (`depr1991`: 0 = no depression, 1 = mild depression, 2 = severe depression), as measured in 1991.

We start by considering a simple example (figure 1) illustrating the association of social class and previous depression status with AH4 (fluid intelligence) in 1995.

```
. riskplot AH4 sclass depr1991, path ytitle(Fluid intelligence 1995)
```

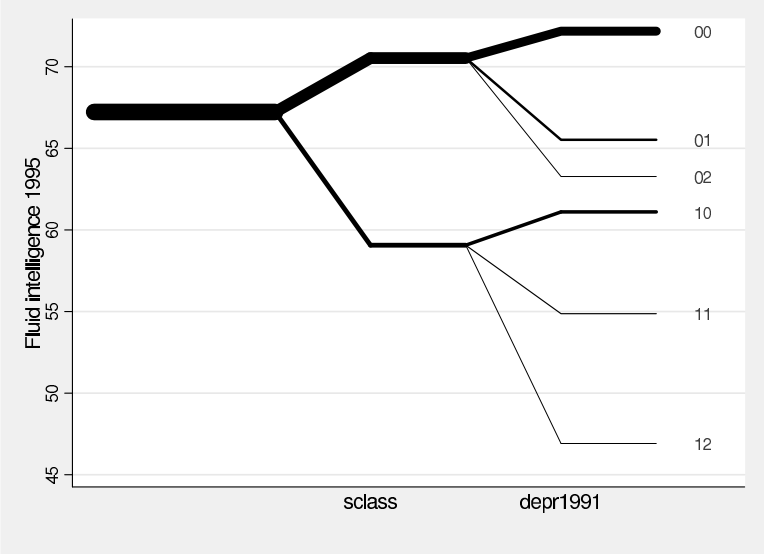

Figure 1. Simple risk plot for assessing the impact of social class and depression in 1991 on fluid intelligence in 1995. Pathway labels are displayed by specifying the `path` option.

The labels distinguish the risk combinations; for example, "12" indicates low social class (1) and severely depressed (2). Substantial differences in 1995 AH4 mean score are shown for both social class and depression, the consistent effect of depression being very apparent within each social class. We can also introduce `sex` as a risk factor, but here it is better to omit the branches with low relative frequency, say, less than 5%. This risk plot is displayed in figure 2 and is generated by the following command:

```
.riskplot AH4 sex sclass depr1991, path obs trim(5) ytitle(Fluid intelligence 1995)
```

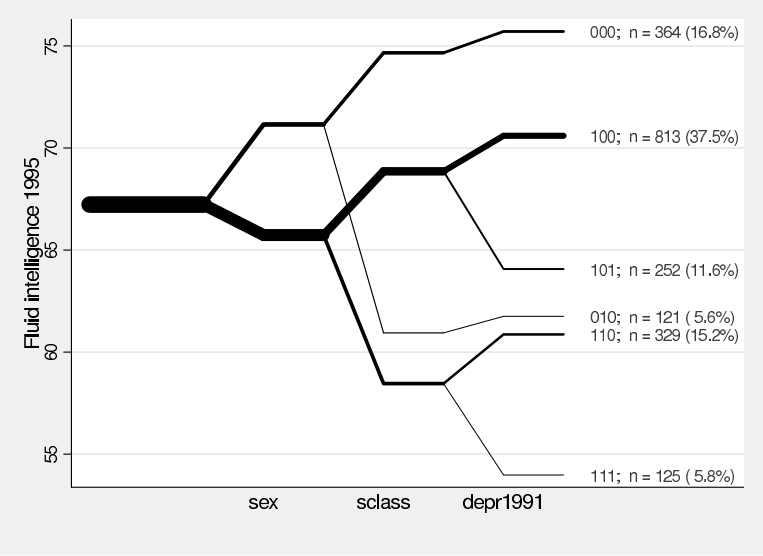

Figure 2. Risk plot with pathway labels and observed frequencies displayed using the `path` and `observed` options. Branches with relative frequency less than 5% are omitted by specifying `trim(5)`.

The plot shows that men with high social class and no depression at baseline tend to have better fluid intelligence performances than the other subjects in the sample.

Let's now consider the depression score in 1995 (`depr1995`) as the outcome of interest and explore the effect of sex and social class for subjects who were found to be depressed in 1991. Let `Idep91` denote a binary variable that takes on the value 1 when depr1991 is equal to 1 or 2, corresponding to subjects with mild (`depr1991` = 1) or severe (`depr1991` = 2) depression at baseline.

A clearer picture of the effect of risk factors on the outcome of interest can be obtained by specifying, for example, options for path colors and thickness. Observed frequencies and those expected under independence can also be added (figure 3). The `expected` option also generates some tabulated output in the standard results window. As the length of the labels increases, it may be necessary to move the right-hand margin of the plot with the `xextended(#)` option.

```
. riskplot depr1995 sex sclass if Idep91==1, all thick(20) scale(0.9) c(. red)
> title(Risk plot for subjects with mild or severe depression at baseline,
> margin(b+5)) ytitle(depression score 1995)

Observed frequencies
--------------------

       00:           48    (14.5%)
       01:           23     (6.9%)
       10:          175    (52.7%)
       11:           86    (25.9%)

Expected frequencies
--------------------

       00:         47.7    (14.4%)
       01:         23.3       (7%)
       10:        175.3    (52.8%)
       11:         85.7    (25.8%)
```

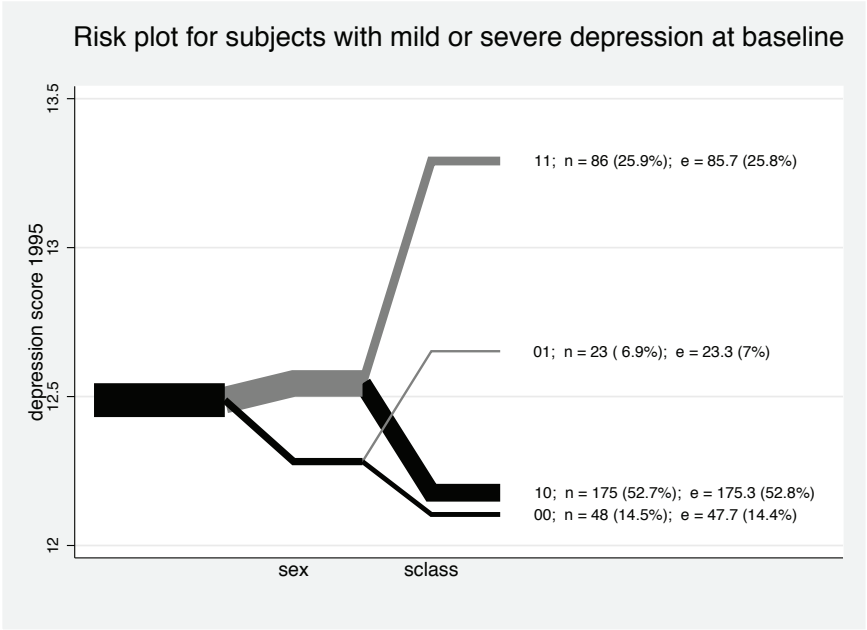

Figure 3. Risk plot with increased line thickness and with additional information. The colors of the branches are specified by using the `c(`*colorstyle*`)` option.

As very often happens in longitudinal studies, the Steel-Dyne Cognitive Ageing cohort is affected by the presence of selective dropout. Sampling weights can be estimated by modeling the probability of dropout via a logistic or probit regression and incorporated into the model to adjust for attrition.

```
. riskplot depr1995 sex sclass if Idep91==1 [pw=wg], path obs thick(20) c(. red)
> title(Risk plot for subjects with mild or severe depression at baseline)
> subtitle((results using sampling weights), margin(b+5)) scale(0.9)
> ytitle(depression score 1995)
```

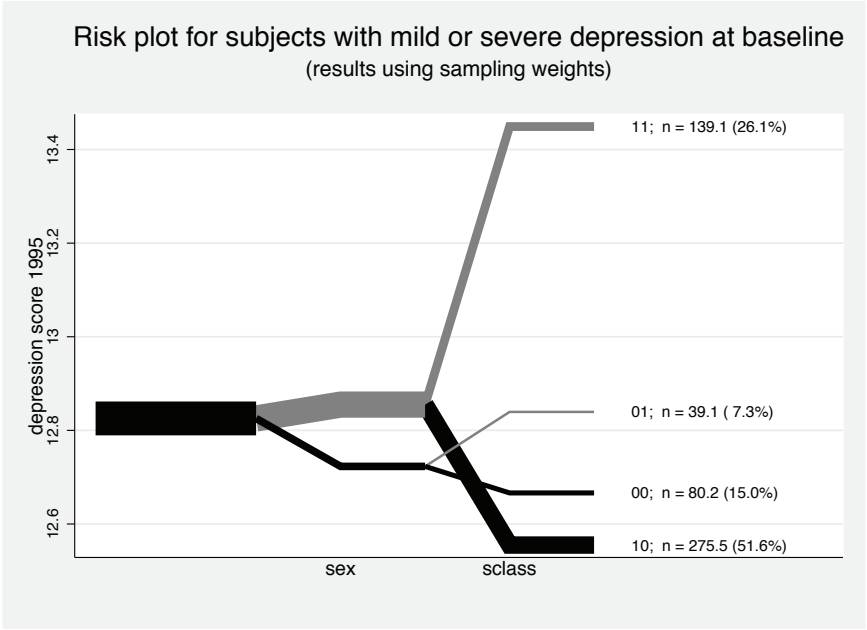

Figure 4. Risk plot using probability weights to account for sample attrition.

4 Conclusion

The `riskplot` command described in this article is a graphical aid to the investigation of the contributions of risk factors on outcomes of interest. In form, this plot has much in common with dendrograms derived from cluster analysis. For each of the risk factor combinations, it is possible to highlight differences in prevalence and to display additional information such as labels, and observed and expected frequencies and percentages. It might also prove useful in illustrating independence between two variables conditional upon a third and for examining possible mediation of effects.

5 Acknowledgments

This work was supported by the Economic and Social Research Council research grant RES-000-22-2921. We would like to thank Neil Pendleton and Mike Horan (Department of Geriatric Medicine, University of Manchester, UK) for allowing us access to the cognitive aging cohort data used in this article.

6 References

Heim, A. W. 1970. *A.H.4 Group Test of General Intelligence Manual*. Windsor, England: NFER Publishing.

Hill, J., A. Pickles, E. Burnside, M. Byatt, L. Rollinson, R. Davis, and K. Harvey. 2001. Child sexual abuse, poor parental care and adult depression: Evidence for different mechanisms. *British Journal of Psychiatry* 179: 104–109.

Mitchell, M. 2004. *A Visual Guide to Stata Graphics*. College Station, TX: Stata Press.

Quinton, D., A. Pickles, B. Maughan, and M. Rutter. 1993. Partners, peers, and pathways: Assortative pairing and continuities in conduct disorder. *Development and Psychopathology* 5: 763–783.

Rabbitt, P. M. A., L. McInnes, P. Diggle, F. Holland, N. Bent, V. Abson, N. Pendleton, and M. Horan. 2004. The University of Manchester longitudinal study of cognition in normal healthy old age, 1983 through 2003. *Aging, Neuropsychology, and Cognition* 11: 245–279.

StataCorp. 2009. *Stata 11 Graphics Reference Manual*. College Station, TX: Stata Press.

Yesavage, J. A., T. L. Brink, T. L. Rose, O. Lum, V. Huang, M. Adey, and V. O. Leirer. 1982. Development and validation of a geriatric depression screening scale: A preliminary report. *Journal of Psychiatric Research* 17: 37–49.

About the authors

Milena Falcaro is a research fellow in biostatistics at the University of Manchester. Her research interests include structural equations modeling, longitudinal data analysis, and genetic epidemiology. The context of much of her work has been longitudinal behavioral and developmental psychopathological studies.

Andrew Pickles is a professor of epidemiological and social statistics at the University of Manchester. He shares similar research interests to Milena, has long used Stata in teaching and research, and together with Sophia Rabe-Hesketh and Anders Skrondal formulated the generalized linear latent and mixed model (`gllamm`) in Stata.

The Stata Journal (2010)
10, Number 1, pp. 69–81

Power transformation via multivariate Box–Cox

Charles Lindsey
Texas A & M University
Department of Statistics
College Station, TX
lindseyc@stat.tamu.edu

Simon Sheather
Texas A & M University
Department of Statistics
College Station, TX
sheather@stat.tamu.edu

Abstract. We present a new Stata estimation program, `mboxcox`, that computes the normalizing scaled power transformations for a set of variables. The multivariate Box–Cox method (defined in Velilla, 1993, *Statistics and Probability Letters* 17: 259–263; used in Weisberg, 2005, *Applied Linear Regression* [Wiley]) is used to determine the transformations. We demonstrate using a generated example and a real dataset.

Keywords: st0184, mboxcox, mbctrans, boxcox, regress

1 Theory and motivation

Box and Cox (1964) detailed normalizing transformations for univariate y and univariate response regression using a likelihood approach. Velilla (1993) formalized a multivariate version of Box and Cox's normalizing transformation. A slight modification of this version is considered in Weisberg (2005), which we will use here.

The multivariate Box–Cox method uses a separate transformation parameter for each variable. There is also no independent/dependent classification of the variables. Since its inception, the multivariate Box–Cox transformation has been used in many settings, most notably linear regression; see Sheather (2009) for examples. When variables are transformed to joint normality, they become approximately linearly related, constant in conditional variance, and marginally normal in distribution. These are very useful properties for statistical analysis.

Stata currently offers several versions of Box–Cox transformations via the `boxcox` command. The multivariate options of `boxcox` are limited to regression settings where at most two transformation parameters are allowed. We present the `mboxcox` command as a useful complement to `boxcox`. We will start by explaining the formal theory of what `mboxcox` does.

First, we define a scaled power transformation as

$$\psi_s(y,\lambda) = \begin{pmatrix} \frac{y^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log y & \text{if } \lambda = 0 \end{pmatrix}$$

Scaled power transformations preserve the direction of associations that the transformed variable had with other variables. So scaled power transformations will not switch collinear relationships of interest.

Next, for n-vector $\mathbf{x}$, we define the geometric mean: $\text{gm}(\mathbf{x}) = \exp\left(1/n \sum_{i=1}^{n} \log x_i\right)$.

Suppose the random vector $\mathbf{x} = (x_1, \ldots, x_p)'$ takes only positive values. Let $\Lambda = (\lambda_1, \ldots, \lambda_p)$ be a vector of real numbers, such that $\{\psi_s(x_1, \lambda_1), \ldots, \psi_s(x_p, \lambda_p)\}$ is distributed $N(\mu, \Sigma)$.

Now we take a random sample of size n from the population of $\mathbf{x}$, yielding data $X = (\mathbf{x}_1, \ldots, \mathbf{x}_p)$. We define the transformed version of the variable X_{ij} as $X_{ij}{}^{(\lambda_j)} = \psi_s(X_{ij}, \lambda_j)$. This yields the transformed data matrix $X^{(\Lambda)} = \left\{\mathbf{x}_1{}^{(\lambda_1)}, \ldots, \mathbf{x}_p{}^{(\lambda_p)}\right\}$.

Finally, we define the normalized transformed data:

$$Z^{(\Lambda)} = \left\{\text{gm}(\underset{\sim}{x}_1)^{\lambda_1} \underset{\sim}{x}_1{}^{(\lambda_1)}, \ldots, \text{gm}(\underset{\sim}{x}_p)^{\lambda_p} \underset{\sim}{x}_p{}^{(\lambda_p)}\right\}$$

Velilla (1993, eq. 3) showed that the concentrated log likelihood of Λ in this situation was given by

$$L_c(\Lambda) = -\frac{n}{2} \log \left| Z^{(\Lambda)'} \left(I_n - \frac{1_n 1_n^{'}}{n}\right) Z^{(\Lambda)} \right|$$

Weisberg (2005) used modified scaled power transformations rather than plain scaled power transformations for each column of the data vector.

$$\psi_m(y_i, \lambda) = \text{gm}(\mathbf{y})^{1-\lambda} \psi_s(y_i, \lambda)$$

Under a modified scaled power transformation, the scale of the transformed variable is invariant to the choice of transformation power. So the scale of a transformed variable is better controlled under the modified scaled power transformation than under the scaled power transformation. Inference on the optimal transformation parameters should be similar under both scaled and modified scaled methods. The transformed data under a scaled power transformation is equivalent to the transformed data under an unscaled power transformation with an extra location/scale transformation. A multivariate normal random vector yields another multivariate normal random vector when a location/scale transformation is applied to it. So the most normalizing scaled transformation essentially yields as normalizing a transformation as its unscaled version. We thus expect great similarity between the optimal scaled, modified scaled, and unscaled parameter estimates.

The new concentrated likelihood (Weisberg 2005, 291, eq. A.36) is

$$L_c(\Lambda) = -\frac{n}{2} \log \left| Z_*{}^{(\Lambda)'} \left(I_n - \frac{1_n 1_n^{'}}{n}\right) Z_*{}^{(\Lambda)} \right|$$

Here $Z^{(\Lambda)}$ has been replaced by the actual transformed data.

$$Z_*{}^{(\Lambda)} = \left\{\text{gm}(\underset{\sim}{x}_1)^{1-\lambda_1} \underset{\sim}{x}_1{}^{(\lambda_1)}, \ldots, \text{gm}(\underset{\sim}{x}_p)^{1-\lambda_p} \underset{\sim}{x}_p{}^{(\lambda_p)}\right\}$$

In terms of the sample covariance of ${Z_*}^{(\Lambda)}$, $L_c(\Lambda)$ is a simple expression. In terms of Λ, it is very complicated. The `mboxcox` command uses $L_c(\Lambda)$ to perform inference on Λ, where the elements of Λ are modified scaled power transformation parameters. Because of the complexity of $L_c(\Lambda)$, a numeric optimization is used to estimate Λ. The second derivative of $L_c(\Lambda)$ is computed numerically during the optimization, and this yields the covariance estimate of Λ.

We should take note of the situation in which the data does not support a multivariate Box–Cox transformation. Problems in data collection may manifest as outliers. As Velilla (1995) states, "it is well known that the maximum likelihood estimates to normality is very sensitive to outlying observations." Additionally, the data or certain variables from it could simply come from a nonnormal distribution. Unfortunately, the method of transformation we use here is not sensitive to these problems. Our method of Box–Cox transformation is not robust. For methods that are robust to problems like these, see Velilla (1995) and Riani and Atkinson (2000). We present the basic multivariate Box–Cox transformation here, as a starting point for more robust transformation procedures to be added to Stata at a later date.

2 Use and a generated example

The `mboxcox` command has the following basic syntax:

`mboxcox` *varlist* [*if*] [*in*] [`, level(`#`)`]

Like other estimation commands, the results of `mboxcox` can be redisplayed with the following simpler syntax:

`mboxcox` [`, level(`#`)`]

The syntax of `mboxcox` is very simple and straightforward. We also provide the `mbctrans` command to create the transformed variables. This command is used to streamline the data transformation process. It takes inputs of the variables to be transformed and a list of transformation powers, and saves the transformed variables under their original names with a `t_` prefix. The command supports unscaled, scaled, and modified scaled transformations. Accomplish scaled transformations by specifying the `scale` option. To obtain modified scaled transformations, specify the `mscale` option.

`mbctrans` *varlist* [*if*] [*in*] [`, power(`*numlist*`) mscale scale`]

We generate 10,000 samples from a three-variable multivariate normal distribution with means $(10, 14, 32)$ and marginal variances $(1, 3, 2)$. The first and second variables are correlated with a covariance of 0.3.

```
. set obs 10000
obs was 0, now 10000
. set seed 3000
```

```
. matrix Means = (10,14,32)
. matrix Covariance = (1,.3,0)\(.3,3,0)\(0,0,2)
. drawnorm x1 x2 x3, means(Means) cov(Covariance)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
x1	10000	10.00191	.9943204	5.42476	13.72735
x2	10000	13.9793	1.713186	7.683866	21.38899
x3	10000	31.98648	1.41477	26.26886	38.04641

Next we transform the data using unscaled power transformations $(2, -1, 3)$. Note that the correlation direction between the first and second variable changes.

```
. mbctrans x1 x2 x3, power(2 -1 3)
. correlate t_x1 t_x2
(obs=10000)
```

	t_x1	t_x2
t_x1	1.0000	
t_x2	-0.1585	1.0000

We will use `mboxcox` to determine the optimal modified scaled power transformation estimates for normalizing the transformed data. The optimal unscaled power transformation vector is $(1/2, -1, 1/3)$, each element being the inverse of the variable's original transformation power.

```
. mboxcox t_x1-t_x3
Multivariate boxcox transformations
                                                Number of obs   =      10000
Likelihood Ratio Tests
```

Test	Log Likelihood	Chi2	df	Prob > Chi2
All powers -1	-67280.73	2078.173	3	0
All powers 0	-66461.51	439.7275	3	0
All powers 1	-66837.99	1192.704	3	0

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
lambda						
t_x1	.5318023	.0402718	13.21	0.000	.452871	.6107336
t_x2	-.9715714	.065297	-14.88	0.000	-1.099551	-.8435915
t_x3	.3647025	.0613916	5.94	0.000	.2443772	.4850278

We find that the modified scaled transformation parameter estimates of `mboxcox` are close to the unscaled parameters. The postestimation features of `mboxcox` tell us that there is no evidence to reject the assertion that the optimal modified scaled transformation parameters are identical to the unscaled parameters. This correspondence between modified scaled and unscaled is not surprising, as we detailed in the last section.

```
. test (t_x1= .5) (t_x2= -1) (t_x3 = 1/3)
 ( 1)  [lambda]t_x1 = .5
 ( 2)  [lambda]t_x2 = -1
 ( 3)  [lambda]t_x3 = .3333333
           chi2(  3) =    1.08
         Prob > chi2 =    0.7831
```

3 Real example

Sheather (2009) provides an interesting dataset involving 2004 automobiles. We wish to perform a regression of the variable `highwaympg` on the predictors `enginesize`, `cylinders`, `horsepower`, `weight`, `wheelbase`, and the dummy variable `hybrid`.

```
. use cars04, clear
. summarize highwaympg enginesize cylinders horsepower weight wheelbase hybrid
```

Variable	Obs	Mean	Std. Dev.	Min	Max
highwaympg	234	29.39744	5.372014	19	66
enginesize	234	2.899145	.925462	1.4	5.5
cylinders	234	5.517094	1.471374	3	12
horsepower	234	199.7991	64.03424	73	493
weight	234	3313.235	527.0081	1850	4474
wheelbase	234	107.1154	5.82207	93	124
hybrid	234	.0128205	.1127407	0	1

```
. regress highwaympg enginesize cylinders horsepower weight wheelbase hybrid
```

Source	SS	df	MS		
Model	5343.19341	6	890.532235	Number of obs	234
Residual	1380.84505	227	6.08301785	F(6, 227)	146.40
				Prob > F	0.0000
				R-squared	0.7946
				Adj R-squared	0.7892
Total	6724.03846	233	28.8585342	Root MSE	2.4664

highwaympg	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
enginesize	.166796	.5237721	0.32	0.750	-.8652809	1.198873
cylinders	-.1942966	.3171983	-0.61	0.541	-.8193262	.4307331
horsepower	-.0182825	.0052342	-3.49	0.001	-.0285963	-.0079687
weight	-.00662	.0007513	-8.81	0.000	-.0081003	-.0051397
wheelbase	.1797597	.0570666	3.15	0.002	.0673117	.2922078
hybrid	20.33805	1.468368	13.85	0.000	17.44467	23.23142
_cons	36.05649	4.726131	7.63	0.000	26.7438	45.36919

The model is not valid. It has a number of problems. Nonconstant variance of the errors is one. As explained in Sheather (2009), this problem can be detected by graphing the square roots of the absolute values of the standardized residuals versus the fitted values and continuous predictors. Trends in these plots suggest that the variance changes at different levels of the predictors and fitted values. We graph these plots and see a variety of increasing and decreasing trends.

```
. predict rstd, rstandard
. predict fit, xb
. generate nsrstd = sqrt(abs(rstd))
. local i = 1
. foreach var of varlist fit enginesize cylinders horsepower weight wheelbase {
  2. twoway scatter nsrstd `var´ || lfit nsrstd `var´,
> ytitle("|Std. Residuals|^.5") legend(off)
> ysize(5) xsize(5) name(gg`i´) nodraw
  3. local i = `i´ + 1
  4. }
. graph combine gg1 gg2 gg3 gg4 gg5 gg6, rows(2) ysize(10) xsize(15)
```

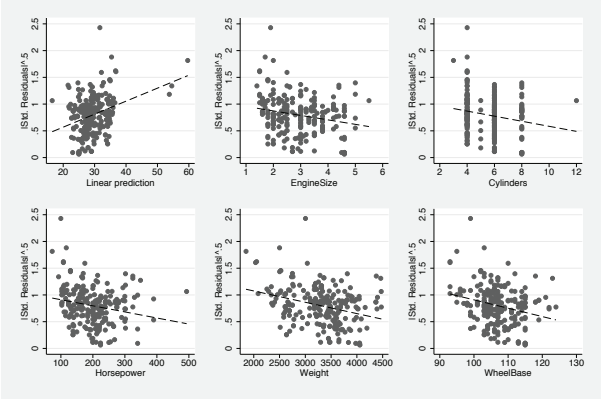

Figure 1. $\sqrt{|\,\text{Standard residuals}\,|}$ versus predictors and fitted values.

Data transformation would be a strategy to solve the nonconstant variance problem. As suggested in Weisberg (2005, 156), we should first examine linear relationships among the predictors. If they are approximately linearly related, we can use the fitted values to find a suitable transformation for the response, perhaps through an inverse response plot (Sheather 2009). A matrix plot of the response and predictors shows that we will not be able to do that. Many appear to share a monotonic relationship, but it is not linear.

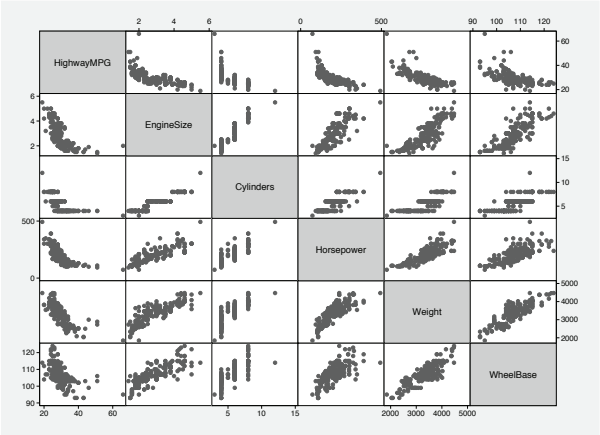

Figure 2. Matrix plot original response and predictors.

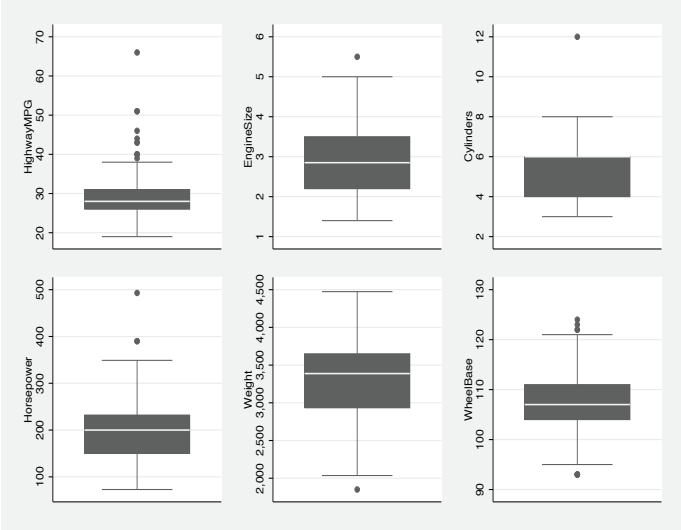

Figure 3. Box plots original response and predictors.

In addition, a look at the box plots reveals that several of the predictors and the response are skewed. The data are not consistent with a multivariate normal distribution. If the predictors and response were multivariate normal conditioned on the value of `hybrid`, then it would follow that the errors of the regression would have constant variance. The conditional variance of multivariate normal variables is always constant with regard to the values of the conditioning variables.

There are actually only three observations of `hybrid` that are nonzero. Data analysis not shown here supports the contention that `hybrid` only significantly affects the

location of the joint distribution of the remaining predictors and response. Successful inference on other more complex properties of the joint distribution, conditional on `hybrid` = 1, would require more data. Hence, we ignore the value of `hybrid` in calculating a normalizing transformation. In the first section, we mentioned that outliers could be a serious problem for our method. Our approach here could lead to outliers that would cause the transformation to fail.

If the marginal transformation that we estimate is suitably equivalent to the transformations obtained by conditioning on `hybrid` and approximately normalizes the other predictors and the response, then the errors of the regression will be at least approximately constant and its predictors and response more symmetric.

```
. mboxcox enginesize cylinders horsepower highwaympg weight wheelbase
Multivariate boxcox transformations
                                                Number of obs   =       234
Likelihood Ratio Tests
```

Test	Log Likelihood	Chi2	df	Prob > Chi2
All powers -1	-2431.978	202.6359	6	0
All powers 0	-2369.889	78.45681	6	7.438e-15
All powers 1	-2483.247	305.1733	6	0

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
lambda						
enginesize	.2550441	.1304686	1.95	0.051	-.0006697	.5107579
cylinders	-.0025143	.1745643	-0.01	0.989	-.344654	.3396255
horsepower	-.0169707	.1182906	-0.14	0.886	-.2488161	.2148747
highwaympg	-1.375276	.1966211	-6.99	0.000	-1.760646	-.9899057
weight	1.069233	.226236	4.73	0.000	.6258187	1.512647
wheelbase	.0674801	.6685338	0.10	0.920	-1.242822	1.377782

```
. test (enginesize=.25)(cylinders=0)(horsepower=0)(highwaympg=-1)
> (weight=1)(wheelbase=0)
 ( 1)  [lambda]enginesize = .25
 ( 2)  [lambda]cylinders = 0
 ( 3)  [lambda]horsepower = 0
 ( 4)  [lambda]highwaympg = -1
 ( 5)  [lambda]weight = 1
 ( 6)  [lambda]wheelbase = 0
           chi2(  6) =    3.99
         Prob > chi2 =    0.6777
```

Following the advice of Sheather (2009), we round the suggested powers to the closest interpretable fractions. We will use the `mbctrans` command to create the transformed variables so that we can rerun our regression. We demonstrate it here for all cases on `highwaympg`. The relationship it holds with the variable `dealercost` is used as a reference. Recall how the unscaled transformation may switch correlation relationships with other variables, and how the modified scaled transformation maintains these relationships and the scale of the input variable. The unscaled transformed `highwaympg` is referred to as `unscaled_hmpg`. The scaled transformed version of `highwaympg` is

named `scaled_hmpg`. The modified scaled transformed version of `highwaympg` is named `mod_scaled_hmpg`.

```
. summarize highwaympg
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
  highwaympg |       234    29.39744    5.372014         19         66

. correlate dealercost highwaympg
(obs=234)
             | dealer~t highwa~g
-------------+------------------
  dealercost |   1.0000
  highwaympg |  -0.5625   1.0000

. mbctrans highwaympg,power(-1)

. rename t_highwaympg unscaled_hmpg

. summarize unscaled_hmpg
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
unscaled_h~g |       234    .0349275    .0052762   .0151515   .0526316

. correlate dealercost unscaled_hmpg
(obs=234)
             | dealer~t unscal~g
-------------+------------------
  dealercost |   1.0000
unscaled_h~g |   0.6779   1.0000

. mbctrans highwaympg,power(-1) scale

. rename t_highwaympg scaled_hmpg

. summarize scaled_hmpg
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
 scaled_hmpg |       234    .9650725    .0052762   .9473684   .9848485

. correlate dealercost scaled_hmpg
(obs=234)
             | dealer~t scaled~g
-------------+------------------
  dealercost |   1.0000
 scaled_hmpg |  -0.6779   1.0000

. mbctrans highwaympg,power(-1) mscale

. rename t_highwaympg mod_scaled_hmpg

. summarize mod_scaled_hmpg
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
mod_scaled~g |       234    810.9419    4.433584   796.0653   827.5595

. correlate dealercost mod_scaled_hmpg
(obs=234)
             | dealer~t mod_sc~g
-------------+------------------
  dealercost |   1.0000
mod_scaled~g |  -0.6779   1.0000
```

Both the scaled and modified scaled transformation kept the same correlation relationship between `highwaympg` and `dealercost`. The unscaled transformation did not. Additionally, the modified scaled transformation maintained a scale much closer to that of the original than either of the other transformations. Now we will use `mbctrans` on all the variables.

```
. mbctrans enginesize cylinders horsepower highwaympg weight wheelbase,
> power(.25 0 0 -1 1 0) mscale
```

The box plots for the transformed data show a definite improvement in marginal normality.

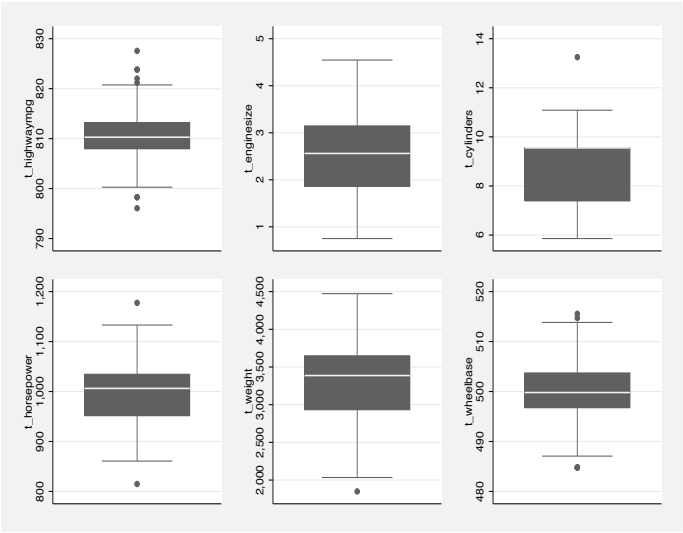

Figure 4. Box plots transformed response and predictors.

A matrix plot of the predictors and response shows greatly improved linearity.

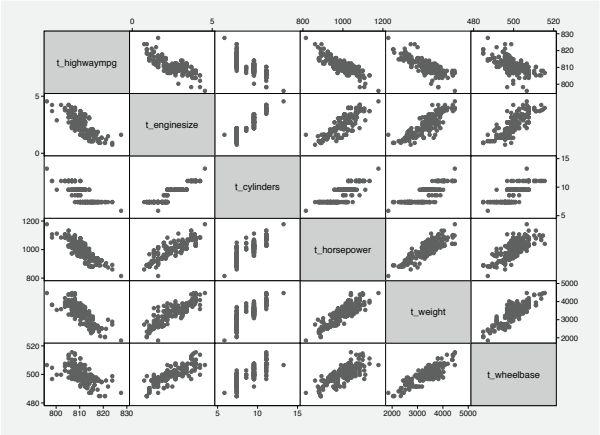

Figure 5. Matrix plot transformed response and predictors.

Now we refit the model with the transformed variables.

```
. regress t_highwaympg t_enginesize t_cylinders t_horsepower t_weight
> t_wheelbase hybrid
```

Source	SS	df	MS		
Model	3581.57374	6	596.928957	Number of obs	= 234
Residual	998.430492	227	4.39837221	F(6, 227)	= 135.72
				Prob > F	= 0.0000
Total	4580.00424	233	19.6566705	R-squared	= 0.7820
				Adj R-squared	= 0.7762
				Root MSE	= 2.0972

t_highwaympg	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
t_enginesize	-.406318	.4557007	-0.89	0.374	-1.304262	.4916264
t_cylinders	-.5353418	.2622172	-2.04	0.042	-1.052033	-.0186507
t_horsepower	-.0280757	.0051522	-5.45	0.000	-.038228	-.0179234
t_weight	-.0042486	.0006911	-6.15	0.000	-.0056103	-.0028868
t_wheelbase	.2456528	.0490344	5.01	0.000	.1490321	.3422736
hybrid	6.552501	1.276605	5.13	0.000	4.03699	9.068012
_cons	735.9331	23.74779	30.99	0.000	689.1388	782.7274

```
. predict trstd, rstandard
. predict tfit, xb
. generate tnsrstd = sqrt(abs(trstd))
. local i = 1
. foreach var of varlist tfit t_enginesize t_cylinders t_horsepower t_weight
> t_wheelbase {
  2. twoway scatter tnsrstd `var´ || lfit tnsrstd `var´,
> ytitle("|Std. Residuals|^.5") legend(off) ysize(5) xsize(5) name(gg`i´)
> nodraw
  3. local i = `i´ + 1
  4. }
. graph combine gg1 gg2 gg3 gg4 gg5 gg6, rows(2) ysize(10) xsize(15)
```

The nonconstant variance has been drastically improved. The use of `mboxcox` helped improve the fit of the model.

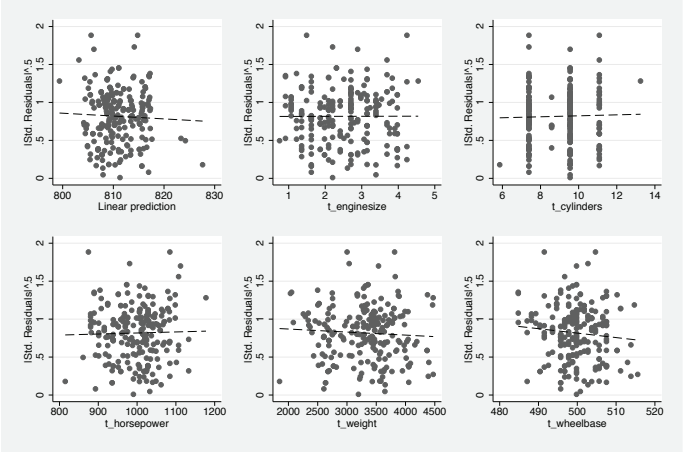

Figure 6. $\sqrt{|\text{Standard residuals}|}$ versus transformed predictors and fitted values.

4 Conclusion

We explored both the theory and practice of the multivariate Box–Cox transformation. Using both generated and real datasets, we have demonstrated the use of the multivariate Box–Cox transformation in achieving multivariate normality and creating linear relationships among variables.

We fully defined the `mboxcox` command as a method for performing the multivariate Box–Cox transformation in Stata. We also introduced the `mbctrans` command and defined it as a method for performing the power transformations suggested by `mboxcox`. Finally, we also demonstrated the process of obtaining transformation power parameter estimates from `mboxcox` and rounding them to theoretically appropriate values.

5 References

Box, G. E. P., and D. R. Cox. 1964. An analysis of transformations. *Journal of the Royal Statistical Society, Series B* 26: 211–252.

Riani, M., and A. C. Atkinson. 2000. Robust diagnostic data analysis: Transformations in regression. *Technometrics* 42: 384–394.

Sheather, S. J. 2009. *A Modern Approach to Regression with R*. New York: Springer.

Velilla, S. 1993. A note on the multivariate Box–Cox transformation to normality. *Statistics and Probability Letters* 17: 259–263.

———. 1995. Diagnostics and robust estimation in multivariate data transformations. *Journal of the American Statistical Association* 90: 945–951.

Weisberg, S. 2005. *Applied Linear Regression*. 3rd ed. New York: Wiley.

About the authors

Charles Lindsey is a PhD candidate in statistics at Texas A & M University. His research is currently focused on nonparametric methods for regression and classification. He currently works as a graduate research assistant for the Institute of Science Technology and Public Policy within the Bush School of Government and Public Service. He is also an instructor of a course on sample survey techniques in Texas A & M University's Statistics Department. In the summer of 2007, he worked as an intern at StataCorp. Much of the groundwork for this article was formulated there.

Simon Sheather is professor and head of the Department of Statistics at Texas A & M University. Simon's research interests are in the fields of flexible regression methods, and nonparametric and robust statistics. In 2001, Simon was named an honorary fellow of the American Statistical Association. Simon is currently listed on http://www.ISIHighlyCited.com among the top one-half of one percent of all mathematical scientists, in terms of citations of his published work.

The Stata Journal (2010)
10, Number 1, pp. 82–103

Centering and reference groups for estimates of fixed effects: Modifications to felsdvreg

Kata Mihaly
The RAND Corporation
Washington, DC
kmihaly@rand.org

Daniel F. McCaffrey
The RAND Corporation
Pittsburgh, PA
danielm@rand.org

J. R. Lockwood
The RAND Corporation
Pittsburgh, PA
lockwood@rand.org

Tim R. Sass
Florida State University
Tallahassee, FL
tsass@fsu.edu

Abstract. Availability of large, multilevel longitudinal databases in various fields including labor economics (with workers and firms observed over time) and education research (with students and teachers observed over time) has increased the application of panel-data models with multiple levels of fixed-effects. Existing software routines for fitting fixed-effects models were not designed for applications in which the primary interest is obtaining estimates of any of the fixed-effects parameters. Such routines typically report estimates of fixed effects relative to arbitrary holdout units. Contrasts to holdout units are not ideal in cases where the fixed-effects parameters are of interest because they can change capriciously, they do not correspond to the structural parameters that are typically of interest, and they are inappropriate for empirical Bayes (shrinkage) estimation. We develop an improved parameterization of fixed-effects models using sum-to-zero constraints that provides estimates of fixed effects relative to mean effects within well-defined reference groups (e.g., all firms of a given type or all teachers of a given grade) and provides standard errors for those estimates that are appropriate for shrinkage estimation. We implement our parameterization in a Stata routine called `felsdvregdm` by modifying the `felsdvreg` routine designed for fitting high-dimensional fixed-effects models. We demonstrate our routine with an example dataset from the Florida Education Data Warehouse.

Keywords: st0185, felsdvreg, felsdvregdm, fixed effects, linked employer–employee data, longitudinal achievement data

1 Introduction

The implementation of models that include a large number of fixed effects for individual units, such as employers and employees or teachers and students, has grown with the increased availability of large longitudinal datasets.[1] This is especially evident

1. Units do not need to be synonymous with physical units. A unit could be a firm or teacher during a period of time (e.g., year) or could be a subset of its function (e.g., a firm's manufacturing division versus its research division or a teacher's first period versus his second period class) so that the same physical entity could be associated with multiple units. The example in section 4 defines units as teacher-years so that teachers can be associated with multiple units.

in the context of employer–employee linked data (Abowd, Kramarz, and Roux 2006; Menezes-Filho, Muendler, and Ramey 2008) and student–teacher linked data (Aaronson, Barrow, and Sander 2007; Clotfelter, Ladd, and Vigdor 2006; Harris and Sass 2006; McCaffrey, Han, and Lockwood 2008; Rockoff 2004). It is usually not possible to estimate all the models' parameters because inclusion of a fixed effect for every individual unit generally makes the model overparameterized because of linear dependence among the independent variables in the model (e.g., the indicator or "dummy" variables for the individual units). Consequently, estimation requires identifying assumptions and reparameterization of the model. In cases where the fixed effects are used to control for unobserved heterogeneity of units (e.g., workers, firms, students, teachers) and interest lies in the estimation of the effects of other time-varying covariates (e.g., exposure to policies or programs), then the reparameterization is irrelevant because parameters and their estimates for time-varying covariates are invariant to different parameterizations of the fixed effects. In fact, when the model contains only one level of fixed effects, it is typically not explicitly included as regressors but rather is "absorbed" out prior to regression through the within-subjects transformation (Greene 2008; Wooldridge 2002). This is how several commonly used packages such as `areg`, `xtreg, fe`, and SAS® PROC GLM (SAS Institute, Inc. 2004) implement single-level fixed-effects estimation. Even if multiple levels of fixed effects are in the model, if the effects at all levels are nuisances, then the absorption can be applied with "spell" fixed effects that allow one effect for each unique combination of the multiple levels of indicators (e.g., each unique worker–firm combination) (Andrews, Schank, and Upward 2006).

However, there are some applications where the fixed effects are of direct interest, either by themselves or in conjunction with the effects of time-varying covariates. This is particularly true in education research, where there is growing interest in using large, longitudinal achievement databases with students linked to teachers and schools to estimate the effects of individual teachers on student achievement (Goldhaber and Hansen 2008; Harris and Sass 2006, 2007; Kane, Rockoff, and Staiger 2006; McCaffrey et al. 2004; McCaffrey et al. 2009) and increasing calls to use such estimates for high-stakes purposes such as pay or tenure decisions (Gordon, Kane, and Staiger 2006). Similarly, in labor economics, there is interest in estimating the effects of individual firms (Abowd, Creecy, and Kramarz 2002; Abowd, Kramarz, and Margolis 1999).

In cases where estimates of unit fixed effects are of interest, the parameterization matters and it is essential for the analyst to be able to control it. However, software for modeling with fixed effects (e.g., `areg`, `xtreg, fe`, or SAS® PROC GLM) all solve the problem of overparameterization by deleting the indicator variables for one or more units until the remaining variables are linearly independent. Although this parameterization suffices in allowing for estimation of the remaining model parameters, it changes the definition of the fixed-effects parameters for indicator variables to be contrasts between the units that were not deleted and the arbitrary holdout unit or units whose indicators were deleted.

Comparison to a holdout unit results in estimates that are not invariant to the chosen holdout unit, and estimates can change as a result of immaterial changes to the data, such as the labeling or ordering of units. In more complex settings and in models

where more than one indicator variable must be deleted to solve the overparameterization of fixed effects, it can be difficult to determine the specific interpretation of the reported contrasts. This situation is unacceptable in cases where obtaining estimates of the fixed-effects parameters is a primary objective. In addition, estimation of contrasts to arbitrary holdout units can result in instability of estimates across time or other settings due to differences in the holdouts. The problems of interpretation and instability potentially can be overcome by using linear combinations of model parameters to estimate individual units' contributions (e.g., combining the intercept and the individual-unit fixed effects to estimate the unit-specific mean, or post hoc mean, centering the estimated effects to estimate the effect of the unit relative to the average unit); however, the standard errors of such combinations are not easily obtained with available software.

Hence, there is a need for software that estimates the fixed effects using a parameterization that is interpretable, not capricious, and that provides standard errors for these estimates. A parameterization meeting these criteria is one in which effects for individual units are defined as deviations from the average unit or the average units within specified reference groups. These contrasts more directly correspond to the parameters of structural (causal) models; for example, in education settings, the notion of an "effective teacher" is a teacher whose students perform better with that teacher than they would with the average teacher in the population. These estimates are obtained by enforcing sum-to-zero constraints among the fixed effects in the model.

Users could code independent variables to correspond to the sum-to-zero parameterization and then include these as variables in standard linear model software. However, coding the independent variables is tedious, especially if multiple constraints are necessary to remove overparameterization. In addition, when there are many units or there are two levels of fixed effects (e.g., firms and workers or teachers and students), creating the complete matrix of independent variables can be computationally burdensome or infeasible.

To address the need for software designed to estimate fixed effects when they are of interest, we introduce the `felsdvregdm` command, which automatically parameterizes the fixed effects using sum-to-zero constraints. The command allows the user to define the collection of units to be compared (called "reference collections") and calculates the sum-to-zero parameterization so that the fixed-effects parameters are deviations from the reference collection means. For example, in the education setting, it is possible to calculate teacher fixed effects that are deviations from the average teacher in the school, the average teacher in the grade across schools, or any other grouping of teachers of interest to researchers. The program works with a single level of fixed effects (firms or teachers) and with two levels of effects (workers and firms or students and teachers). The program builds on the efficient computational methods of `felsdvreg` (Cornelissen 2008), which allow it to handle datasets with a large number of units and individuals.

`felsdvregdm` complements several commands that already exist in Stata to handle fixed effects. For a model with one level of fixed effects, `felsdvregdm` produces estimates and inferences for effects of time-varying covariates that are identical to `areg`,

`xtreg`, or `fese` (Nichols 2008). All four of these procedures will produce the same overall model fit and residual error because the alternative parameterizations of fixed effects do not affect the overall model fit. The only difference in the results from `felsdvregdm` and the other procedures is the estimates of the fixed-effects parameters and their standard errors. Similarly, for a model with two levels of fixed effects, the estimates and inferences for time-varying covariates and model fit from `felsdvregdm` are identical to those produced by `felsdvreg`, `a2reg` (based on Abowd, Creecy, and Kramarz [2002]) or `gpreg` (based on Guimarães and Portugal [2009]), with differences only in the estimates of the unit fixed-effects parameters and their standard errors.[2] In neither case does the fixed-effects parameterization used by `felsdvregdm` add any additional constraints to the model.

In this article, `felsdvregdm` is introduced and implemented in an example dataset from the Florida Education Data Warehouse in the context of educational achievement of students linked to teachers.

2 Sum-to-zero constraints

2.1 One level of fixed effects

We roughly follow the notation established by Cornelissen (2008). The basic model with a single level of fixed effects assumes that the outcome for a "person" i with time-varying covariates x_{it} associated with "unit" j at time t is given by

$$y_{it} = x_{it}^{'}\boldsymbol{\beta} + \psi_{j(i,t)} + \epsilon_{it}$$

where ϵ_{it} is a mean zero error term and there is a separate factor ψ_j for each unit, with the index $j(i,t)$ indicating the unit j to which person i is linked at time t. The fundamental problem with this model is that without additional assumptions, the average of the ψ_j cannot be distinguished from the average person outcome, or more generally, the model with separate factors for every unit is overparameterized and model parameters cannot be estimated uniquely.

The model for the outcomes from a typical sample of panel data from N persons, with a total of N^* person-time observations is given by

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{F}\boldsymbol{\psi} + \boldsymbol{\epsilon} \tag{1}$$

where $\boldsymbol{F}$ is an $N^* \times J$ incidence matrix consisting of only zeros and ones and satisfying $\boldsymbol{F}\mathbf{1}_J = \mathbf{1}_{N^*}$ where $\mathbf{1}_k$ is a k-vector of ones. That is, each observation is linked to exactly one of the J units. $\boldsymbol{X}$ is $N^* \times K$ containing time-varying covariates, as well as time effects or an overall intercept. Overparameterization is then characterized by the

2. The two-level model can also be fit via `areg` by explicitly including indicator variables for the unit fixed effects via the `xi` command. This will yield the same estimates of time-varying covariates and model fit as `felsdvregdm`, but it will not match the estimates for the fixed effects because the `xi` command automatically deletes dummy variables to identify the model. Using `areg` for two-level models will not be feasible when there are large numbers of units because of memory constraints and Stata's 11,000-variable matrix size limit.

matrix $(\boldsymbol{X}, \boldsymbol{F})$ being less than full column rank. Under the assumptions about $\boldsymbol{F}$, this will always be true when $\boldsymbol{X}$ contains either an intercept or separate means for every time point, because in either case it is possible to recover $\mathbf{1}_{N^*}$ as a linear combination of columns of $\boldsymbol{X}$ or columns of $\boldsymbol{F}$ separately, and thus the columns of $(\boldsymbol{X}, \boldsymbol{F})$ are not linearly independent. As noted previously, the most common approach to this overparameterization is to remove as many columns of $\boldsymbol{F}$ as there are linear constraints among the columns of $(\boldsymbol{X}, \boldsymbol{F})$.[3] In the simplest case, $\text{rank}\{(\boldsymbol{X}, \boldsymbol{F})\} = \text{rank}(\boldsymbol{X}) + J - 1$; that is, the introduction of $\boldsymbol{F}$ results in only one linear dependency. For instance, there is one linear dependency when $\boldsymbol{X}$ consists of only the intercept or includes only indicators for time-point means.

Our solution to the problem is to replace $\boldsymbol{F}$ with an $N^* \times (J-1)$ matrix $\boldsymbol{F}^*$ with associated parameter vector $\boldsymbol{\psi}^*$. $\boldsymbol{F}^*$ is obtained by dropping an arbitrary column j of $\boldsymbol{F}$ and setting all rows of $\boldsymbol{F}^*$ corresponding to observations linked to unit j to a vector of negative ones. That is, we define the effect of unit j to be $1 - \sum_{i=1,\ldots,J; i \neq j} \psi_i$ so that $\boldsymbol{\psi}^*$ contains $J-1$ free parameters rather than J free parameters because the elements of $\boldsymbol{\psi}^*$ sum to zero. Unlike the default holdout parameterization, these contrasts and their estimates are invariant to which unit-j column is eliminated from $\boldsymbol{F}$, and without loss of generality, we eliminate the first column of $\boldsymbol{F}$.[4] In this case, we can express $\boldsymbol{F}^* = \boldsymbol{F}\boldsymbol{C}$, where $\boldsymbol{C}$ is the $J \times (J-1)$ matrix $(-\mathbf{1}, \boldsymbol{I})^T$ (Venables and Ripley 2001).

More generally, analysts may be interested in controlling for differences among collections of firms or teachers. For example, firms may have classification factors such as size or sector that the analysts want to remove from the individual unit effects. In education applications, partitioning of teachers into comparison groups is commonplace. For datasets following a single cohort of students across grades, it is typical to compare teachers of the same grade or same group of grades (i.e., elementary versus middle grades). In more complex datasets, where multiple cohorts of students are followed and teachers are observed for multiple years across different cohorts, it may be desirable to estimate separate teacher-year effects for each teacher and to compare teachers within the same year and perhaps the same grade or grade range within year (McCaffrey et al. 2009).

In the context of estimating firm or teacher effects, controlling for a classification variable creates additional redundant parameters and requires additional constraints on the unit-effect parameters for identification. We cannot distinguish average performance for units in a group from the effects of being in the group. One approach that yields interpretable estimates for the firms and teachers is to parameterize the model so that units are compared within groups defined by the classification variables but not across those groups. In general, establishing the comparison groups is an analyst decision that depends on the substantive application.

3. Sometimes, depending on the specific routine and the order in which regressors are specified, elements of $\boldsymbol{X}$ rather than $\boldsymbol{F}$ may be dropped. This is equally problematic even though no elements of $\boldsymbol{F}$ are dropped because the estimates of the fixed effects contain the omitted intercept or time effect(s) and are thus still subject to arbitrary choices made by the regression routine.
4. This parameterization is invariant to the number of units, J, and is equally effective on a small or large scale.

We refer to such comparison groups as "reference collections" to avoid confusion with the "groups" described by Cornelissen (2008). Cornelissen's groups are relevant in models with two levels of fixed effects and are discussed in the next section. Specifically, we define reference collections as any partitioning of the J units into G mutually exclusive and exhaustive categories, and we parameterize the effects within each of these $g = 1, \ldots, G$ categories with sum-to-zero constrained effects. Thus there are $J - G$ parameters for the effects of the J units. If the columns of $\boldsymbol{F}$ are sorted by reference collection, then the $N^* \times (J - G)$ design matrix for the fixed effects is given by $\boldsymbol{F}^* = \boldsymbol{F}(\oplus \boldsymbol{C}_g)$ where $\boldsymbol{C}_g$ is the matrix $\boldsymbol{C}$ defined previously, restricted to reference collection g.

We would expect that analysts would almost always include reference collection means in the model, because the motivation for creating a reference collection will typically be to control for mean differences between those collections. However, some analysts might not follow this convention; in those cases, to ensure that our parameterization has the correct interpretation, we add those means to the model. We let $\boldsymbol{G}$ be the $N^* \times G$ incidence matrix such that $\boldsymbol{G}_{GI} = 1$ if the unit to which person-time observation i is linked is in reference collection g and $\boldsymbol{G}_{GI} = 0$ otherwise. The columns of $\boldsymbol{G}$ correspond to the reference collection means and the column space of $(\boldsymbol{F}^*, \boldsymbol{G})$ is identical to $\boldsymbol{F}$. We define $\boldsymbol{X}^* = (\boldsymbol{X}, \boldsymbol{G})$ with parameter $\boldsymbol{\beta}^*$, and as discussed in more detail later in the article, our routine uses $\boldsymbol{X}^*$ rather than the user-supplied $\boldsymbol{X}$ to ensure that our parameterization of unit fixed effects introduces no other changes to the model.[5] The resulting equation estimated in the case of one-level fixed effects is given by

$$\boldsymbol{Y} = \boldsymbol{X}^*\boldsymbol{\beta}^* + \boldsymbol{F}^*\boldsymbol{\psi}^* + \boldsymbol{\epsilon}$$

2.2 Two levels of fixed effects

We extend (1) to include person-level fixed effects using the following model:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{F}\boldsymbol{\psi} + \boldsymbol{D}\boldsymbol{\theta} + \boldsymbol{\epsilon} \tag{2}$$

where $\boldsymbol{D}$ is an $N^* \times N$ matrix of person (e.g., worker or student) fixed effects. Unlike the unit effects, in the vast majority of applications, these fixed effects are nuisance parameters used to control for person-level heterogeneity when estimating unit effects or the effects of time-varying covariates. `felsdvregdm` can implement both this two-level fixed-effects model or the one-level model (1). When fitting the two-level model, person-level fixed effects are treated as nuisance parameters and are absorbed using the within transformation (Cornelissen 2008).

The essential logic of our approach to parameterizing the unit effects is unchanged by the use of (2) and the within transformation to remove the person fixed effects. However, the elements of $\boldsymbol{X}^*$ and $\boldsymbol{F}^*$ reflect the within transformation. The transformed version of (2) after the reparameterization can be written as

5. If any of the columns $\boldsymbol{G}$ are in the column space of $\boldsymbol{X}$, they are deleted from the model during estimation.

$$\widetilde{\boldsymbol{Y}} = \widetilde{\boldsymbol{X}}^{*}\boldsymbol{\beta}^{*} + \widetilde{\boldsymbol{F}}^{*}\boldsymbol{\psi}^{*} + \widetilde{\boldsymbol{\epsilon}}$$

The $\sim$ reflects the within-person transformed data in which the person mean has been removed from each observation and which leads to the $\boldsymbol{D}$ matrix becoming the null matrix. $\widetilde{\boldsymbol{X}}^{*} = (\widetilde{\boldsymbol{X}}, \widetilde{\boldsymbol{G}})$ is an $N^{*} \times (K + G)$ matrix that includes the time-varying covariates and the reference collection indicators after the within transformation, and $\widetilde{\boldsymbol{F}}^{*}$ is the design matrix for the unit fixed effects after the sum-to-zero reparameterization and within-person transformation.

Often the inclusion of the person fixed effects results in no additional lost degrees of freedom beyond those already discussed and so no additional considerations are necessary. However, there are certain cases in which additional degrees of freedom may be lost. These are cases in which multiple disconnected "groups" or "strata" exist in the data (Abowd, Creecy, and Kramarz 2002; Cornelissen 2008; McCaffrey et al. 2009). Groups or strata are subgroups of units and persons that are disconnected from other subgroups in the sense that no person from that subgroup is ever linked to units outside the subgroup, and the units in the subgroup only ever link to persons within the subgroup. The simplest example can occur in education data: if students and teachers never switched schools, then each school and the students attending it would be disconnected groups.

In two-level models, groups must be considered when defining reference collections because they create additional linear dependencies among the columns of $\boldsymbol{D}$ and $\boldsymbol{F}$. If centering within a particular reference collection is desired but that reference collection spans two or more disconnected groups, sum-to-zero constraints within the reference collection would fail to identify the model parameters in the context of (2). This is because comparisons of units that are in the same reference collection but in different groups conflate differences between the groups with differences between the units. One remedy for this issue is to redefine the reference collections by intersecting them with group, i.e., splitting any reference collections that spanned groups into separate reference collections by group. Other alternatives might also be possible. The key is that for each reference collection, all the units must belong entirely to one and only one group, although groups can include multiple reference collections. The appropriate solution will depend on the substantive goals of the analysis. We discuss how `felsdvregdm` deals with the relationship between groups and reference collections for the two-level model in section 3.3, and we give an example of modeling with reference collections and groups in section 4.

An extreme case of grouping occasionally occurs in which a group contains a single firm (teacher) because every worker employed by the firm (student taught by the teacher) is linked only to this firm (teacher). Cornelissen (2008), refers to these persons (workers or students) associated with a single unit as "nonmovers" because they do not move across the units but stay at the same one. These nonmovers can occur in any group and do not contribute to the estimation of the unit effects in the two-level model. However, when the data for a unit comes only from nonmovers (i.e., the unit creates a single unit group), then the unit effect cannot be estimated in the two-level model. The

unit fixed effect is excluded from the model. The persons are included in the estimation with their fixed effects and can contribute to the estimation of the coefficients for the time-varying covariates.

3 Implementation of felsdvregdm

3.1 Syntax

`felsdvregdm` *varlist* [*if*] [*in*], `ivar(`*varname*`) jvar(`*varname*`) reff(`*varname*`) peff(`*name*`) feff(`*name*`) xb(`*name*`) res(`*name*`) mover(`*name*`) mnum(`*name*`) pobs(`*name*`) group(`*name*`) [grouponly onelevel noisily cholsolve nocompress feffse(`*name*`)]`

3.2 Options

`ivar(`*varname*`)` specifies the identification variable for the person-level effect, such as the person or student ID. `ivar()` is required.

`jvar(`*varname*`)` specifies the identification variable for the unit-level effect, such as the firm or teacher ID. `jvar()` is required.

`reff(`*varname*`)` specifies the identification variable for the reference collection. Reference collections specify collections of units to be compared with the unit mean for the collection. For two-level fixed-effects applications, reference collections must be nested within groupings or strata of connected units. `reff()` is required.

`peff(`*name*`)` specifies the name of the new variable in which to store the person-level effect.[6] `peff()` is required.

`feff(`*name*`)` specifies the name of the new variable in which to store the unit effect.[7] `feff()` is required.

`xb(`*name*`)` specifies the name of the new variable in which to store the linear prediction $\mathbf{X}'\mathbf{b}$. `xb()` is required.

`res(`*name*`)` specifies the name of the new variable in which to store the residual. `res()` is required.

`mover(`*name*`)` specifies the name of the new variable in which to store an indicator variable for whether the person is observed with multiple firms or teachers. `mover()` is required.

`mnum(`*name*`)` specifies the name of the new variable in which to store the number of movers per unit. In the two-level model, units for which `mnum()` is zero do not have identified effects. `mnum()` is required.

6. This is referred to as the first effect in the `felsdvreg` documentation.
7. This is referred to as the second effect in the `felsdvreg` documentation.

`pobs(`*name*`)` specifies the name of the new variable in which to store the number of observations per person. Persons for whom only one observation is available will have person effects computed with a residual of zero for the two-level model. `pobs()` is required.

`group(`*name*`)` specifies the name of the variable in which to store the group (strata) identifier calculated during the routine. If the variable does not already exist in the data, it will be created; if there is an existing variable *name*, it will be overwritten. `group()` is required.

`grouponly` causes `felsdvregdm` to run only the grouping algorithm and produce group indicator variables. No estimates are computed in this case.

`onelevel` omits the person fixed effects, resulting in the one-level fixed-effects model being fit.

`noisily` provides more detailed outputs with tables and summary statistics on the number of movers, units, and persons, and the mobility variable.

`cholsolve` directs Stata to use the `cholsolve()` function to invert the moment matrix instead of using the default `invsym()` function. This may result in greater internal precision.

`nocompress` specifies not to compress the dataset; by default, the `compress` command is implemented to save memory.

`feffse(`*name*`)` specifies the name of the new variable in which to store the standard errors of the estimated unit fixed effects.

3.3 Remarks

Because `felsdvregdm` extends the functionality of `felsdvreg` and uses syntax and its estimation methods as its base, the procedures for the two commands are very similar. In particular, `felsdvregdm` requires the same specification of person-related variables as `felsdvreg`. This is true for both one- and two-level applications, even though in principle, several variables might only apply to models with two levels of fixed effects (e.g., `ivar()`, `peff()`, `mover()` and `pobs()`). All variables are required regardless of whether one or two levels are specified because the computational methods developed in `felsdvreg` and used by `felsdvregdm` rely on both the unit and person levels of indicators to save time and memory during estimation. Also consistent with `felsdvreg`, when specifying the `grouponly` option, all the variable names need to be defined even though only the `group()` variable will be modified and saved.

Although the syntax of `felsdvregdm` and `felsdvreg` are very similar, there are several important differences to note between the procedures:

- The `reff()` option, which is not part of `felsdvreg`, is required by `felsdvregdm`. This variable specifies the reference collections inside which the user wishes to impose sum-to-zero constraints among the unit effects.

As noted in the previous section, maintaining correspondence between the models in the original parameterization and our sum-to-zero constrained parameterization requires that $\boldsymbol{G}$, or something equivalent to it (in the sense of having identical column space), be included in the model. We encourage users to create indicator variables corresponding to the levels of the `reff(`*varname*`)` variable and to include these as regressors in the model, if they are not already part of the model. For example, in the setting where teachers of the same grade are being compared in the reference collections, the explanatory variables should contain dummy variables for each of the grades. In the case of a single reference collection, where all of the units are being compared to one another, the regression should include a constant term. If the user-specified explanatory variables do not span $\boldsymbol{G}$, `felsdvregdm` will add variables corresponding to $\boldsymbol{G}$ to the explanatory variable list as well as to the dataset after estimation, where they are labeled as `D`*varname*`_1`, ..., `D`*varname*`_`G, where *varname* is the name of the `reff()` variable in the function call.[8]

- Similarly to `felsdvreg`, `felsdvregdm` checks for collinearity among the columns of $\boldsymbol{X}^*$, the matrix of explanatory variables (augmented by $\boldsymbol{G}$, if necessary), and drops redundant variables if any exist. That is, $\boldsymbol{X}^*$ (or $\widetilde{\boldsymbol{X}}^*$ in the two-level model) is restricted to a set of its linearly independent columns. `felsdvregdm` further checks that the columns $(\boldsymbol{X}^*, \boldsymbol{F}^*)$ [or $(\widetilde{\boldsymbol{X}}^*, \widetilde{\boldsymbol{F}}^*)$] are all linearly independent and aborts estimation with an error message if they are not because any collinearity among the columns would prevent the unambiguous interpretation of the fixed-effects estimates, defeating the purpose of our reparameterization.

- Like `felsdvreg`, `felsdvregdm` fits a two-level model, but unlike `felsdvreg`, it also provides the option of fitting a one-level fixed-effects model with the sum-to-zero constraint using the `onelevel` option.

- For the two-level model, `felsdvregdm` runs a grouping algorithm prior to estimation to determine groups, stores the results in the `group()` variable, and checks that no reference collection is ever associated with more than one group. If this test fails, the program aborts and returns an error message, because in this case, the sum-to-zero constraints within reference collection are insufficient to identify all the model parameters. In practice, we suggest that prior to attempting to fit a two-level model, users call `felsdvregdm` with the `grouponly` option. In addition, users should use the group information stored in the `group()` variable to check that reference collections do not contain multiple groups and as necessary redefine reference collections to be within group boundaries and remain substantively meaningful.

- Many options that were available in `felsdvreg` can no longer be implemented. Because automatic selection of parameterizations might not coincide with the substantive interests of users, the `takegroup` option of `felsdvreg` has been elimi-

8. On occasion, the user-specified $\boldsymbol{G}$ will be replaced with variables that are equivalent to it. Stata developers are working to fix this inconsistency.

nated. Similarly, because the matrix of explanatory variables is augmented by $\boldsymbol{G}$, the reference collection fixed-effects matrix, the `cons` option is no longer possible.

- The F tests conducted by `felsdvregdm` are different from those conducted by `felsdvreg`. In particular, in `felsdvregdm` we made the decision to test the unit effects only within reference collections rather than testing the unit effects along with the reference collection means. That is, if there are G reference collections and J total unit effects, we include the G reference collection means in the reduced model and test the significance of the remaining $J-G$ centered unit effects. This choice was consistent with the notion that a user may choose a particular set of reference collections because between-collections variability may not be attributable to units. For example, it is common to compare teachers within grades, because test scores may vary across grades due to scaling of tests, which users generally would not want to include as part of the teacher effects.

  For both the one-level and two-level models, `felsdvregdm` uses a model that includes $\boldsymbol{X}$ for the first set of F tests and $\boldsymbol{X}^* = (\boldsymbol{X}, \boldsymbol{G})$ for the second set of F tests as the reduced model.[9] For the one-level model, it tests the joint significance of the within-collection unit effects relative to the reduced model. For the two-level model, it reports three tests: the joint significance of the person and unit effects relative to the reduced model, the significance of the unit effects conditional on the person effects and the covariates, and the significance of the person effects conditional on the unit effects and the covariates.

- The `normalize` option in `felsdvreg` normalizes the group means of the unit effects to zero. This normalization is equivalent to the case in `felsdvregdm` if all the observations are specified to be in a single reference collection. However, in many situations, the user may wish to compare smaller collections of units, in which case `felsdvregdm` is more desirable. In addition, the `normalize` option does not adjust the standard errors for the normalization and continues to report standard errors that are relative to an arbitrary holdout teacher. For more information on the impact of using an arbitrary holdout teacher to calculate standard errors, see the discussion surrounding table 1 in section 4.1.

- Forthcoming in later versions of `felsdvregdm` are the `robust` and `cluster(`*varname*`)` options, which compute robust and clustered standard errors; the `noadji` and `noadjj` options, which adjust the degrees of freedom for the clustered standard errors; and the `hat(`*varlist*`)` and `orig(`*varlist*`)` options, which allow for two-stage least-squares estimation.

4 Example

We illustrate the use of `felsdvregdm` with an example of longitudinally linked student–teacher data from the Florida Education Data Warehouse. The data consist of a single

9. For the two-level model, time-invariant person covariates that were included among the list of independent variables are not included in the reduced model.

cohort of 9,283 students from an anonymous district followed from grades 4 to 8. In each grade, the students are linked to their mathematics teachers, and there are a total of 1,360 unique teacher-year units in the dataset.[10] For the purposes of the example, we focus on estimating separate effects for each teacher for each school year. Any teacher who teaches students from the cohort in multiple years through changes in teaching assignment across years will have a distinct effect for each year they teach the cohort; hence, our units in this study are teacher-years rather than individual teachers. To simplify our presentation, we refer to our estimated effects as teacher effects although they are more precisely teacher-year effects for the small number of teachers who teach the cohort in multiple years. Because the data include only one cohort of students, there is an exact mapping of the grade level of the cohort and year. Teacher effects from any given year correspond to effects of all the teachers instructing the students in mathematics during one grade level. For instance, the year one teacher effects are the effects of the fourth grade mathematics teachers for the cohort, and the year two effects are the effects of the fifth grade mathematics teachers, and so on.

The outcome variable of interest is `nrtgain`, measured at each grade and equal to the student's gain (current year scale score minus prior year scale score) on a statewide mathematics exam. The data also include five time-invariant student-level indicators of basic demographics: `female`, `white`, `black`, `asian`, and `hispanic`. Finally, the data include the time-varying student-level covariate `nonstrucmove` indicating whether the student made a school move between years that was not also undertaken by a significant proportion of his prior-year schoolmates (i.e., not due to matriculation from elementary to middle school). This variable proxies for student family mobility that may be related to achievement.

In this section, we demonstrate the use of `felsdvregdm` to fit two different types of models for student mathematics achievement gains as a function of teacher and student inputs: a one-level fixed-effects model with teacher fixed effects and both time-invariant and time-varying student covariates, and a two-level model with both teacher and student fixed effects as well as the time-varying student covariate. For the one-level model, we show how inferences about the relationships of the student covariates to the outcome obtained by `felsdvregdm` are identical to those reported by `areg`. For the two-level model, we compare `felsdvregdm` to `felsdvreg`, showing that the estimates of covariates are identical but that the teacher effects from `felsdvregdm` have better properties. We also demonstrate how the issue of disconnected groups needs to be addressed in `felsdvregdm`.

Student gains can differ by grade in part because of the structure of the tests and general trends in student learning. We want to control for these grade-level differences in our model and avoid conflating them with our teacher effects. Consequently, for both the one-level and the two-level models, we include grade means in the model and we define reference collections for the teacher fixed effects at the level of individual grades; i.e., grade 4 teachers are one reference collection, grade 5 teachers are another, etc.,

10. We combined students from six teachers into two "teachers" to collapse three single-student strata into one stratum to improve the exposition of the example.

for a total of five reference collections. There are 412, 399, 178, 197, and 177 teachers in grades 4 through 8, respectively, with the smaller numbers in grades 6 through 8 reflecting the period structure of middle schools in which there are fewer teachers and more students per teacher. As noted above, because grades map uniquely to years and there is a distinct teacher effect for every teacher-year unit, reference collections defined by grades fully partition our teacher effects into disjoint sets. If we were interested in estimating a single effect for every teacher that is constant across years for teachers who teach the cohort in multiple years or grades, then reference collections defined by grade would not properly partition teachers and an alternative definition would be required.

4.1 One level of fixed effects

The model used in this example regresses gains in students' mathematics achievement, `nrtgain`, on teacher fixed effects and all student-level covariates. This is an instance of (1) with $\boldsymbol{Y}$ being the vector of annual mathematics gain scores, $\boldsymbol{X}$ including grade means and all student-level variables, and $\boldsymbol{F}$ representing the teacher effects.

This model is fit using `felsdvregdm` as follows:

```
. felsdvregdm nrtgain female white black asian hispanic nonstrucmove grade4
> grade5 grade6 grade7 grade8, ivar(student) jvar(teachergrd) reff(gradelvl)
> feff(teachfe_dm) peff(stufe_dm) feffse(teachse_dm) mover(mover) group(group)
> xb(xb) res(resid) mnum(mnum) pobs(pobs) onelevel

Memory requirement for moment matrices in GB:
  .01502616

Computing generalized inverse, dimension: 1365
  Start:  9 Dec 2009 10:15:40
  End:    9 Dec 2009 10:15:46

N=34326
```

	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
female	.3093813	.2556106	1.21	0.226	-.1916247	.8103874
white	-.0732443	.8370731	-0.09	0.930	-1.713938	1.567449
black	.2331433	.8516812	0.27	0.784	-1.436182	1.902469
asian	3.080624	1.082118	2.85	0.004	.9596337	5.201615
hispanic	1.448387	1.031368	1.40	0.160	-.5731313	3.469905
nonstrucmove	.0243751	.3639665	0.07	0.947	-.6890122	.7377625
grade4	18.43987	.8813575	20.92	0.000	16.71238	20.16736
grade5	20.66716	.8959979	23.07	0.000	18.91097	22.42335
grade6	11.972	1.242489	9.64	0.000	9.536679	14.40733
grade7	15.04787	1.232244	12.21	0.000	12.63263	17.46312
grade8	19.07987	1.260358	15.14	0.000	16.60953	21.55022

```
F-test that firm effects are equal to zero:   F(1354,32961)=2.558 Prob > F = .0031
```

The student identifiers are passed using `ivar()`, and the teacher identifiers are passed using `jvar()`.[11] The reference collection is denoted by `gradelvl`, which takes on values of 4 through 8 and is specified in `reff()`. The `felsdvregdm` call includes indicator variables corresponding to each grade level in the list of explanatory variables. The remaining options denote variables created to store the teacher and student fixed effects, the standard errors, and other variables of interest, similar to `felsdvreg`. The `onelevel` option was included to fit the one-level model that does not include student fixed effects. As discussed previously, if the reference collection indicator variables had not been included in the explanatory variable list, `felsdvregdm` would have added them and provided a message noting this addition and the names of the added variables. The fitted model would be identical to the one above in all other respects.

We find that student covariates generally are not related to student gains in mathematics achievement; only the `asian` variable is significant, but there are large differences in grade-level means. Comparing teachers across grade levels could conflate test differences and other factors with teacher effects. Hence, we need to compare teachers within grade level, as was accomplished via our choice of reference collection. The significant value for the F test indicates that within grade levels, there are meaningful differences among teachers, which is consistent with literature on teacher effects.

The Stata `areg` command is a commonly used alternative to `felsdvregdm` with the `onelevel` option for fitting the one-level (1). For example, `areg nrtgain female white black asian hispanic nonstrucmove grade4 grade5 grade6 grade7 grade8, absorb(teachergrd)` is an alternative command for fitting the model above. The `predict` postestimation command with the `d` or `xbd` option will recover the teacher effects without standard errors. Another alternative for the one-level model is the `fese` command, which provides estimates for both the teacher effects and the standard errors (Nichols 2008).[12]

`felsdvregdm` and `areg` (or `fese`) produce similar results but with some subtle, important differences. Both models produce the same estimates and standard errors for the student-level variables (`female`, `white`, `black`, `asian`, `hispanic`, and `nonstrucmove`) and both models give the same overall model fit, test for the overall model, residuals, and root mean squared error. However, because the grade indicators (`grade4`–`grade8`) are collinear with the absorbed teacher effects, `areg` drops these variables from the model and makes them part of the estimated teacher effects. In particular, `predict`'s `d` option yields teacher-effect estimates that equal the mean for each teacher, adjusted for the student-level covariates, less the grand mean adjusted for student-level covariates. `predict`'s `xbd` option yields teacher-effect estimates that equal the mean for each teacher, adjusted for the student-level covariates. The estimates make no adjustment

11. Even though this model contains one level of fixed effects, both the `ivar()` and the `jvar()` variables need to be specified. This is because even with one fixed-effects model, the memory saving computation of the fixed-effects parameters in `felsdvregdm` is possible by taking advantage of the panel structure of the dataset.

12. Similarly, `xtreg, fe` could be used to estimate the coefficients on the time-varying and time-invariant variables and has options to recover the teacher effects with the `predict` postestimation command using the `u` or `xbu` option. Again, `xtreg` does not provide the control over that parameterization that `felsdvregdm` offers, and standard errors are difficult to estimate in this case.

for the grade-level means and are not parameterized to sum to zero overall or within grade level. As a result, the teacher-effect estimates produced by `areg` with `predict`'s `d` option differ from the estimates from `felsdvregdm` within each grade level by a constant equal to the adjusted mean of the outcome variables for the entire sample minus the grade-level mean of the adjusted teacher means (table 1). `fese` produces the same estimates as `areg` with `predict`'s `xbd` option, and both are estimates of the teacher means assuming the grade-level means and the overall mean are part of the parameter of interest. These are unbiased estimates of the means given the model, but they may not truly be the parameters of interest and would be inappropriate for empirical Bayes (shrinkage) estimation.

Table 1. Mean differences in teacher fixed-effects estimates, by reference collection, produced by `areg` and `felsdvreg`.

Grade	Difference
4	−1.966
5	−4.193
6	4.502
7	1.426
8	−2.606

Whether the grade-level means should be part of the teacher effects cannot be determined by the data and must be determined by the user. `felsdvregdm` provides users with the option of treating the grade-level mean differences as an external source of error and estimating the teacher effects relative to the mean of their grade-level peers. `areg` and `fese` provide the alternative approach. It would be possible to post hoc center the estimate from `areg` or `fese` by grade level, but the standard errors would not be provided as they are with `felsdvregdm`.

4.2 Two levels of fixed effects

The model used in this example regresses gains in students' mathematics achievement, `nrtgain`, on teacher fixed effects, student fixed effects, and the time-varying student covariate `nonstrucmove`. Compared with the previous example, this model uses fixed effects for each individual student rather than time-invariant student covariates, to control for student heterogeneity. This is an instance of (2). Because the model includes two levels of fixed effects, we compare `felsdvregdm` with `felsdvreg` rather than `areg`.

As discussed previously, issues of grouping must be considered in the two-level model. Because of `felsdvregdm`'s goal to maintain strict control over the interpretation of the unit fixed effects, reference collections must not attempt to compare teachers in disconnected strata. To check the grouping structure, we first run `felsdvregdm` with the `grouponly` option:

```
. felsdvregdm nrtgain nonstrucmove grade4 grade5 grade6 grade7 grade8,
> ivar(student) jvar(teachergrd) reff(gradelvl) feff(teachfe) peff(stufe)
> feffse(teachse) mover(mover) group(groupout) xb(xb) res(resid) mnum(mnum)
> pobs(pobs) grouponly
Note: You specified ´grouponly´. Only the group variable was modified and saved.
      No estimates were produced.

Groups of firms connected by worker mobility:
```

	Person-years	Persons	Movers	Firms
groupout	N(__000000)	N(__000009)	sum(__00000D)	N(__000008)
1	34,320	9,280	9280	1,357
2	6	3	3	2
Total	34,326	9,283	9283	1,359

The results of the grouping algorithm are stored in `groupout`. These data contain two groups: one with the vast majority of the observations and one with three students and two teachers. Given the groups and the reference collections, we modify the reference collections to have the five grade-specific collections in the large group and one collection for both teachers in the small group.

The output below shows the composition of the six reference groups and the results from fitting the model using `felsdvregdm` with the new reference collections:

```
. egen newreff = group(gradelvl) if groupout==1
(6 missing values generated)
. replace newreff = 6 if groupout==2
(6 real changes made)
. preserve
. collapse groupout (sum) freq, by(newreff gradelvl)
. order newreff
. list, noobs
```

newreff	gradelvl	groupout	freq
1	4	1	6940
2	5	1	7622
3	6	1	6991
4	7	1	6894
5	8	1	5873
6	7	2	3
6	8	2	3

```
. restore
. quietly xi i.newreff, noomit
```

```
. felsdvregdm nrtgain nonstrucmove female white black asian hispanic
> _Inewreff_1  _Inewreff_2 _Inewreff_3 _Inewreff_4 _Inewreff_5 _Inewreff_6,
> ivar(student) jvar(teachergrd) reff(newreff) feff(teachfe) peff(stufe)
> feffse(teachse) mover(mover) group(groupout) xb(xb) res(resid) mnum(mnum)
> pobs(pobs)
The group variable (called groupout) that exists in the dataset will be replaced!
note: _Inewreff_6 dropped because of collinearity

Some variables were added or dropped from the independent variables
Reference collection fixed effects may not be sufficiently defined,
Variables could be collinear due to multiple groups in the dataset,
Or some variables may be collinear with the second level of fixed effects
The following variable(s) were added to the explanatory variables

The following variable(s) were ignored or dropped from the explanatory
variables
 Dnewreff_6 Dnewreff_5 Dnewreff_4 Dnewreff_3 Dnewreff_2 Dnewreff_1 _Inewreff_5
> hispanic asian black white female

Memory requirement for moment matrices in GB:
  .01489488

Computing generalized inverse, dimension: 1358
  Start:  9 Dec 2009 10:16:18
  End:    9 Dec 2009 10:16:24

N=34326
```

	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
nonstrucmove	.346897	.4944743	0.70	0.483	-.6223043	1.316098
_Inewreff_1	-.0394583	1.46098	-0.03	0.978	-2.903073	2.824156
_Inewreff_2	2.52157	1.498037	1.68	0.092	-.4146783	5.457819
_Inewreff_3	-6.279251	1.882227	-3.34	0.001	-9.968537	-2.589965
_Inewreff_4	-4.169235	1.853738	-2.25	0.025	-7.80268	-.5357898

```
F-test that person and firm effects are zero: F(10635,23685)=.601 Prob > F =.7295
F-test that person effects are equal to zero: F(9281,23685)=.38 Prob > F = 1
F-test that firm effects are equal to zero:   F(1353,23685)=2.196 Prob > F = 0
```

In general, in a two-level model, one reference-collection indicator will be dropped per group because of collinearity between the set of student fixed effects for students in each group and the set of reference collection means in each group (both sets add to an indicator for the group). In this example, the indicator for reference collection 6 is dropped for the small group and the indicator for reference collection 5 is dropped for the large group. However, the teacher effects are identified in both of the groups.

The crucial difference between `felsdvregdm` and other fixed-effects routines is in the estimated teacher effects. To demonstrate this, we calculated the mean estimated teacher effect, as well as the mean standard error of these estimates, by reference collection for both `felsdvregdm` and `felsdvreg`. We did this for two different datasets: the one used in the example reported above, in which the teacher with the lowest ID number has a large positive effect and many linked students, and an alternative version in which the teacher ID numbers were deliberately changed so that the teacher with the lowest ID number has a large negative effect and few linked students. These purposeful

manipulations of the teacher ID numbers are designed to demonstrate the impacts on both the estimates and their standard errors of changing the holdout teacher used by `felsdvreg`. The results are given in table 2.

Table 2. Mean estimated teacher effects by reference collection, with mean standard errors of these estimates by reference collection in parentheses. The number following the procedure name refers to the dataset used for estimation, where only the holdout teacher identifier was changed.

Reff Coll	**felsdvregdm 1**	**felsdvregdm 2**	**felsdvreg 1**	**felsdvreg 2**
1	0.000 (8.216)	0.000 (8.216)	−41.844 (11.926)	74.502 (37.146)
2	0.000 (8.649)	0.000 (8.649)	−39.283 (12.447)	77.063 (37.499)
3	0.000 (13.716)	0.000 (13.716)	−48.084 (17.222)	68.262 (40.058)
4	0.000 (14.703)	0.000 (14.703)	−45.974 (18.092)	70.372 (40.615)
5	0.000 (13.697)	0.000 (13.697)	−41.805 (17.198)	74.542 (40.065)
6	0.000 (10.385)	0.000 (10.385)	−13.667 (20.770)	−13.667 (20.770)

As intended, the mean teacher effect by reference collection for `felsdvregdm` is zero, regardless of how the teacher identifiers are labeled. For `felsdvreg`, the mean teacher effect is strongly negative for each reference collection in the first dataset, where the holdout teacher was above average, and strongly positive in the second dataset, where the holdout teacher was well below average. The estimated teacher effects from the two routines are perfectly correlated within reference collection (not shown); however, the parameterization used by `felsdvregdm` removes the instability caused by changing the holdout. An additional difference is that the fixed effect is not estimated for the holdout teacher in `felsdvreg`, and the standard error for this teacher is missing. `felsdvregdm` reports fixed effects and standard errors for all the teachers.

Because the teacher effects estimated by `felsdvregdm` are invariant to the labeling of teacher identifiers, the standard errors for those effects reported by `felsdvregdm` are also invariant to the labeling of the teacher identifiers. The standard errors of the effects reported by `felsdvreg` vary dramatically depending on the holdout teacher, with the average standard errors for the second dataset being much larger because the holdout teacher had very few students. Both sets of standard errors are larger than those reported by `felsdvregdm` because they are for contrasts of each teacher relative to the holdout instead of relative to the group mean, the former being a less precise comparison.

It would be easy to recover the desired mean-centered teacher effects reported by `felsdvreg` by post hoc centering; however, the standard errors of those contrasts cannot easily be obtained because they are a function of the full design matrix and not simple functions of the standard errors reported by `felsdvreg`. Having the appropriate standard errors is essential to many post hoc inferences about teachers made from the model. For example, in education applications, it is commonly of interest to estimate whether each teacher's effect is statistically significantly different from its

reference collection mean. These hypothesis tests are a straightforward function of the estimates and standard errors provided by `felsdvregdm` but are not easily conducted with the output from `felsdvreg`. In addition, it is often of interest to use the estimates and standard errors to estimate the true between-teacher variability in effectiveness within reference groups and then subsequently use this estimated variance to conduct empirical Bayes (shrinkage) estimators (Boyd et al. 2008; Koedel and Betts 2005; Jacob and Lefgren 2008; McCaffrey, Han, and Lockwood 2008). A simple method of moments estimator of this variance component is obtained by subtracting the average squared standard error of the estimates from the sample variance of estimates. For example, the average method of moments estimate of the between-teachers variance across reference collections obtained by the `felsdvregdm` output is 0.03, which is roughly consistent with values commonly reported in the teacher effectiveness literature.[13] Applying the same computations to the two different `felsdvreg` specifications yields infeasible values of -0.02 and -0.78, since the sample variance of the estimated effects is biased because of the correlation resulting from contrasting every estimated effect to the same holdout. Therefore, estimates and standard errors provided by `felsdvregdm` are appropriate for these computations, but those provided by `felsdvreg` are not.

5 Conclusion

As linked longitudinal datasets from various fields become more available, the desire to analyze these data to estimate the effects of individual units is poised to increase. These models can be computationally challenging, particularly when multiple levels of fixed effects are specified, and existing software routines for dealing with these challenges are generally not designed to provide estimates that are useful for inferences about individual fixed effects. Arbitrary decisions about how to deal with the lack of full identification of the fixed effects is at odds with the desire for unambiguous inferences about those effects. Sum-to-zero constrained effects within reference collections as implemented in `felsdvregdm` is an intuitive way for analysts to specifically control the parameterization of fixed effects when those effects are of interest and should prove valuable in many applications.

Although we implemented sum-to-zero constrained effects within reference collections in the context of longitudinal data, the logic of the parameterization applies to other types of models and data structures that are not addressed by `felsdvregdm`. For example, in education research, some analysts fit models where current year test scores or gain scores are modeled as a function of current teacher indicators and other student-level covariates such as demographic variables and prior test scores (McCaffrey, Han, and Lockwood 2008; Rothstein 2007). That is, rather than treating the data as panel data with the sequence of outcomes all being on the left-hand side of the model equation, these models fit the regression of the current outcome on past out-

13. Let $\widehat{\phi}_{gi}$ and se_{gi} equal the estimated effect and standard error of that effect for teacher $i = 1, \ldots, N_g$ of grade $g = 4, \ldots, 8$. The method of moment estimator for the between-teachers variance for teachers in grade g is $v_g = \sum_i (\widehat{\phi}_{gi} - \overline{\widehat{\phi}}_{g.})^2/(N_g - 1) - \sum_i se_{gi}^2/N_g$, where $\overline{\widehat{\phi}}_{g.}$ equals the average of the estimates for grade g. We report the average of v_g across grade levels.

comes and other predictors. Our parameterization is equally desirable to this type of model, but `felsdvregdm` is designed for data with separate records for each person and time point rather than for data with one record per person and observations from prior time points included as additional fields instead of additional records. Future work to implement sum-to-zero constrained effects within reference collections in the context of cross-sectional models or other applications with similar data structures would be valuable.

6 References

Aaronson, D., L. Barrow, and W. Sander. 2007. Teachers and student achievement in the Chicago public high schools. *Journal of Labor Economics* 25: 95–135.

Abowd, J. M., R. H. Creecy, and F. Kramarz. 2002. Computing person and firm effects using linked longitudinal employer–employee data. Technical Paper No. TP-2002-06, Center for Economic Studies, U.S. Census Bureau. http://lehd.did.census.gov/led/library/techpapers/tp-2002-06.pdf.

Abowd, J. M., F. Kramarz, and D. N. Margolis. 1999. High wage workers and high wage firms. *Econometrica* 67: 251–333.

Abowd, J. M., F. Kramarz, and S. Roux. 2006. Wages, mobility and firm performance: Advantages and insights from using matched worker–firm data. *Economic Journal* 116: F245–F285.

Andrews, M., T. Schank, and R. Upward. 2006. Practical fixed-effects estimation methods for the three-way error-components model. *Stata Journal* 6: 461–481.

Boyd, D., P. Grossman, H. Lankford, S. Loeb, and J. Wyckoff. 2008. Who leaves? Teacher attrition and student achievement. NBER Working Paper No. 14022, National Bureau of Economic Research. http://www.nber.org/papers/w14022.

Clotfelter, C., H. Ladd, and J. Vigdor. 2006. Teacher–student matching and the assessment of teacher effectiveness. NBER Working Paper No. 11936, National Bureau of Economic Research. http://www.nber.org/papers/w11936.

Cornelissen, T. 2008. The Stata command felsdvreg to fit a linear model with two high-dimensional fixed effects. *Stata Journal* 8: 170–189.

Goldhaber, D., and M. Hansen. 2008. Is it just a bad class? Assessing the stability of measured teacher performance. CPRE Working Paper #2008-5, Center on Reinventing Public Education. http://www.crpe.org/cs/crpe/view/csr_pubs/249.

Gordon, R., T. J. Kane, and D. O. Staiger. 2006. Identifying effective teachers using performance on the job. Discussion Paper 2006-01, The Brookings Institution. http://www.brookings.edu/papers/2006/04education_gordon.aspx.

Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice Hall.

Guimarães, P., and P. Portugal. 2009. A simple feasible alternative procedure to estimate models with high-dimensional fixed effects. IZA Discussion Paper No. 3935, Institute for the Study of Labor (IZA). http://ideas.repec.org/p/iza/izadps/dp3935.html.

Harris, D. N., and T. R. Sass. 2006. Value-added models and the measurement of teacher quality. Unpublished manuscript.

———. 2007. What makes for a good teacher and who can tell? Manuscript, Florida State University, Tallahassee.

Jacob, B. A., and L. Lefgren. 2008. Can principals identify effective teachers? Evidence on subjective performance evaluation in education. *Journal of Labor Economics* 26: 101–136.

Kane, T. J., J. E. Rockoff, and D. O. Staiger. 2006. What does certification tell us about teacher effectiveness? Evidence from New York City. Unpublished manuscript.

Koedel, C., and J. Betts. 2005. Re-examining the role of teacher quality in the educational production function. Working Paper 0708, Department of Economics, University of Missouri. http://ideas.repec.org/p/umc/wpaper/0708.html.

McCaffrey, D., B. Han, and J. R. Lockwood. 2008. From data to bonuses: A case study of the issues related to awarding teachers pay on the basis of their students' progress. NCPI Working Paper No. 2008-14, National Center for Performance Incentives.

McCaffrey, D. F., J. R. Lockwood, D. Koretz, T. A. Louis, and L. Hamilton. 2004. Models for value-added modeling of teacher effects. *Journal of Educational and Behavioral Statistics* 29: 67–101.

McCaffrey, D. F., T. R. Sass, J. R. Lockwood, and K. Mihaly. 2009. The intertemporal variability of teacher effect estimates. *Education Finance and Policy* 4: 572–606.

Menezes-Filho, N. A., M.-A. Muendler, and G. Ramey. 2008. The structure of worker compensation in Brazil, with a comparison to France and the United States. *Review of Economics and Statistics* 90: 324–346.

Nichols, A. 2008. fese: Stata module to calculate standard errors for fixed effects. Statistical Software Components S456914, Department of Economics, Boston College. http://ideas.repec.org/c/boc/bocode/s456914.html.

Rockoff, J. E. 2004. The impact of individual teachers on student achievement: Evidence from panel data. *American Economic Review: Papers and Proceedings* 94: 247–252.

Rothstein, J. 2007. Do value-added models add value? Tracking, fixed effects, and causal inference. Working Paper 1036, Center for Economic Policy Studies, Princeton University. http://ideas.repec.org/p/pri/cepsud/1036.html.

SAS Institute, Inc. 2004. *SAS 9.1.3 Help and Documentation*. Cary, NC: SAS Institute.

Venables, W. N., and B. D. Ripley. 2001. *Modern Applied Statistics with S-plus*. 3rd ed. New York: Springer.

Wooldridge, J. M. 2002. *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.

About the authors

Kata Mihaly is an associate economist at RAND. Her research on education focuses on value-added models of teacher quality and the effect of peers on student achievement. Her other research interests include peer effects on adolescent substance abuse and the role of social networks on the decision of adults to save for retirement.

J. R. Lockwood is a senior statistician at RAND. His education research has focused on longitudinal modeling of student achievement, value-added models for estimating teacher effects, and experimental and quasi-experimental methods in educational evaluation. He has led two projects to develop enhanced statistical models for estimating teacher effects and to develop computational methods for implementing complex models with large datasets.

Daniel F. McCaffrey is a senior statistician at RAND, where he holds the PNC Chair in Policy Analysis. His current research focuses on value-added modeling for the estimation of teacher effects and the use of those estimates in teacher evaluation and school reform. He is currently working with the National Center on Performance Incentives to study the effects of using value-added measures to determine teacher compensation in two randomized trials. He is also working on several projects to study the relationship between value-added and other forms of teacher evaluations. His other research interests include causal modeling, primarily with application to adolescent substance abuse treatment.

Tim Sass is a professor of economics at Florida State University, where he teaches courses in applied microeconomics at the undergraduate and graduate levels. He is also a member of the National Center for Analysis of Longitudinal Data in Education Research. His current research focuses on various aspects of education policy, including teacher quality, charter schools, and peer effects.

The Stata Journal (2010)
10, Number 1, pp. 104–124

Creating synthetic discrete-response regression models

Joseph M. Hilbe
Arizona State University
and Jet Propulsion Laboratory, CalTech
Hilbe@asu.edu

Abstract. The development and use of synthetic regression models has proven to assist statisticians in better understanding bias in data, as well as how to best interpret various statistics associated with a modeling situation. In this article, I present code that can be easily amended for the creation of synthetic binomial, count, and categorical response models. Parameters may be assigned to any number of predictors (which are shown as continuous, binary, or categorical), negative binomial heterogeneity parameters may be assigned, and the number of levels or cut points and values may be specified for ordered and unordered categorical response models. I also demonstrate how to introduce an offset into synthetic data and how to test synthetic models using Monte Carlo simulation. Finally, I introduce code for constructing a synthetic NB2-logit hurdle model.

Keywords: st0186, synthetic, pseudorandom, Monte Carlo, simulation, logistic, probit, Poisson, NB1, NB2, NB-C, hurdle, offset, ordered, multinomial

1 Introduction

Statisticians use synthetic datasets to evaluate the appropriateness of fit statistics and to determine the effect of modeling after making specific alterations to the data. Models based on synthetically created datasets have proved to be extremely useful in this respect and appear to be used with increasing frequency in texts on statistical modeling.

In this article, I demonstrate how to construct synthetic datasets that are appropriate for various popular discrete-response regression models. The same methods may be used to create data specific to a wide variety of alternative models. In particular, I show how to create synthetic datasets for given types of binomial, Poisson, negative binomial, proportional odds, multinomial, and hurdle models using Stata's pseudorandom-number generators. I demonstrate standard models, models with an offset, and models having user-defined binary, factor, or nonrandom continuous predictors. Typically, synthetic models have predictors with values distributed as pseudorandom uniform or pseudorandom normal. This will be our paradigm case, but synthetic datasets do not have to be established in such a manner—as I demonstrate.

In 1995, Walter Linde-Zwirble and I developed several pseudorandom-number generators using Stata's programming language (Hilbe and Linde-Zwirble 1995, 1998), including the binomial, Poisson, negative binomial, gamma, inverse Gaussian, beta binomial, and others. Based on the rejection method, random numbers that were based on

distributions belonging to the one-parameter exponential family of distributions could rather easily be manipulated to generate full synthetic datasets. A synthetic binomial dataset could be created, for example, having randomly generated predictors with corresponding user-specified parameters and denominators. One could also specify whether the data was to be logit, probit, or any other appropriate binomial link function.

Stata's pseudorandom-number generators are not only based on a different method from those used in the earlier `rnd*` suite of generators but also, in general, use different parameters. The examples in this article all rely on the new Stata functions and are therefore unlike model creation using the older programs. This is particularly the case for the negative binomial.

I divide this article into four sections. First, I discuss creation of synthetic count response models—specifically, Poisson, log-linked negative binomial (NB2), linear negative binomial (NB1), and canonical negative binomial (NB-C) models. Second, I develop code for binomial models, which include both Bernoulli or binary models and binomial or grouped logit and probit models. Because the logic of creating and extending such models was developed in the preceding section on count models, I do not spend much time explaining how these models work. The third section provides a relatively brief overview of creating synthetic proportional slopes models, including the proportional odds model, and code for constructing synthetic categorical response models, e.g., the multinomial logit. Finally, I present code on how to develop synthetic hurdle models, which are examples of two-part models having binary and count components. Statisticians should find it relatively easy to adjust the code that is provided to construct synthetic data and models for other discrete-response regression models.

2 Synthetic count models

I first create a simple Poisson model because Stata's `rpoisson()` function is similar to my original `rndpoi` (used to create a single vector of Poisson-distributed numbers with a specified mean) and `rndpoix` (used to create a Poisson dataset) commands. Uniform random variates work as well as and at times superior to random normal variates for the creation of continuous predictors, which are used to create many of the models below. The mean of the resultant fitted value will be lower using the uniform distribution, but the model results are nevertheless identical.

(*Continued on next page*)

```
* SYNTHETIC POISSON DATA
* [With predictors x1 and x2, having respective parameters of 0.75 and -1.25
* and an intercept of 2]
* poi_rng.do  22Jan2009
clear
set obs 50000
set seed 4744
generate x1 = invnormal(runiform())   // normally distributed: values between
                                      // ~ -4.5 - 4.5
generate x2 = invnormal(runiform())   // normally distributed: values between
                                      // ~ -4.5 - 4.5
generate xb = 2 + 0.75*x1 - 1.25*x2   // linear predictor; define parameters
generate exb = exp(xb)                // inverse link; define Poisson mean
generate py = rpoisson(exb)    // generate random Poisson variate with mean=exb
glm py x1 x2, nolog family(poi)       // model resultant data
```

The model output is given as

```
. glm py x1 x2, nolog family(poi)
Generalized linear models                         No. of obs      =     50000
Optimization     : ML                             Residual df     =     49997
                                                  Scale parameter =         1
Deviance         =  52295.46204                   (1/df) Deviance =  1.045972
Pearson          =  50078.33993                   (1/df) Pearson  =  1.001627
Variance function: V(u) = u                       [Poisson]
Link function    : g(u) = ln(u)                   [Log]
                                                  AIC             =  4.783693
Log likelihood   = -119589.3262                   BIC             =   -488661
------------------------------------------------------------------------------
             |                 OIM
          py |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |   .7488765   .0009798   764.35   0.000     .7469562    .7507967
          x2 |  -1.246898   .0009878 -1262.27   0.000    -1.248834   -1.244962
       _cons |   2.002672   .0017386  1151.91   0.000     1.999265     2.00608
------------------------------------------------------------------------------
```

Notice that the parameter estimates approximate the user-defined values. If we delete the seed line, add code to store each parameter estimate, and convert the do-file to an r-class ado-file, it is possible to perform a Monte Carlo simulation of the synthetic model parameters. The above synthetic Poisson data and model code may be amended to do a simple Monte Carlo simulation as follows:

```
* MONTE CARLO SIMULATION OF SYNTHETIC POISSON DATA
* 9Feb2009
program poi_sim, rclass
     version 11
     drop _all
     set obs 50000
     generate x1 = invnormal(runiform())
     generate x2 = invnormal(runiform())
     generate xb = 2 + 0.75*x1 - 1.25*x2
     generate exb = exp(xb)
     generate py = rpoisson(exb)
     glm py x1 x2, nolog family(poi)
     return scalar sx1 = _b[x1]
     return scalar sx2 = _b[x2]
     return scalar sc  = _b[_cons]
end
```

The model parameter estimates are stored in `sx1`, `sx2`, and `sc`. The following simple `simulate` command is used for a Monte Carlo simulation involving 100 repetitions. Essentially, what we are doing is performing 100 runs of the `poi_rng` do-file, and averaging the values of the three resultant parameter estimates.

```
. simulate mx1=r(sx1) mx2=r(sx2) mcon=r(sc), reps(100): poi_sim
  (output omitted)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
mx1	100	.7499039	.000987	.7473155	.7524396
mx2	100	-1.250145	.0009411	-1.25298	-1.248092
mcon	100	1.9999	.0015481	1.995079	2.003942

Using a greater number of repetitions will result in mean values closer to the user-specified values of 0.75, −1.25, and 2. Standard errors may also be included in the above simulation, as well as values of the Pearson-dispersion statistic, which will have a value of 1.0 when the model is Poisson. The value of the heterogeneity parameter, alpha, may also be simulated for negative binomial models. In fact, any statistic that is stored as a return code may be simulated, as well as any other statistic for which we provide the appropriate storage code.

It should be noted that the Pearson-dispersion statistic displayed in the model output for the generated synthetic Poisson data is 1.001627. This value indicates a Poisson model with no extra dispersion; that is, the model is Poisson. Values of the Pearson dispersion greater than 1.0 indicate possible overdispersion in a Poisson model. See Hilbe (2007) for a discussion of count model overdispersion and Hilbe (2009) for a comprehensive discussion of binomial extradisperson. A good overview of overdispersion may also be found in Hardin and Hilbe (2007).

Most synthetic models use either pseudorandom uniform or normal variates for predictors. However, it is possible to create both random and fixed-level categorical predictors as well. Next I create a three-level predictor and a binary predictor to build the synthetic model. I create the categorical variables by using the `irecode()` function, with specified percentages indicated. `x1` is partitioned into three levels: `x1_1` consists

of the first 50% of the data (or approximately 25,000 observations). `x1_2` has another 30% of the data (approximately 15,000 observations), and `x1_3` has the final 10% of the data (approximately 10,000 observations). `x1_1` is the referent. `x2` is binary with approximately 30,000 zeros and 20,000 ones. The user-defined parameters are `x1_2` = 2, `x1_3` = 3, and `x2` = −2.5. The intercept is specified as 1.

```
* SYNTHETIC POISSON DATA
* poif_rng.do 6Feb2009
* x1_2=2, x1_3=3, x2=-2.5, _cons=1
clear
set obs 50000
set seed 4744
generate x1 = irecode(runiform(), 0, 0.5, 0.8, 1)
generate x2 = irecode(runiform(), 0.6, 1)
tabulate x1, gen(x1_)
generate xb = 1 + 2*x1_2 + 3*x1_3 - 2.5*x2
generate exb = exp(xb)
generate py = rpoisson(exb)
glm py x1_2 x1_3 x2, nolog family(poi)
```

The model output is given as

```
. glm py x1_2 x1_3 x2, nolog family(poi)
Generalized linear models                         No. of obs      =     50000
Optimization     : ML                             Residual df     =     49996
                                                  Scale parameter =         1
Deviance         =  50391.75682                   (1/df) Deviance =  1.007916
Pearson          =  50115.71287                   (1/df) Pearson  =  1.002394

Variance function: V(u) = u                       [Poisson]
Link function    : g(u) = ln(u)                   [Log]

                                                  AIC             =  3.959801
Log likelihood   = -98991.02229                   BIC             = -490553.9
```

py	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1_2	1.995445	.0053683	371.71	0.000	1.984923	2.005966
x1_3	2.996465	.0051336	583.70	0.000	2.986404	3.006527
x2	-2.490218	.0059027	-421.88	0.000	-2.501787	-2.478649
_cons	1.00166	.0048605	206.08	0.000	.9921333	1.011186

We can obtain exact numbers of observations for each level by using the `inrange()` function. Using the same framework as above, we can amend `x1` to have exactly 25,000, 15,000, and 10,000 observations in the factored levels by using the following example code:

```
generate x1 = _n
replace x1 = inrange(_n, 1, 25000)*1 + inrange(_n, 25001, 40000)*2 + //
     inrange(_n, 40001, 50000)*3
tabulate x1, gen(x1_)
```

The tabulation output is given as

```
. tabulate x1, gen(x1_)
```

x1	Freq.	Percent	Cum.
1	25,000	50.00	50.00
2	15,000	30.00	80.00
3	10,000	20.00	100.00
Total	50,000	100.00	

Poisson models are commonly parameterized as rate models. As such, they use an offset, which reflects the area or time over which the count response is generated. Because the natural log is the canonical link of the Poisson model, the offset must be logged prior to entry into the estimating algorithm.

A synthetic offset may be randomly generated or may be specified by the user. For this example, I will create an area offset having increasing values of 100 for each 10,000 observations in the 50,000-observation dataset. The shortcut code used to create this variable is given below. I have commented code that can be used to generate the same offset as in the single-line command that is used in this algorithm. The commented code better shows what is being done and can be used by those who are uncomfortable using the shortcut.

```
* SYNTHETIC RATE POISSON DATA
* poio_rng.do  22Jan2009
clear
set obs 50000
set seed 4744
generate off = 100 + 100*int((_n-1)/10000)     // creation of offset

* generate off = 100 in 1/10000   // These lines duplicate the single line above
* replace off = 200 in 10001/20000
* replace off = 300 in 20001/30000
* replace off = 400 in 30001/40000
* replace off = 500 in 40001/50000

generate loff = ln(off)           // log offset prior to entry into model
generate x1 = invnormal(runiform())
generate x2 = invnormal(runiform())
generate xb = 2 + 0.75*x1 - 1.25*x2 + loff   // offset added to linear predictor
generate exb = exp(xb)
generate py = rpoisson(exb)
glm py x1 x2, nolog family(poi) offset(loff) // added offset option
```

We expect that the resultant model will have approximately the same parameter values as the earlier model but with different standard errors. Modeling the data without using the `offset` option results in similar parameter estimates to those produced when an offset is used, with the exception that the estimated intercept is highly inflated.

(*Continued on next page*)

The results of the rate-parameterized Poisson algorithm above are displayed below:

```
. glm py x1 x2, nolog family(poi) offset(loff)
Generalized linear models                         No. of obs      =     50000
Optimization     : ML                             Residual df     =     49997
                                                  Scale parameter =         1
Deviance         =  49847.73593                   (1/df) Deviance =  .9970145
Pearson          =  49835.24046                   (1/df) Pearson  =  .9967646

Variance function: V(u) = u                       [Poisson]
Link function    : g(u) = ln(u)                   [Log]

                                                  AIC             =  10.39765
Log likelihood   = -259938.1809                   BIC             = -491108.7

------------------------------------------------------------------------------
             |                 OIM
          py |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |   .7500656   .0000562  1.3e+04   0.000     .7499555    .7501758
          x2 |  -1.250067   .0000576 -2.2e+04   0.000     -1.25018   -1.249954
       _cons |   1.999832   .0001009  2.0e+04   0.000     1.999635    2.00003
        loff |   (offset)
------------------------------------------------------------------------------
```

I mentioned earlier that a Poisson model having a Pearson dispersion greater than 1.0 indicates possible overdispersion. The NB2 model is commonly used in such situations to accommodate the extra dispersion.

The NB2 parameterization of the negative binomial can be generated as a Poisson-gamma mixture model, with a gamma scale parameter of 1. We use this method to create synthetic NB2 data. The negative binomial random-number generator in Stata is not parameterized as NB2 but rather derives directly from the NB-C model (see Hilbe [2007]). `rnbinomial()` may be used to create a synthetic NB-C model, but not NB2 or NB1. Below is code that can be used to construct NB2 model data. The same parameters are used here as for the above Poisson models.

```
* SYNTHETIC NEGATIVE BINOMIAL (NB2) DATA
* nb2_rng.do  22Jan2009
clear
set obs 50000
set seed 8444
generate x1 = invnormal(runiform())
generate x2 = invnormal(runiform())
generate xb = 2 + 0.75*x1 - 1.25*x2  // same linear predictor as Poisson above
generate a = .5                      // value of alpha, the NB2 heterogeneity
                                        parameter
generate ia = 1/a                    // inverse alpha
generate exb = exp(xb)               // NB2 mean
generate xg = rgamma(ia, a)          // generate random gamma variate given alpha
generate xbg = exb*xg                // gamma variate parameterized by linear
                                        predictor
generate nby = rpoisson(xbg)         // generate mixture of gamma and Poisson
glm nby x1 x2, family(nb ml) nolog   // model as negative binomial (NB2)
```

The model output is given as

```
. glm nby x1 x2, family(nb ml) nolog
Generalized linear models                         No. of obs      =     50000
Optimization     : ML                             Residual df     =     49997
                                                  Scale parameter =         1
Deviance         =  54131.21274                   (1/df) Deviance =  1.082689
Pearson          =   49994.6481                   (1/df) Pearson  =   .999953
Variance function: V(u) = u+(.5011)u^2            [Neg. Binomial]
Link function    : g(u) = ln(u)                   [Log]
                                                  AIC             =  6.148235
Log likelihood   = -153702.8674                   BIC             = -486825.2
```

nby	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1	.7570565	.0038712	195.56	0.000	.749469	.764644
x2	-1.252193	.0040666	-307.92	0.000	-1.260164	-1.244223
_cons	1.993917	.0039504	504.74	0.000	1.986175	2.00166

```
Note: Negative binomial parameter estimated via ML and treated as fixed once
```

The values of the parameters and of alpha closely approximate the values specified in the algorithm. These values may of course be altered by the user. Note also the values of the dispersion statistics. The Pearson dispersion approximates 1.0, indicating an approximate "perfect" fit. The deviance dispersion is 8% greater, demonstrating that it is not to be used as an assessment of overdispersion. In the same manner in which a Poisson model may be Poisson overdispersed, an NB2 model may be overdispersed as well. It may, in fact, overadjust for Poisson overdispersion. Scaling standard errors or applying a robust variance estimate can be used to adjust standard errors in the case of NB2 overdispersion. See Hilbe (2007) for a discussion of NB2 overdispersion and how it compares with Poisson overdispersion.

If you desire to more critically test the negative binomial dispersion statistic, then you should use a Monte Carlo simulation routine. The NB2 negative binomial heterogeneity parameter, α, is stored in `e(a)` but must be referred to using single quotes, `'e(a)'`. Observe how the remaining statistics we wish to use in the Monte Carlo simulation program are stored.

```
* SIMULATION OF SYNTHETIC NB2 DATA
* x1=.75, x2=-1.25, _cons=2, alpha=0.5
program nb2_sim, rclass
version 11
clear
set obs 50000
generate x1 = invnormal(runiform())         // define predictors
generate x2 = invnormal(runiform())
generate xb = 2 + 0.75*x1 - 1.25*x2         // define parameter values
generate a = .5
generate ia = 1/a
generate exb = exp(xb)
generate xg = rgamma(ia, a)
generate xbg = exb*xg
generate nby = rpoisson(xbg)
```

```
glm nby x1 x2, nolog family(nb ml)
return scalar sx1 = _b[x1]             // x1
return scalar sx2 = _b[x2]             // x2
return scalar sxc = _b[_cons]          // intercept (_cons)
return scalar pd  = e(dispers_p)       // Pearson dispersion
return scalar dd  = e(dispers_s)       // deviance dispersion
return scalar _a = `e(a)´              // alpha
end
```

To obtain the Monte Carlo averaged statistics we desire, use the following options with the `simulate` command:

```
. simulate mx1=r(sx1) mx2=r(sx2) mxc=r(sxc) pdis=r(pd) ddis=r(dd) alpha=r(_a),
> reps(100): nb2_sim
  (output omitted)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
mx1	100	.750169	.0036599	.7407614	.758591
mx2	100	-1.250081	.0037403	-1.258952	-1.240567
mxc	100	2.000052	.0040703	1.987038	2.010417
pdis	100	1.000241	.0050856	.9881558	1.01285
ddis	100	1.084059	.0015233	1.079897	1.087076
alpha	100	.5001092	.0042068	.4873724	.509136

Note the range of parameter and dispersion values. The code for constructing synthetic datasets produces quite good values; i.e., the mean of the parameter estimates is very close to their respective target values, and the standard errors are tight. This is exactly what we want from an algorithm that creates synthetic data.

We may use an offset into the NB2 algorithm in the same manner as we did for the Poisson. Because the mean of the Poisson and NB2 are both `exp(xb)`, we may use the same method. The synthetic NB2 data and model with offset is in the `nb2o_rng.do` file.

The NB1 model is also based on a Poisson-gamma mixture distribution. The NB1 heterogeneity or ancillary parameter is typically referred to as δ, not α as with NB2. Converting the NB2 algorithm to NB1 entails defining `idelta` as the inverse of the value of delta, the desired value of the model ancillary parameter, multiplying the result by the fitted value, `exb`. The terms `idelta` and `1/idelta` are given to the `rgamma()` function. All else is the same as in the NB2 algorithm. The resultant synthetic data are modeled using Stata's `nbreg` command with the `disp(constant)` option.

```
* SYNTHETIC LINEAR NEGATIVE BINOMIAL (NB1) DATA
* nb1_rng.do 3Apr2006
* Synthetic NB1 data and model
* x1= 1.1;  x2= -.8;  x3= .2;  _c= .7
* delta = .3  (1/.3 = 3.3333333)
quietly {
     clear
     set obs 50000
     set seed 13579
     generate x1 = invnormal(runiform())
     generate x2 = invnormal(runiform())
     generate x3 = invnormal(runiform())
     generate xb = .7 + 1.1*x1 - .8*x2 + .2*x3
     generate exb = exp(xb)
     generate idelta = 3.3333333*exb
     generate xg = rgamma(idelta, 1/idelta)
     generate xbg = exb*xg
     generate nb1y = rpoisson(xbg)
}
nbreg nb1y x1 x2 x3, nolog disp(constant)
```

The model output is given as

```
. nbreg nb1y x1 x2 x3, nolog disp(constant)
Negative binomial regression                    Number of obs   =      49910
                                                LR chi2(3)      =   82361.44
Dispersion     = constant                       Prob > chi2     =     0.0000
Log likelihood = -89323.313                     Pseudo R2       =     0.3156

------------------------------------------------------------------------------
        nb1y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |    1.098772   .0022539   487.49   0.000     1.094354    1.103189
          x2 |   -.8001773   .0022635  -353.51   0.000    -.8046137   -.7957409
          x3 |    .1993391   .0022535    88.46   0.000     .1949223    .2037559
       _cons |    .7049061   .0038147   184.79   0.000     .6974294    .7123827
-------------+----------------------------------------------------------------
    /lndelta |   -1.193799    .029905                     -1.252411   -1.135186
-------------+----------------------------------------------------------------
       delta |    .3030678   .0090632                      .2858147    .3213623
------------------------------------------------------------------------------
Likelihood-ratio test of delta=0:  chibar2(01) = 1763.21 Prob>=chibar2 = 0.000
```

The parameter values and value of delta closely approximate the specified values.

The NB-C, however, must be constructed in an entirely different manner from NB2, NB1, or Poisson. NB-C is not a Poisson-gamma mixture and is based on the negative binomial probability distribution function. Stata's `rnbinomial(`a`,`b`)` function can be used to construct NB-C data. Other options, such as offsets, nonrandom variance adjusters, and so forth, are easily adaptable for the `nbc_rng.do` file.

```
* SYNTHETIC CANONICAL NEGATIVE BINOMIAL (NB-C) DATA
* nbc_rng.do 30dec2005
clear
set obs 50000
set seed 7787
generate x1 = runiform()
generate x2 = runiform()
```

```
generate xb = 1.25*x1 + .1*x2 - 1.5
generate a =  1.15
generate mu = 1/((exp(-xb)-1)*a)         // inverse link function
generate p = 1/(1+a*mu)                  // probability
generate r = 1/a
generate y = rnbinomial(r, p)
cnbreg y x1 x2, nolog
```

I wrote a maximum likelihood NB-C command, `cnbreg`, in 2005, which was posted to the Statistical Software Components (SSC) site, and I posted an amendment in late February 2009. The statistical results are the same in the original and the amended version, but the amendment is more efficient and pedagogically easier to understand. Rather than simply inserting the NB-C inverse link function in terms of xb for each instance of μ in the log-likelihood function, I have reduced the formula for the NB-C log likelihood to

$$\text{LL}_{\text{NB-C}} = \sum [y(xb) + (1/\alpha)\ln\{1 - \exp(xb)\} + \ln\Gamma(y + 1/\alpha) - \ln\Gamma(y + 1) - \ln\Gamma(1/\alpha)]$$

Also posted to the site is a heterogeneous NB-C regression command that allows parameterization of the heterogeneity parameter, α. Stata calls the NB2 version of this a generalized negative binomial. However, as I discuss in Hilbe (2007), there are previously implemented generalized negative binomial models with entirely different parameterizations. Some are discussed in that source. Moreover, LIMDEP has offered a heterogeneous negative binomial for many years that is the same model as is the generalized negative binomial in Stata. For these reasons, I prefer labeling Stata's `gnbreg` command a heterogeneous model. A `hcnbreg` command was also posted to SSC in 2005.

The synthetic NB-C model of the above created data is displayed below. I have specified values of `x1` and `x2` as 1.25 and 0.1, respectively, and an intercept value of −1.5. alpha is given as 1.15. The model closely reflects the user-specified parameters.

```
. cnbreg y x1 x2, nolog
initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -85868.162
rescale:       log likelihood = -78725.374
rescale eq:    log likelihood = -71860.156
Canonical Negative Binomial Regression            Number of obs   =      50000
                                                  Wald chi2(2)    =    6386.70
Log likelihood = -62715.384                       Prob > chi2     =     0.0000
```

y	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1	1.252675	.015776	79.40	0.000	1.221754	1.283595
x2	.1009038	.0091313	11.05	0.000	.0830068	.1188008
_cons	-1.504659	.0177159	-84.93	0.000	-1.539382	-1.469937
/lnalpha	.133643	.0153947	8.68	0.000	.1034699	.1638161
alpha	1.142985	.0175959			1.109012	1.177998

```
AIC Statistic   =       2.509
```

3 Synthetic binomial models

Synthetic binomial models are constructed in the same manner as synthetic Poisson data and models. The key lines are those that generate pseudorandom variates, a line creating the linear predictor with user-defined parameters, a line using the inverse link function to generate the mean, and a line using the mean to generate random variates appropriate to the distribution.

A Bernoulli distribution consists entirely of binary values, 0/1. y is binary and is considered here to be the response variable that is explained by the values of $x1$ and $x2$. Data such as this is typically modeled using a logistic regression. A probit or complementary log-log model can also be used to model the data.

	y	$x1$	$x2$
1:	1	1	1
2:	0	1	1
3:	1	0	1
4:	1	1	0
5:	1	0	1
6:	0	0	1

The above data may be grouped by covariate patterns. The covariates here are, of course, $x1$ and $x2$. With y now the number of successes, i.e., a count of 1s, and m the number of observations having the same covariate pattern, the above data may be grouped as

	y	m	$x1$	$x2$
1:	1	2	1	1
2:	2	3	0	1
3:	1	1	1	0

The distribution of y/m is binomial. y is a count of observations having a value of $y = 1$ for a specific covariate pattern, and m is the number of observations having the same covariate pattern. One can see that the Bernoulli distribution is a subset of the binomial, i.e., a binomial distribution where $m = 1$. In actuality, a logistic regression models the top data as if there were no m, regardless of the number of separate covariate patterns. Grouped logistic, or binomial-logit, regression assumes appropriate values of y and m. In Stata, grouped data such as the above can be modeled as a logistic regression using the `blogit` or `glm` command. I recommend using the `glm` command because `glm` is accompanied with a wide variety of test statistics and is based directly on the binomial probability distribution. Moreover, alternative linked binomial models may easily be applied.

Algorithms for constructing synthetic Bernoulli models differ little from creating synthetic binomial models. The only difference is that for the binomial, m needs to be accommodated. I shall demonstrate the difference—and similarity—of the Bernoulli and binomial models by generating data using the same parameters. First, the Bernoulli-logit model, or logistic regression:

```
* SYNTHETIC BERNOULLI-LOGIT DATA
* berl_rng.do 5Feb2009
* x1=.75, x2=-1.25, _cons=2
clear
set obs 50000
set seed 13579
generate x1 = invnormal(runiform())
generate x2 = invnormal(runiform())
generate xb = 2 + 0.75*x1 - 1.25*x2
generate exb = 1/(1+exp(-xb))                  // inverse logit link
generate by = rbinomial(1, exb)                // specify m=1 in function
logit by x1 x2, nolog
```

The output is displayed as

```
. logit by x1 x2, nolog
Logistic regression                               Number of obs   =      50000
                                                  LR chi2(2)      =   10861.44
                                                  Prob > chi2     =     0.0000
Log likelihood =   -18533.1                       Pseudo R2       =     0.2266
```

by	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1	.7555715	.0143315	52.72	0.000	.7274822	.7836608
x2	-1.256906	.016125	-77.95	0.000	-1.28851	-1.225301
_cons	2.018775	.0168125	120.08	0.000	1.985823	2.051727

Second, the code for constructing a synthetic binomial, or grouped, model:

```
* SYNTHETIC BINOMIAL-LOGIT DATA
*  binl_rng.do 5feb2009
* x1=.75, x2=-1.25, _cons=2
clear
set obs 50000
set seed 13579
generate x1 = invnormal(runiform())
generate x2 = invnormal(runiform())
* ================================================
* Select either User Specified or Random denominator.
* generate d = 100 + 100*int((_n-1)/10000)  // specified denominator values
generate d = ceil(10*runiform())            // integers 1-10, mean of ~5.5
* ================================================
generate xb = 2 + 0.75*x1 - 1.25*x2
generate exb = 1/(1+exp(-xb))
generate by = rbinomial(d, exb)
glm by x1 x2, nolog family(bin d)
```

The final line calculates and displays the output below:

```
. glm by x1 x2, nolog family(bin d)
Generalized linear models                         No. of obs      =     50000
Optimization     : ML                             Residual df     =     49997
                                                  Scale parameter =         1
Deviance         =  47203.16385                   (1/df) Deviance =  .9441199
Pearson          =   50135.2416                   (1/df) Pearson  =  1.002765
Variance function: V(u) = u*(1-u/d)               [Binomial]
Link function    : g(u) = ln(u/(d-u))             [Logit]
                                                  AIC             =  1.854676
Log likelihood   = -46363.90908                   BIC             = -493753.3
------------------------------------------------------------------------------
             |                 OIM
          by |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |   .7519113   .0060948   123.37   0.000     .7399657    .7638569
          x2 |  -1.246277   .0068415  -182.16   0.000    -1.259686   -1.232868
       _cons |    2.00618   .0071318   281.30   0.000     1.992202    2.020158
------------------------------------------------------------------------------
```

The only difference between the two is the code between the lines and the use of `d` rather than `1` in the `rbinomial()` function. Displayed is code for generating a random denominator and code for specifying the same values as were earlier used for the Poisson and negative binomial offsets.

See Cameron and Trivedi (2009) for a nice discussion of generating binomial data; their focus, however, differs from the one taken here. I nevertheless recommend reading chapter 4 of their book, written after the do-files that are presented here were developed.

Note the similarity of parameter values. Use of Monte Carlo simulation shows that both produce identical results. I should mention that the dispersion statistic is only appropriate for binomial models, not for Bernoulli. The binomial-logit model above has a dispersion of 1.002765, which is as we would expect. This relationship is discussed in detail in Hilbe (2009).

It is easy to amend the above code to construct synthetic probit or complementary log-log data. I show the probit because it is frequently used in econometrics.

```
* SYNTHETIC BINOMIAL-PROBIT DATA
* binp_rng.do 5feb2009
* x1=.75, x2=-1.25, _cons=2
clear
set obs 50000
set seed 4744
generate x1 = runiform()                  // use runiform() with probit data
generate x2 = runiform()
* ================================================
* Select User Specified or Random Denominator. Select Only One
* generate d = 100+100*int((_n-1)/10000)  // specified denominator values
generate d = ceil(10*runiform())          // pseudorandom-denominator values
* ================================================
generate xb = 2 + 0.75*x1 - 1.25*x2
generate double exb = normal(xb)
generate double by = rbinomial(d, exb)
glm by x1 x2, nolog family(bin d) link(probit)
```

The model output is given as

```
. glm by x1 x2, nolog family(bin d) link(probit)
Generalized linear models                         No. of obs      =     50000
Optimization     : ML                             Residual df     =     49997
                                                  Scale parameter =         1
Deviance         =  35161.17862                   (1/df) Deviance =  .7032658
Pearson          =  50277.67366                   (1/df) Pearson  =  1.005614

Variance function: V(u) = u*(1-u/d)               [Binomial]
Link function    : g(u) = invnorm(u/d)            [Probit]

                                                  AIC             =  1.132792
Log likelihood   = -28316.80908                   BIC             = -505795.3

------------------------------------------------------------------------------
             |                 OIM
          by |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |   .7467577   .0148369    50.33   0.000      .717678    .7758374
          x2 |  -1.247248   .0157429   -79.23   0.000    -1.278103   -1.216392
       _cons |   2.003984   .0122115   164.11   0.000      1.98005    2.027918
------------------------------------------------------------------------------
```

The `normal()` function is the inverse probit link and replaces the inverse logit link.

4 Synthetic categorical response models

I have previously discussed the creation of synthetic ordered logit, or proportional odds, data in Hilbe (2009), and I refer you to that source for a more thorough examination of the subject. I also examine multinomial logit data in the same source. Because of the complexity of the model, the generated data are a bit more variable than with synthetic logit, Poisson, or negative binomial models. However, Monte Carlo simulation (not shown) proves that the mean values closely approximate the user-supplied parameters and cut points.

I display code for generating synthetic ordered probit data below.

```
* SYNTHETIC ORDERED PROBIT DATA AND MODEL
* oprobit_rng.do 19Feb 2008
display in ye "b1 = .75; b2 = 1.25"
display in ye "Cut1=2; Cut2=3,; Cut3=4"
quietly {
     drop _all
     set obs 50000
     set seed 12345
     generate double x1 = 3*runiform() + 1
     generate double x2 = 2*runiform() - 1
     generate double y = .75*x1 + 1.25*x2 + invnormal(runiform())
     generate int ys = 1 if y<=2
     replace ys=2 if y<=3 & y>2
     replace ys=3 if y<=4 & y>3
     replace ys=4 if  y>4
}
oprobit ys x1 x2, nolog
* predict double (oppr1 oppr2 oppr3 oppr4), pr
```

The modeled data appears as

```
. oprobit ys x1 x2, nolog
Ordered probit regression                         Number of obs   =      50000
                                                  LR chi2(2)      =   24276.71
                                                  Prob > chi2     =     0.0000
Log likelihood = -44938.779                       Pseudo R2       =     0.2127

------------------------------------------------------------------------------
          ys |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |   .7461112    .006961   107.18   0.000     .7324679    .7597544
          x2 |   1.254821   .0107035   117.23   0.000     1.233842    1.275799
-------------+----------------------------------------------------------------
       /cut1 |   1.994369   .0191205                      1.956894    2.031845
       /cut2 |   2.998502   .0210979                      2.957151    3.039853
       /cut3 |   3.996582   .0239883                      3.949566    4.043599
------------------------------------------------------------------------------
```

The user-specified slopes are 0.75 and 1.25, which are closely approximated above. Likewise, the specified cuts of 2, 3, and 4 are nearly identical to the synthetic values, which are the same to the hundredths place.

The proportional-slopes code is created by adjusting the linear predictor. Unlike the ordered probit, we need to generate pseudorandom-uniform variates, called `err`, which are then used in the logistic link function, as attached to the end of the linear predictor. The rest of the code is the same for both algorithms. The lines required to create synthetic proportional odds data are the following:

```
generate err = runiform()
generate y = .75*x1 + 1.25*x2 + log(err/(1-err))
```

Finally, synthetic ordered slope models may easily be expanded to having more predictors as well as additional levels by using the same logic as shown in the above algorithm. Given three predictors with values assigned as `x1` = 0.5, `x2` = 1.76, and `x3` = 1.25, and given five levels with cuts at 0.8, 1.6, 2.4, and 3.2, the amended part of the code is as follows:

```
generate double x3 = runiform()
generate y = .5*x1 + 1.75*x2 - 1.25*x3 + invnormal(uniform())
generate int ys = 1 if y<=.8
replace ys=2 if y<=1.6 & y>.8
replace ys=3 if y<=2.4 & y>1.6
replace ys=4 if y<=3.2 & y>2.4
replace ys=5 if  y>3.2
oprobit ys x1 x2 x3, nolog
```

(*Continued on next page*)

Synthetic multinomial logit data may be constructed using the following code:

```
* SYNTHETIC MULTINOMIAL LOGIT DATA AND MODEL
* mlogit_rng.do 15Feb2008
* y=2: x1= 0.4, x2=-0.5, _cons=1.0
* y=3: x1=-3.0, x2=0.25, _cons=2.0
quietly {
     clear
     set memory 50m
     set seed 111322
     set obs 100000
     generate x1 = runiform()
     generate x2 = runiform()
     generate denom = 1+exp(.4*x1 - .5*x2 + 1) + exp(-.3*x1 + .25*x2 + 2)
     generate p1 = 1/denom
     generate p2 = exp(.4*x1 - .5*x2 + 1)/denom
     generate p3 = exp(-.3*x1 + .25*x2 + 2)/denom
     generate u = runiform()
     generate y = 1 if u <= p1
     generate p12 = p1 + p2
     replace y=2 if y==. & u<=p12
     replace y=3 if y==.
}
mlogit y x1 x2, baseoutcome(1) nolog
```

I have amended the `uniform()` function in the original code to `runiform()`, which is Stata's newest version of the pseudorandom-uniform generator. Given the nature of the multinomial probability function, the above code is rather self-explanatory. The code may easily be expanded to have more than three levels. New coefficients need to be defined and the probability levels expanded. See Hilbe (2009) for advice on expanding the code. The output of the above `mlogit_rng.do` is displayed as

```
. mlogit y x1 x2, baseoutcome(1) nolog
Multinomial logistic regression                 Number of obs   =     100000
                                                LR chi2(4)      =    1652.17
                                                Prob > chi2     =     0.0000
Log likelihood = -82511.593                     Pseudo R2       =     0.0099

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
1            |  (base outcome)
-------------+----------------------------------------------------------------
2            |
          x1 |   .4245588   .0427772     9.92   0.000     .3407171    .5084005
          x2 |  -.5387675   .0426714   -12.63   0.000    -.6224019    -.455133
       _cons |   1.002834   .0325909    30.77   0.000     .9389566    1.066711
-------------+----------------------------------------------------------------
3            |
          x1 |  -.2953721    .038767    -7.62   0.000    -.371354   -.2193902
          x2 |   .2470191   .0386521     6.39   0.000     .1712625    .3227757
       _cons |   2.003673   .0295736    67.75   0.000      1.94571    2.061637
------------------------------------------------------------------------------
```

By amending the `mlogit_rng.do` code to an r-class ado-file, with the following lines added to the end, the following Monte Carlo simulation may be run, verifying the parameters displayed from the do-file:

```
return scalar x1_2 = [2]_b[x1]
return scalar x2_2 = [2]_b[x2]
return scalar _c_2 = [2]_b[_cons]
return scalar x1_3 = [3]_b[x1]
return scalar x2_3 = [3]_b[x2]
return scalar _c_3 = [3]_b[_cons]
end
```

The ado-file is named `mlogit_sim`.

```
. simulate mx12=r(x1_2) mx22=r(x2_2) mc2=r(_c_2) mx13=r(x1_3) mx23=r(x2_3)
> mc3=r(_c_3), reps(100): mlogit_sim
  (output omitted)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
mx12	100	.4012335	.0389845	.2992371	.4943814
mx22	100	-.4972758	.0449005	-.6211451	-.4045792
mc2	100	.9965573	.0300015	.917221	1.0979
mx13	100	-.2989224	.0383149	-.3889697	-.2115128
mx23	100	.2503969	.0397617	.1393684	.3484274
mc3	100	1.998332	.0277434	1.924436	2.087736

The user-specified values are reproduced by the synthetic multinomial program.

5 Synthetic hurdle models

Finally, I show an example of how to expand the above synthetic data generators to construct synthetic negative binomial-logit hurdle data. The code may be easily amended to construct Poisson-logit, Poisson-probit, Poisson-cloglog, NB2—probit, and NB2-cloglog models. In 2005, I published several hurdle models, which are currently on the SSC web site. This example is shown to demonstrate how similar synthetic models may be created for zero-truncated and zero-inflated models, as well as a variety of different types of panel models. Synthetic models and correlation structures are found in Hardin and Hilbe (2003) for generalized estimating equations models.

Hurdle models are discussed in Long and Freese (2006), Hilbe (2007), Winkelmann (2008), and Cameron and Trivedi (2009). The traditional method of parameterizing hurdle models is to have both binary and count components be of equal length, which makes theoretical sense. However, they may be of unequal lengths, as are zero-inflated models. Moreover, hurdle models can be used to estimate both over- and underdispersed count data, unlike zero-inflated models.

The binary component of a hurdle model is typically a logit, probit, or cloglog binary response model. However, the binary component may take the form of a right-censored Poisson model or a censored negative binomial model. In fact, the earliest applications

of hurdle models consisted of Poisson–Poisson and Poisson-geometric models. However, it was discovered that the censored geometric component has an identical log likelihood to that of the logit, which has been preferred in most recent applications. I published censored Poisson and negative binomial models to the SSC web site in 2005, and truncated and econometric censored Poisson models in 2009. They may be used for constructing this type of hurdle model.

The synthetic hurdle model below is perhaps the most commonly used version—a NB2-logit hurdle model. It is a combination of a 0/1 binary logit model and a zero-truncated NB2 model. For the logit portion, all counts greater than 0 are given the value of 1. There is no estimation overlap in response values, as is the case for zero-inflated models.

The parameters specified in the example synthetic hurdle model below are

```
* SYNTHETIC NB2-LOGIT HURDLE DATA
* nb2logit_hurdle.do  J Hilbe  26Sep2005; Mod 4Feb2009.
* LOGIT: x1=-.9, x2=-.1, _c=-.2
* NB2  : x1=.75, n2=-1.25, _c=2, alpha=.5
clear
set obs 50000
set seed 1000
generate x1 = invnormal(runiform())
generate x2 = invnormal(runiform())
* NEGATIVE BINOMIAL- NB2
generate xb = 2 + 0.75*x1 - 1.25*x2
generate a = .5
generate ia = 1/a
generate exb = exp(xb)
generate xg = rgamma(ia, a)
generate xbg = exb*xg
generate nby = rpoisson(xbg)
* BERNOULLI
drop if nby==0
generate pi = 1/(1+exp(-(.9*x1 + .1*x2 + .2)))
generate bernoulli = runiform()>pi
replace nby=0 if bernoulli==0
rename nby y
* logit bernoulli x1 x2, nolog /// test
* ztnb y x1 x2 if y>0, nolog   /// test
* NB2-LOGIT HURDLE
hnblogit y x1 x2, nolog
```

Output for the above synthetic NB2-logit hurdle model is displayed as

```
. hnblogit y x1 x2, nolog
Negative Binomial-Logit Hurdle Regression         Number of obs   =      43443
                                                  Wald chi2(2)    =    5374.14
Log likelihood = -84654.938                       Prob > chi2     =     0.0000
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
logit						
x1	-.8987393	.0124338	-72.28	0.000	-.9231091	-.8743695
x2	-.0904395	.011286	-8.01	0.000	-.1125597	-.0683194
_cons	-.2096805	.0106156	-19.75	0.000	-.2304867	-.1888742
negbinomial						
x1	.743936	.0069378	107.23	0.000	.7303381	.7575339
x2	-1.252363	.0071147	-176.02	0.000	-1.266307	-1.238418
_cons	2.003677	.0070987	282.26	0.000	1.989764	2.01759
/lnalpha	-.6758358	.0155149	-43.56	0.000	-.7062443	-.6454272

```
AIC Statistic =     3.897
```

The results approximate the specified values. A Monte Carlo simulation was preformed, demonstrating that the algorithm does what it is aimed to do.

6 Summary remarks

Synthetic data can be used with substantial efficacy for the evaluation of statistical models. In this article, I have presented algorithmic code that can be used to create several different types of synthetic models. The code may be extended to use for the generation of yet other synthetic models.

I am a strong advocate of using these types of models to better understand the models we apply to real data. I have used these models, or ones based on earlier random-number generators, in Hardin and Hilbe (2007) and in both of my single authored texts (Hilbe 2007, 2009) for assessing model assumptions. With computers gaining in memory and speed, it will soon be possible to construct far more complex synthetic data than we have here. I hope that the rather elementary examples discussed in this article will encourage further use and construction of artificial data.

7 References

Cameron, A. C., and P. K. Trivedi. 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.

Hardin, J. W., and J. M. Hilbe. 2003. *Generalized Estimating Equations*. Boca Raton, FL: Chapman & Hall/CRC.

———. 2007. *Generalized Linear Models and Extensions*. 2nd ed. College Station, TX: Stata Press.

Hilbe, J. M. 2007. *Negative Binomial Regression*. New York: Cambridge University Press.

———. 2009. *Logistic Regression Models*. Boca Raton, FL: Chapman & Hall/CRC.

Hilbe, J. M., and W. Linde-Zwirble. 1995. sg44: Random number generators. *Stata Technical Bulletin* 28: 20–21. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 118–121. College Station, TX: Stata Press.

———. 1998. sg44.1: Correction to random number generators. *Stata Technical Bulletin* 41: 23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, p. 166. College Station, TX: Stata Press.

Long, J. S., and J. Freese. 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.

Winkelmann, R. 2008. *Econometric Analysis of Count Data*. 5th ed. Berlin: Springer.

About the author

Joseph M. Hilbe is an emeritus professor (University of Hawaii), an adjunct professor of statistics at Arizona State University, and Solar System Ambassador with the NASA/Jet Propulsion Laboratory, CalTech. He has authored several texts on statistical modeling, two of which are *Logistic Regression Models* (Chapman & Hall/CRC) and *Negative Binomial Regression* (Cambridge University Press). Hilbe is also chair of the ISI Astrostatistics Committee and Network and was the first editor of the *Stata Technical Bulletin* (1991–1993).

The Stata Journal (2010)
10, Number 1, pp. 125–142

Mata Matters: Stata in Mata

William Gould
StataCorp
College Station, TX
wgould@stata.com

Abstract. Mata is Stata's matrix language. In the Mata Matters column, we show how Mata can be used interactively to solve problems and as a programming language to add new features to Stata. The subject of this column is using Mata to solve data analysis problems with the new Stata commands `putmata` and `getmata`, which were added to official Stata 11 in the update of 11 February 2010.

Keywords: pr0050, Mata, getmata, putmata

1 Introduction

Some problems are more easily solved in Mata than they are in Stata. The problem is that putting data from Stata to Mata and getting the result back again is difficult for casual users and tedious even for experienced users. The new Stata commands `putmata` and `getmata` solve that problem. These commands were added to official Stata 11 in the update of 11 February 2010.

With `putmata`, we can type

```
. putmata *
(12 vectors posted)
```

and thus create a column vector in Mata for each variable in our data. The vectors will have the same names as the variables. If we typed `putmata *` with the automobile data in memory, we would then have vectors named `make`, `price`, `mpg`, `rep78`, `headroom`, `trunk`, `weight`, `length`, `turn`, `displacement`, `gear_ratio`, and `foreign` available for use in Mata.

Note that you type `putmata` at the Stata dot prompt, not at the Mata colon prompt. Rather than typing `putmata *`, let's type

```
. putmata y=mpg X=(weight foreign 1)
(1 vector, 1 matrix posted)
```

Typing that creates a vector in Mata called `y` (which is just `mpg`, renamed) and a matrix called `X` (which contains the columns corresponding to `weight`, `foreign`, and a vector of 1s). We could then enter Mata and type

```
. mata
: b = invsym(X´X)*X´y
: yhat = X*b
: end
. -
```


Vector `yhat` now contains the predicted values from a regression of `y` on `X`. To post the Mata vector back into our Stata dataset, we could type

```
. getmata yhat
```

We would now have the new variable `yhat` in our Stata dataset.

The demonstration is intended to be motivational; I am not seriously suggesting you type the above instead of

```
. regress mpg weight foreign
  (output omitted)
. predict yhat
```

Nonetheless, the motivational example is the outline for what follows. We are going to discuss the details of `putmata` and `getmata`, we are going to use `putmata` as a jumping-off point to discuss writing Mata code to solve both statistical and data-management problems, and we are going to discuss how to package solutions in do-files.

Before we start, verify that you have the new commands `putmata` and `getmata`. In Stata, type

```
. help putmata
```

If you are told that help for `putmata` is not found, you need to update your Stata. You do that by typing

```
. update all
```

2 The putmata command

2.1 Syntax

The syntax of `putmata` is

`putmata` *putlist* [*if*] [*in*] [`, replace` `omit`missing `view`]

putlist can be any combination of the following:

varname or *varlist*
vecname`=`*varname*
matname`=(`*varlist*`)`
matname`=(`*varlist* # ...`)`

For example,

1. You can type `putmata mpg` to create in Mata the vector `mpg`.
2. You can type `putmata mpg weight` to create in Mata the vectors `mpg` and `weight`.

3. You can type `putmata *` to create in Mata vectors for every variable in the Stata dataset.

4. You can type `putmata y=mpg` to create Mata vector `y` containing the contents of Stata variable `mpg`.

5. You can type `putmata X=(weight foreign)` to create Mata matrix `X` containing `weight` in its first column and `foreign` in its second.

6. You can type `putmata X=(weight foreign 1)` to create Mata matrix `X` containing `weight` in its first column, `foreign` in its second, and constant `1` in its third.

You can even type `putmata y=mpg X=(weight foreign 1)` to perform the actions of examples 5 and 6 in a single line. If you specify the `omitmissing` option, however, it does matter whether you type separate or single commands, and you do *not* want to type separate commands:

```
. putmata y=mpg, omitmissing
. putmata X=(weight foreign 1), omitmissing
```

With the above commands, vector `y` will omit observations in which `mpg` contains missing. Matrix `X` will omit observations in which `weight` or `foreign` contain missing. What you want, however, is to omit observations from both `y` and `X` in which any of the variables contain missing. You want to type

```
. putmata y=mpg X=(weight foreign 1), omitmissing
```

2.2 Options

`replace` indicates that it is okay to replace an existing Mata vector or matrix. If you do not specify `replace` and the Mata vector or matrix already exists, `putmata` issues an error.

`omitmissing` specifies to omit observations that contain missing values in any of the variables in *putlist* from the rows of the vectors and matrices created in Mata. In the motivational example in section 1, we coded `b = invsym(X'X)*X'y`, and we created `y` and `X` by typing `putmata y=mpg X=(weight foreign 1)`. We just assumed there were no missing values. Had there been missing values, we would have wanted to create `y` and `X` by typing

```
. putmata y=mpg X=(weight foreign 1), omitmissing
```

`view` specifies that the vector and matrices be created as views onto the Stata data rather than as copies of the contents of the data. Views can save considerable amounts of memory and they have other advantages as well, although sometimes those advantages can turn into disadvantages. All of which is to say, views should be used with caution. We will discuss views later.

3 Using putmata to produce mathematical and statistical results

`putmata` is all you need to solve some problems. For instance, consider solving the set of linear equations $\mathbf{y} = \mathbf{Xb}$ for $\mathbf{b}$. The solution can be obtained by premultiplying both sides by $\mathbf{X}^{-1}$, which results in $\mathbf{X}^{-1}\mathbf{y} = \mathbf{X}^{-1}\mathbf{Xb}$ or $\mathbf{b} = \mathbf{X}^{-1}\mathbf{y}$. If you were teaching a course on linear algebra, you could demonstrate this solution. You might start with `y` and `x` values entered into a Stata dataset:

```
. list
```

	y	x1	x2	x3
1.	27	2	-5	6
2.	-20	3	7	-9
3.	-9	-8	2	1

You might type the following:

```
. putmata y X=(x*)
(1 vector, 1 matrix posted)
. mata
------------------------------------------------- mata (type end to exit) -----
: y
        1
    +-----+
  1 |  27 |
  2 | -20 |
  3 |  -9 |
    +-----+

: X
        1    2    3
    +---------------+
  1 |  2   -5    6  |
  2 |  3    7   -9  |
  3 | -8    2    1  |
    +---------------+

: b = luinv(X)*y
: b
        1
    +-----+
  1 |  4  |
  2 |  7  |
  3 |  9  |
    +-----+

: X*b
        1
    +-----+
  1 |  27 |
  2 | -20 |
  3 |  -9 |
    +-----+

: end
--------------------------------------------------------------------------------
```

You can read the online help or the manual about the Mata function `luinv()`. I chose it because I needed a matrix inverter that could handle nonsymmetric matrices.

More interestingly, let's consider the overdetermined linear set of equations $\mathbf{y} = \mathbf{Xb}$ when $\mathbf{X}$ is $n \times k$, $n > k$. We have more equations than the unknown coefficients. Linear regression $\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ provides one solution. It turns out that $\mathbf{b} = \mathbf{X}^{-1}\mathbf{y}$ provides the same solution if you define $\mathbf{X}^{-1}$ to be the Moore–Penrose generalized inverse for nonsquare matrices! In the Moore–Penrose inverse, $\mathbf{X}^{-1}\mathbf{X}$ equals the identity matrix, but $\mathbf{X}\mathbf{X}^{-1}$ does not. In any case, we can demonstrate the equivalence:

```
. sysuse auto
(1978 Automobile Data)
. putmata y=mpg X=(weight foreign 1)
(1 vector, 1 matrix posted)
. mata
------------------------------------------------- mata (type end to exit) -----
: pinv(X)*y
                  1
  +-----------------+
1 |  -.0065878864   |
2 |  -1.650029106   |
3 |   41.67970233   |
  +-----------------+
: end
--------------------------------------------------------------------------------
```

You could compare the above result with the coefficients reported by typing

```
. regress mpg weight foreign
```

or you could compare it with the Mata calculation of `invsym(X'X)*X'y`.

Mata is a great way to teach. Just as importantly, if you have a matrix calculation you need to make based on your data, you can use `putmata` to post the appropriate vector and matrices from your data and then use Mata to calculate the result.

If you are going to use Mata to make real statistical calculations, I recommend you normalize your data so that the variables are roughly scaled similarly because, when you write matrix formulas, you are not going to concern yourself with using variants that are more numerically accurate. For instance, Stata does not calculate linear regression using $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$, although the calculation it makes is algebraically equivalent, which is to say, would yield the same results on an infinite-precision computer. The calculation Stata makes is more precise on finite-precision computers. Stata removes the means (and later solves for the intercept separately), and it uses a solver to obtain the coefficients, and more. The details are long and involved and the point is this: you are not going to invest that kind of effort. You are going to code $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ or whatever is the equivalent for your problem. There is nothing numerically wrong with using such formulas as long as you do not tax them by having variables that differ too wildly in scale.

The automobile data is an example of a dataset that is sufficiently scaled. In the automobile data, `mpg` varies between 12 and 41 (mean 21.3), `weight` varies between 1,760 and 4,840 (mean 3,019.5), and `foreign` is either 0 or 1 (mean 0.2973). Scaling that varies by a few orders of magnitude is usually of no concern. Let me show you, however, that results would be more accurate if we divided `weight` by 1,000 and `mpg` by 10.

It is a theoretical property of linear regression—and easy to prove—that the sum of the residuals will be zero when the coefficient vector **b** is set to the least-squares result. When we calculate the sum of those residuals using **b** obtained from any finite-precision calculation, however, the sum will not be precisely zero. Using the example above, if we use the **b** obtained by Stata's `regress` command, the sum is –5.1e–15 (meaning -5.1×10^{-15}). If we use **b** = `pinv(X)*y`, the sum is −2.3e–13. If we use **b** = `invsym(X'X)*X'y`, the sum is 7.1e–13. Actually, I have made an adjustment to all those numbers, which I will explain, but these are the right numbers for comparison. If we rescaled the data by dividing `weight` by 1,000 and `mpg` by 10, the errors would be 3.22e–14 for `pinv(X)*y` and −6.48e–13 for `invsym(X'X)*X'y`, and unchanged for `regress`. The two matrix calculations are more accurate when made on better scaled data—the errors were closer to zero—and the results from `regress` remain unchanged. `regress` is robust to scaling.

In any case, all the errors are small. The maximum average error per observation was a mere 7.1e–13/74 = 9.6e–15 miles per gallon. Nonetheless, errors were smaller when we used scaled data.

I mentioned that I adjusted the errors reported above. I did that because when one calculates error on a finite-precision computer, one obtains the desired error plus the error in making the error calculation itself! Were you to calculate these sums of the residuals in the obvious way, which you could do using Mata and by typing `sum(y-X*b)`, you would obtain results different from what I reported. You would obtain –2.5e–13 for `b` obtained from `regress`, –2.5e–12 for `b = pinv(X)*y`, and –6.7e–12 for `b = invsym(X'X)*X'y`. Those error calculations include not just the error caused by numerical error in the calculation of **b** but also the numerical error in the calculation of `sum(y-X*b)`. Such unadjusted results are usually adequate for ranking techniques, and in some sense they are actually better because they also include the error in how you would be likely to use the calculated results. The results are adequate for ranking because, whatever is the error in `sum(y-X*b)`, it is a function of `y` and `X`, and you are using the same `y` and `X` in all three calculations, so the error is roughly held constant. I say roughly because the error in the error calculation is also a function of `b` and `b` is not being held constant; but it is precisely the effect of the different `b`'s that we want to evaluate, so we will have to accept some contamination in our results. The various `b` vectors are nearly identical anyway, so the variation in the contamination cannot be much.

I, however, want to compare results from unscaled and rescaled data, which is to say, the `y` and `X` that will be used in `sum(y-X*b)` will differ, and thus the error in calculating the error could differ across comparisons. To prevent that, after obtaining `b` from each

method, I made the error calculations on the scaled data in all cases, which is to say, on the same `y` and `X`. Thus, when calculating errors for **b** calculated on unscaled data, I multiplied the calculated `weight` coefficient by 100 and divided the other calculated coefficients by 10 to put them on the scale for data that had `mpg` divided by 10 and `weight` divided by 1,000. Multiplication and division introduce no error on modern digital computers (proof omitted). That rescaling, however, allowed `sum(y-X*b)` to be calculated using the same `y` and `X` in all cases, and thus I held roughly constant the error in the error calculation. My adjustment also resulted in more accurate results, but that is for other reasons I am not going to explain here because it is not necessary that my results be more accurate. It is sufficient that I have held the error in the error calculation roughly constant.

By the way, I have still not told you what the true error is because I do not know it. To calculate the true error, I would have to calculate yet another rescaling that would minimize the error in the error calculation, and then I would report to you the error `y-sum(X*b)` calculated using that data, and I would add a plus-or-minus to the end of the reported result that represented the error in the error calculation itself.

All of which is a long way of saying that you should think about putting all your variables on roughly the same scale when you are not willing to think through the numerical issues.

4 Using putmata on subsets of observations

Assume we have the following code:

```
putmata y=mpg X=(weight length 1)
mata:
b = pinv(X)*y
b
end
```

where `b = pinv(X)*y` is standing in for some more complicated calculation you wish to make.

Say that we now wish to run this code on only the foreign cars in the data. We would modify the `putmata` command; the Mata code would remain unchanged:

```
putmata y=mpg X=(weight length 1) if foreign
mata:
b = pinv(X)*y
b
end
```

Whereas previously `y` would have been 74×1 and `X` would have been 74×3, now `y` will be 22×1 and `X` will be 22×3 because `foreign` is true in 22 observations in the data.

Say that we want to run on all our data, but this time, let's assume variables `mpg`, `weight`, and `length` have missing values. They do not in the automobile data, but we will imagine we are using some other dataset. The missing values in `y` and `X` will result

in a 3×1 vector `b` containing missing values. If we want to run on only the complete observations, one solution would be

```
putmata  y=mpg  X=(weight length 1)  if mpg<. & weight<. & length<.
mata:
b = pinv(X)*y
b
end
```

An easier solution is

```
putmata  y=mpg  X=(weight length 1), omitmissing
mata:
b = pinv(X)*y
b
end
```

The `omitmissing` option omits observations with missing values in any of the variables to which we refer. If you specify `omitmissing`, it is important that you specify all the vectors and matrices you want to create with a single `putmata` command. If we typed `putmata y=mpg, omitmissing` and `putmata X=(weight length 1), omitmissing`, then vector `y` would omit observations for which `mpg` contains missing and `X` would omit observations for which `weight` or `length` contain missing, with the result that the `X` and `y` might not be conformable or, worse, be conformable but omit different observations.

5 The getmata command

5.1 Description

`getmata` is the reverse of `putmata`—it creates Stata variables from Mata vectors and matrices. In many cases, you will not need `getmata`. In the problems above, it was sufficient merely to report results. We used `putmata` to put our data into Mata, and we used Mata to calculate and display results. In other problems, you may create new vectors in Mata and need to put them back as variables in your data.

Here is a simplified version of a real problem that showed up on Statalist: You need to create new Stata variable d from existing Stata variable c, to be defined as

$$d_i = \sum_{j|c_j>c_i} (c_j - c_i)$$

where i and j index observations. This problem can be solved in Stata, but it is easier to solve it in Mata because the Mata code we write is nearly identical to the mathematical statement of the problem. If `c` and `d` were Mata vectors, the code would be

```
d = J(rows(c), 1, 0)
for (i=1; i<=rows(c); i++) {
        for (j=1; j<=rows(c); j++) {
                if (c[j]>c[i]) d[i] = d[i] + (c[j] - c[i])
        }
}
```

The most difficult part of this solution to understand is the first line, `d = J(rows(c), 1, 0)`, and that is only because you may not be familiar with Mata's `J()` function. `d = J(rows(c), 1, 0)` creates a `rows(`*c*`)` × `1` column vector of `0`s. The arguments of `J()` are in just that order.

`c` is not a vector in Mata, however. We already know how to solve that:

```
. putmata c
```

It will hardly surprise you to learn that the way we get Mata vector `d` back into Stata afterward is

```
. getmata d
```

5.2 Syntax

Before we put all this together, let me describe the `getmata` command, the syntax of which is

`getmata` *getlist* [`, double` [`update`|`replace`] `id(`*name*`) force`]

A *getlist* is much like a *putlist*, but reversed. A *getlist* can be any combination of the following:

vecname
varname`=`*vecname*
`(`*varname varname ... varname*`)=`*matname*
`(`*varname*`*)=`*matname*

For example,

1. You can type `getmata x1` to create in Stata the new variable `x1` containing the contents of Mata vector `x1`.
2. You can type `getmata x1, update` to create or replace in Stata the variable `x1` containing the contents of Mata vector `x1`.
3. You can type `getmata x1 x2` to create in Stata the new variables `x1` and `x2` containing the contents of Mata vectors `x1` and `x2`.
4. You can type `getmata x1 x2, update` to create or replace in Stata the variables `x1` and `x2` containing the contents of Mata vectors `x1` and `x2`.
5. You can type `getmata (firstvar secondvar) = X` to create in Stata the new variables `firstvar` and `secondvar` containing the first and second columns of matrix `X`. `X` must be $N \times 2$. If `X` had three columns, then you would need to specify three variable names. Obviously, this construction can be used with the `update` option, as can all `getmata` constructions, so I will not mention it again.
6. You can type `getmata (myvar*) = X` to create in Stata the new variables `myvar1`, `myvar2`, ..., equal to the first, second, ..., columns of Mata matrix `X`.

5.3 Options

`double` creates new numeric variables as `double`s rather than the default `float`.

`update` or `replace` allows a vector to be placed in an existing variable. The two options have the same meaning unless the `id()` option is also specified.

`id(`*name*`)` is the topic of an entire section below.

`force` allows getting vectors that have fewer or more columns than observations in the data. You should never have to specify this option.

6 Using putmata and getmata

So now we can put together the solution of creating d from c. To remind you, we wish to create new variable d from existing variable c, where

$$d_i = \sum_{j|c_j>c_i} (c_j - c_i)$$

To show you that the solution works, I use a dataset containing the integers from 1 to 4. The solution is

```
. list

     +---+
     | c |
     |---|
  1. | 1 |
  2. | 2 |
  3. | 3 |
  4. | 4 |
     +---+

. putmata c
(1 vector posted)
. mata
------------------------------------------------- mata (type end to exit) -----
: d = J(rows(c), 1, 0)
: for (i=1; i<=rows(c); i++) {
>         for (j=1; j<=rows(c); j++) {
>                 if (c[j]>c[i]) d[i] = d[i] + (c[j] - c[i])
>         }
> }
: end
--------------------------------------------------------------------------------

. getmata d
```

```
. list
```

	c	d
1.	1	6
2.	2	3
3.	3	1
4.	4	0

If I had to solve this problem, I would package my solution as a do-file.

begin `myfile1.do`

```
version 11                                              // see note 1

clear mata                                              // see note 2
capture drop d                                          // see note 3

putmata c

mata:                                                   // see note 4
d = J(rows(c), 1, 0)
for (i=1; i<=rows(c); i++) {
        for (j=1; j<=rows(c); j++) {
                if (c[j]>c[i]) d[i] = d[i] + (c[j] - c[i])
        }
}
end

getmata d
```

end `myfile1.do`

Notes:

1. Do-files should always begin with a `version` statement. That is what ensures that the do-file continues to work in years to come as new versions of Stata are released.

2. The do-file should not depend on Mata having certain vectors, matrices, or programs already loaded. To ensure this is true, we clear Mata.

3. It was easier for me to debug this do-file if I did not have to remember to `drop d` each time I reran it.

4. I coded `mata:` (`mata` with a colon), yet previously when I used Mata interactively, I omitted the colon. Coding `mata:` tells Mata to stop if any error occurs, which is exactly how I want my do-file to behave. Using `mata` without the colon tells Mata not to stop, but to instead give me an opportunity to fix what I mistyped, which is how I work interactively.

7 Using putmata and getmata on subsets of observations

In the example above where we created variable `d` from `c`, we assumed that there were no missing values in `c`, or at least we did not consider the issue. It turns out that our code produces several missing values in the presence of just one missing value. Below I have already dropped the data used in the previous example and have entered another dataset:

```
. list
```

	c
1.	1
2.	2
3.	.
4.	3
5.	4

```
. do myfile1.do
```
(*output omitted*)
```
. list
```

	c	d
1.	1	.
2.	2	.
3.	.	0
4.	3	.
5.	4	.

We could modify the Mata code in `myfile1.do` to watch for missing values and to exclude them from the calculation, but we already know an easier way. Rather than creating Mata vector `c` to include all the observations from Stata variable `c`, we could create the vector to include only the nonmissing values by changing `putmata c` to read

```
putmata c if c<.
```

or

```
putmata c, omitmissing
```

The result of either of those commands will be to create vector `c` to be 4×1 rather than 5×1.

There is, however, an issue. At the end of our code where we post the Mata solution vector `d` to new Stata variable `d`—we coded `getmata d`—we will need to specify which five observations are to receive the four calculated results. `getmata` has a syntax for that, but before we can use it, we will need a variable that uniquely identifies the observations. In real data, you would be likely to already have such a variable, but in case you do not, it is easy to create such a variable. You type `generate` *newvar* `= _n`. Let's create such a variable in our data:

```
. generate fid = _n
. list
```

	c	fid
1.	1	1
2.	2	2
3.	.	3
4.	3	4
5.	4	5

`fid` is a perfectly good identification variable, but I am about to multiply `fid` by 10 just to emphasize to you that the identification variable does not have to correspond to observation numbers.

```
. replace fid = fid*10
(5 real changes made)
. list
```

	c	fid
1.	1	10
2.	2	20
3.	.	30
4.	3	40
5.	4	50

An identification variable is a variable that takes on different values for each observation in the data. The values could be 1, 2, ...; or they could be 1.25, –2, ...; or they could be Nick, Mary, and so on. The values can be numeric or string, and they can be in any order. All that is important is that the variable contain distinct values for each observation.

Now that we have an identification variable, we can modify the ending `getmata` command to read

```
getmata d, id(fid)
```

instead of just `getmata d`. The `id(fid)` option specifies that values in variable `fid` are to be matched with the values in vector `fid` to determine the observations of variable `d` that are to be filled in from vector `d`. For that to work, we must post to Mata the values of `fid`, so the entire solution reads

```
putmata fid c, omitmissing
mata:
Mata code goes here
end
getmata d, id(fid)
```

When we `putmata fid c, omitmissing` with our example data, two 4×1 vectors will be created in Mata, `fid` and `c`. The vectors will contain values from observations 1, 2, 4, and 5, omitting observation 3 because `c==.` in that observation. Thus vector

`fid` will contain $(10, 20, 40, 50)'$. Later, at the end of our code, when we `getmata d, id(fid)`, Stata will compare the contents of vector `fid` $= (10, 20, 40, 50)'$ with the values of variable `fid`, and Stata will be able to work out that vector row 1 corresponds to observation 1, row 2 corresponds to observation 2, row 3 to observation 4, and row 4 to observation 5. In this example, `fid` increases with observation number, but that is not required.

Our updated do-file reads

begin `myfile2.do`

```
version 11

clear mata
capture drop d

putmata fid c, omitmissing                              // (changed)

mata:
d = J(rows(c), 1, 0)
for (i=1; i<=rows(c); i++) {
        for (j=1; j<=rows(c); j++) {
                if (c[j]>c[i]) d[i] = d[i] + (c[j] - c[i])
        }
}
end

getmata d, id(fid)                                      // (changed)
```

end `myfile2.do`

Here is the result of running the do-file:

```
. list
```

	c	fid
1.	1	10
2.	2	20
3.	.	30
4.	3	40
5.	4	50

```
. do myfile2
  (output omitted)
. list
```

	c	fid	d
1.	1	10	6
2.	2	20	3
3.	.	30	.
4.	3	40	1
5.	4	50	0

8 Using views

When you type or code `putmata x`, vector `x` is created as a copy of the Stata variable `x`. The variable and vector are separate things. An alternative is to make the Mata vector a view onto the Stata variable. You do that by typing `putmata x, view`. Now the variable and vector share the same recording of the values. Views use less memory than copies, although views are slightly less efficient in terms of execution time. Views have other advantages and disadvantages, too.

Say that you type `putmata x` and then, in Mata, code `x[1]=20`. Changing vector `x` leaves the variable `x` unchanged. If you had typed `putmata x, view`, however, changing vector `x` would simultaneously change variable `x`, because the variable and the vector are the same thing. Sometimes, that is an advantage. At other times, it is a disadvantage.

There is more to know. If you are working with views and, in the middle of the Mata session, take a break and return to Stata, it is important that you do not modify the Stata data in certain ways. When you create a view, Stata records notes about the mapping. Those notes might read that variable vector `x` is a view onto variable 3, observations 2 though 20 and observation 39. If you change the sort order of the data, the view will still be working with observations 2 through 20 and 39 even though those observations now contain different data! If you were to drop the first or second variable, the view would still be working with variable 3 even though that will now be a different variable! Alternatively, if you update variable 3 with improved values, those improvements will appear in the Mata vector, too.

The memory savings offered by views is considerable when working with large datasets. Say that you have a dataset containing 1,000,000 observations on 200 variables. That dataset might be 800,000,000 bytes in size, or 763 megabytes. (To obtain megabytes, you divide by $1{,}024^2$.) Typing `putmata *` would create copies of each variable, meaning creation of two hundred 1,000,000-element double-precision vectors. You would just have consumed another $200 \times 1{,}000{,}000 \times 8/1{,}024^2 = 1{,}526$ megabytes of memory, or $1{,}526/1{,}024 = 1.5$ gigabytes. Typing `putmata *, view`, however, would consume only 24 or so kilobytes of memory, a practically insignificant amount.

All the examples shown so far work equally well with copies or views. We simply would need to add the `view` option to the `putmata` commands.

If we are going to work with views, we could make `d` a view, too. If we make `d` a view, we can eliminate the `getmata` commands at the end of our code, because views are the variable and thus they put themselves back. This even means we could eliminate the `fid` variable because views will handle their own alignment of vectors and variables.

Remember that the do-file creates new variable `d` from existing variable `c`. We modify the do-file to create new variable `d` at the outset, in Stata, and then create views onto both `c` and `d`.

In the creation of those views, we can omit the observations that have `c>=.` by simply including the `omitmissing` option with `putmata`. Finally, we delete the now irrelevant `getmata` command at the end. Our code reads

begin `myfile3.do`

```
version 11

clear mata
capture drop d
generate d = 0                                          // see note 1

putmata c d, omitmissing view                           // see note 2

mata:
d[.] = J(rows(c), 1, 0)                                 // see note 3
for (i=1; i<=rows(c); i++) {
        for (j=1; j<=rows(c); j++) {
                if (c[j]>c[i]) d[i] = d[i] + (c[j] - c[i])
        }
}

end

replace d=. if c==.                                     // see note 4
                                                        // see note 5
```

end `myfile3.do`

Notes:

1. We now create new variable `d` at the outset. We create it containing 0, not missing values. That is important because we are about to issue a `putmata` command with the `omitmissing` option, and we do not want the missing values in `d` to cause all the observations to be omitted.

2. We include the `view` option on the `putmata` command, and we include variable `d`.

3. We could have deleted this line, but instead I modified it to remind you not to make a terrible error. The line `d[.] = J(rows(c), 1, 0)` fills in `d` with zeros. I could have omitted the line because `d` is already filled with zeros. I did not delete it because I wanted an excuse to call your attention to the left-hand side of the assignment. I changed what was previously

   ```
   d = J(rows(c), 1, 0)
   ```

   to

   ```
   d[.] = J(rows(c), 1, 0)
   ```

   I changed `d` to `d[.]`. That change is of great importance. What we previously coded created vector `d`. What I now code changes the values stored in existing vector `d`. If I left what we coded previously, Mata would discard the view stored in `d` and create a new `d` as a regular Mata vector unconnected to Stata. Our Mata code would have worked, but none of the values stored in regular vector `d` would have made it back to Stata variable `d`.

4. We add the line `replace d=. if c<=.`. I admit that was something that I discovered I needed to add the first time I tested this do-file and looked at the output.

What I saw was that `d` = 0 in the observation in which `c==.`. That happened because we created `d` containing zeros at the outset. It would have been better if we had created `d` by coding

```
generate d = 0 if c<.
```

rather than `generate d = 0`. I left the mistake in, however, to show that the author is not infallible.

5. We omit the line `putmata d` or `putmata d, id(fid)`. Vector `d` is variable `d`. We need not worry about alignment because when the view `d` was created, it was created as a view onto only the relevant observations.

My personal opinion concerning views is that I avoid them for variables that appear on the left-hand side of the assignment operator. That is, I would have left `d` as a regular vector and left in the `getmata d, id(fid)`. If you review the above notes, all the complication was caused by `d` being a view. I had to remember to code `d[.] =` ... rather than `d =`, which I invariably forget. I cannot fill `d` with missing at the outset because `putmata, omitmissing` will then omit all the observations. Concerning the latter, there are more clever ways I could have handled that. I could have filled in `d` with 0 and performed the `putmata`, as I did, and then immediately changed the contents of `d` to be missing. Even so, I try to avoid using views for variables to which I will be making assignments. I do use views for right-hand-side variables because, in that case, views have no implications for subsequent code.

Anyway, this do-file works:

```
. list
```

	c
1.	1
2.	2
3.	.
4.	3
5.	4

```
. do myfile3
  (output omitted)
. list
```

	c	d
1.	1	6
2.	2	3
3.	.	.
4.	3	1
5.	4	0

9 Conclusion

Some problems are more easily solved in Mata than in Stata. In fact, Mata and Stata complement each other well because problems that are easy in one are often difficult in the other. With `putmata`, it is easy to move your data into Mata. With `getmata`, you can move data back from Mata to Stata if necessary. I showed two classes of examples:

1. *Analysis*. In analysis situations, you use `putmata`, but you do not need `getmata`. I showed how to obtain $\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$, and I showed that the same results could be obtained by $\mathbf{b} = \mathbf{X}^{-1}\mathbf{y}$ for a suitable definition of matrix inversion. Both of these examples would be useful in teaching, but you are to imagine that these simple formulas stand in for more lengthy calculations implementing the latest result found in the professional journals. I once gave a talk where I dropped into Mata to calculate a generalized method of moments estimator for a Poisson model with an endogenous variable, and I did so in a dozen or so lines of Mata code using the formulas right from the original paper. Stata now does generalized method of moments, so there is no reason to rehash an old talk here.

2. *Data management*. I showed how to create a difficult-to-calculate variable using Mata. Here you use `putmata` to get the data into Mata, and you use `getmata` to get the result back into Stata. Stata is wonderful at data management and most complicated tasks are made easy. Every so often, however, one comes upon a problem where the Stata solution is elusive. There is one, you know, and usually it requires only a few lines, but you cannot imagine what they might be. In such cases, it is usually quicker to drop into Mata and go directly at the solution.

`putmata` and `getmata` are useful commands, but bear in mind that they were designed to help solve custom data analysis problems: the types of problems that arise in a particular analysis and that one solves in do-files. They were not designed for use by programmers coding general solutions implemented as ado-files. `putmata` and `getmata` create and work with global vectors and matrices, and that is why their results are so easy to use. That same feature makes them inappropriate for ado-files. Programmers writing ado-files need results stored in local vectors and matrices. Stata already has tools for creating such local vectors and matrices, namely, `st_data()`, `st_view()`, and `st_store()`; see [M-5] **st_data( )** and [M-5] **st_view( )**. Programmers may wish to think of `putmata` as `st_data()` and `st_view()`, and `getmata` as `st_store()`, for interactive and do-file use.

About the author

William Gould is president of StataCorp, head of development, and principal architect of Mata.

The Stata Journal (2010)
10, Number 1, pp. 143–151

Speaking Stata: The statsby strategy

Nicholas J. Cox
Department of Geography
Durham University
Durham City, UK
n.j.cox@durham.ac.uk

Abstract. The `statsby` command collects statistics from a command yielding r-class or e-class results across groups of observations and yields a new reduced dataset. `statsby` is commonly used to graph such data in comparisons of groups; the `subsets` and `total` options of `statsby` are particularly useful in this regard. In this article, I give examples of using this approach to produce box plots and plots of confidence intervals.

Keywords: gr0045, statsby, graphics, groups, comparisons, box plots, confidence intervals

1 Introduction

Datasets are often subdivided at one or more levels according to some kind of group structure. Statistically minded researchers are typically strongly aware of the need for, and the value of, comparisons between patients, hospitals, firms, countries, regions, sites, or whatever the framework is for collecting and organizing their data. Indeed, for many people, that kind of comparison is at the heart of what they do daily within their research.

Stata supports separate group analyses in various ways. Perhaps the most well-known and important is the `by:` construct, a subject of one of the earliest *Speaking Stata* columns (Cox 2002). This column focuses on [D] **statsby**, a command that until now has received only passing mention in *Speaking Stata* (Cox 2001, 2003). The main idea of `statsby` is simple and it is well documented. However, experience on Statalist and elsewhere indicates that many users who would benefit from `statsby` are unaware of its possibilities. The extra puff of publicity here goes beyond the manual entry in stressing its potential for graphical comparisons.

Focusing exclusively on `statsby` is not intended as a denial that there are other solutions to the same, or related, problems. The work of Newson (1999, 2000, 2003) is especially notable in this regard and goes beyond the singular purpose explored here.

2 The main idea

The main idea of `statsby` is that it offers a framework, not only for automating separate analyses for each of several groups, but also for collating the results. The effect is to relieve users of much of the tedious organizing work that would be needed otherwise. The

default mode of operation is that `statsby` overwrites the original dataset, subdivided in some way, with a reduced dataset with just one observation for each group. The `saving()` option, however, permits results to be saved on the side so that the original dataset remains in memory.

A common and essentially typical example of applying `statsby` is that a panel dataset containing one or more observations for each panel would be reduced to a dataset with precisely one observation for each panel. Those observations contain panel identifiers together with results for each panel, usually e- or r-class results from some command. Because there is no stipulation that the command called is an official command, there is scope for users to write their own programs leaving such results in their wake and thus to automate essentially any kind of calculation.

`statsby` does not support graphs directly, but the implications for graphics are immediate. Graphics for groups imply the collation of group results followed by graphing operations. Using `statsby` can reduce the problem to just the second of these two, subject as usual to minor questions of titling, labeling, and so forth.

3 Box plots for all possible subsets

I will not recapitulate the details of the manual entry, which those unfamiliar with the command can read for themselves. Rather, I will underline the value of `statsby` by showing how it makes several graphical tasks much easier.

Variants on box plots remain popular in statistical science. In an earlier column (Cox 2009), I underlined how `graph twoway` allows your own alternatives if ever the offerings of `graph box` or `graph hbox` are not quite suitable.

Let us pick up that theme and give it a new twist. The `subsets` option of `statsby` makes easy a division into all possible subsets of a dataset. That can be useful so long as you remember enough elementary combinatorics to avoid trying to produce an impracticable or impossible graph.

We will use the `sj` scheme standard for the *Stata Journal* and `auto.dta` bundled with Stata.

```
. set scheme sj
. sysuse auto
(1978 Automobile Data)
```

A small piece of foresight—benefiting from the hindsight given by earlier attempts excised from public view—is now to save a variable label that would otherwise disappear on reduction. In this example, we could also just type in the label or some other suitable text afterward. But if you try something similar yourself, particularly if you want to automate the production of several graphs, the small detail of saving text you want as a graph title may avoid some frustration.

```
. local xtitle "`: var label mpg´"
```

Our call to `statsby` spells out that we want five quantiles, the median, two quartiles, and two extremes. `summarize, detail` does that work. Because `summarize` is an r-class command, we need to look up the codes used, either by reverse engineering from the results of `return list` or by looking at the command help or the manual entry.

The principle with an e-class command is identical, except that we would reverse engineer from `ereturn list`. If this detail on r- and e-class results goes beyond your present familiarity, start at `help saved results` and follow the documentation pointers there if and as desired.

We are subdividing `auto.dta` by the categorical variables `foreign` and `rep78`, but with a twist given by the `subsets` option. Another useful option—in practice, probably even more useful—is to use the `total` option to add results for the whole set.

```
. statsby p50=r(p50) p25=r(p25) p75=r(p75) min=r(min) max=r(max),
> by(foreign rep78) subsets total: summarize mpg, detail
(running summarize on estimation sample)

      command:  summarize mpg, detail
          p50:  r(p50)
          p25:  r(p25)
          p75:  r(p75)
          min:  r(min)
          max:  r(max)
           by:  foreign rep78

Statsby subsets
-+- 1 -+- 2 -+- 3 -+- 4 -+- 5
.................
```

Let us look at the results. To emphasize the key point: This is a reduced dataset and the original dataset is gone, although overwriting is avoidable through `saving()`. We have results for all combinations of `foreign` and `rep78` that exist in the data; for all categories of `foreign` and for all categories of `rep78`; and for all observations.

(*Continued on next page*)

```
. list
```

	foreign	rep78	p50	p25	p75	min	max
1.	Domestic	1	21	18	24	18	24
2.	Domestic	2	18	16.5	23	14	24
3.	Domestic	3	19	16	21	12	29
4.	Domestic	4	18	15	21	14	28
5.	Domestic	5	32	30	34	30	34
6.	Domestic	.	19	16	22	12	34
7.	Foreign	3	23	21	26	21	26
8.	Foreign	4	25	23	25	21	30
9.	Foreign	5	25	18	35	17	41
10.	Foreign	.	25	21	28	17	41
11.	.	1	21	18	24	18	24
12.	.	2	18	16.5	23	14	24
13.	.	3	19	17	21	12	29
14.	.	4	22.5	18	25	14	30
15.	.	5	30	18	35	17	41
16.	.	.	20	18	25	12	41

Plotting that data directly would produce a reasonable working graph. Largely as a matter of personal taste, I chose to reorganize and edit the data slightly to get something more attractive. First, I wanted all two-group categories together, then all one-group categories, and then all the data. The number of groups in each category is the complement of the number of missing values of the first two variables in each observation or row, so that can be calculated by counting missing values in each row and `sort`ing accordingly. The `stable` option minimizes departure from the present sort order.

```
. egen order = rowmiss(foreign rep78)
. sort order, stable
```

To get group labels, I combine the value labels (where used) and the values (otherwise) with `egen`'s `concat()` function, and I remove the periods indicating missing and any marginal spaces:

```
. egen label = concat(foreign rep78), decode p(" ")
. replace label = trim(subinstr(label, ".", "", .))
(8 real changes made)
```

The total category for results for all observations deserves due prominence:

```
. replace label = "Total" in L
(1 real change made)
```

The final detail of preparation is to use a couple of helper programs to set up one axis variable with gaps and to map the values in the `label` variable to the value labels of that axis variable:

```
. seqvar x = 1/5 7/9 11/12 14/18 20
. labmask x, values(label)
```

For more details on `seqvar` and `labmask`, see Cox (2008).

Now we assemble the box plot from ingredients produced by members of the `twoway` family. `rspike` draws spikes between each quartile and each tailward extreme. `rbar` draws boxes between the quartiles. `scatter` draws point symbols showing the medians. The result is shown in figure 1. At this point, we use the title carefully stored in a local macro before the call to `statsby`.

```
. twoway rspike min p25 x, horizontal bcolor(gs12) ||
> rspike p75 max x, horizontal bcolor(gs12) ||
> rbar p25 p75 x, horizontal barw(0.8) bcolor(gs12) ||
> scatter x p50, ms(O) yla(1/5 7/9 11/12 14/18 20, val nogrid noticks ang(h))
> legend(off) ysc(reverse) xtitle(`xtitle´)
```

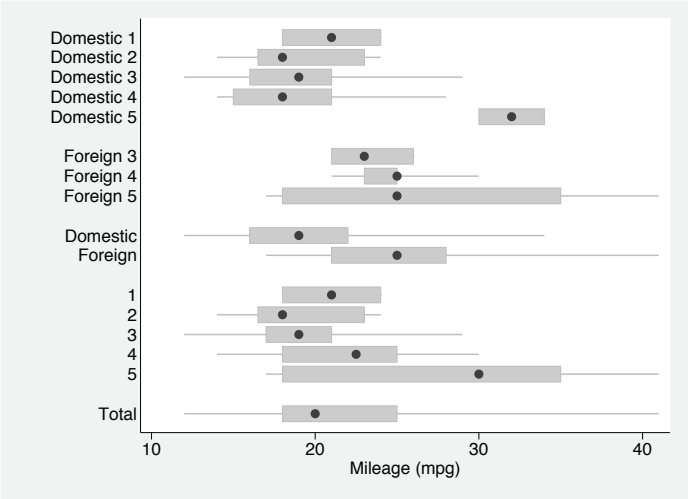

Figure 1. All subsets box plot of mileage for 78 cars by domestic or foreign origin, repair record in 1978, and combinations thereof. Spikes extend to extremes, boxes show quartiles, and circles show medians.

Beyond question, the statistical and stylistic choices here of what to show and how to show it are all arguable and variable. However, that is not the main point. Rather, you should appreciate how `statsby` with its `subsets` and `total` options made a different kind of plot much easier.

(*Continued on next page*)

4 Confidence-interval plots

Plotting confidence intervals for some group statistic, such as the mean, is another common application. The basic trick, which now starts to look fairly obvious, is to use a command such as `ci` (see [R] **ci**) under the aegis of `statsby` to produce a reduced dataset that is then ready for graphics.

We read in the U.S. National Longitudinal Survey data available from Stata's web site. We will look at the relationship between wage (on an adjusted logarithmic scale) and highest education grade.

```
. webuse nlswork, clear
(National Longitudinal Survey.  Young Women 14-26 years of age in 1968)
```

Saving the variable label is a trick we saw before. At worst, it does no harm.

```
. local ytitle "`: var label ln_wage´"
```

`ci` is our workhorse. The default gives 95% confidence intervals, but evidently other choices may suit specific purposes. As the dataset is panel data, we need to decide how far to respect its structure. One of several possible approaches is to select one observation from each panel randomly (but reproducibly). In addition to estimates and confidence intervals, we save the sample sizes, which are a key part of the information.

```
. set seed 2803
. generate rnd = runiform()
. bysort idcode (rnd): generate byte select = _n == 1
. statsby mean=r(mean) ub=r(ub) lb=r(lb) N=r(N) if select, by(grade) clear:
> ci ln_wage
(running ci on estimation sample)
      command:  ci ln_wage if select
         mean:  r(mean)
           ub:  r(ub)
           lb:  r(lb)
            N:  r(N)
           by:  grade
Statsby groups
─────┼─── 1 ───┼─── 2 ───┼─── 3 ───┼─── 4 ───┼─── 5
(2 missing values generated)
..................
```

Here the grades run over all the integers 0/18, but `levelsof` (see [P] **levelsof**) simplifies capture for later graphical use of all values that occur, especially in more complicated cases.

```
. levelsof grade, local(levels)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
```

A basic plot is now at hand. Using `scatter` for the means and `rcap` for the intervals themselves is widely conventional. A delicate detail is that means on top of intervals look better than the converse. The result is shown in figure 2.

```
. twoway rcap ub lb grade || scatter mean grade, yti(`ytitle´) legend(off)
> subtitle(95% confidence intervals for mean, place(w)) xla(`levels´)
```

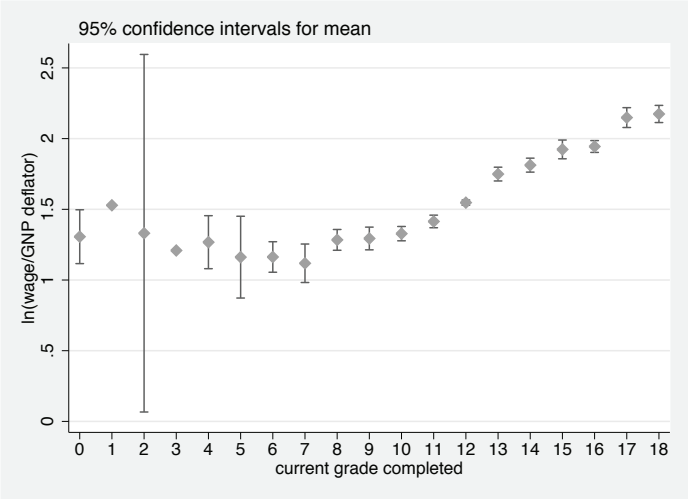

Figure 2. Graph of 95% confidence intervals of mean adjusted log wage by education grade.

The graph can be improved in various minor cosmetic ways and also by showing sample sizes. After some experimenting, the method for the latter was refined to showing sizes as marker labels on a horizontal line. Horizontal alignment of those labels would have been preferable, except that they then would run into one another. Exchanging axes so that grade is plotted vertically seems too awkward for this kind of data. In other circumstances, exchanging axes might well be a good idea. Thus the vertical alignment here is regarded as the lesser of two evils. Figure 3 shows the result.

```
. generate where = 2.7
. twoway rcap ub lb grade || scatter mean grade, yti(`ytitle´) legend(off)
> subtitle(95% confidence intervals for mean, place(w)) xla(`levels´)
> || scatter where grade, ms(none) mla(N) mlabangle(v) mlabpos(0) ysc(r(. 2.8))
> yla(0(.5)2.5, ang(h))
```

(*Continued on next page*)

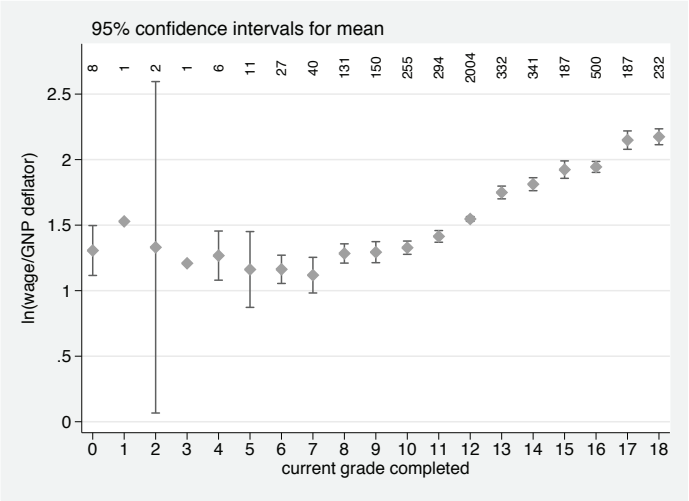

Figure 3. Graph of 95% confidence intervals of mean adjusted log wage by education grade. Text labels show sample sizes at each grade.

5 Conclusions

This column has promoted one simple idea, using `statsby` to prepare a reduced dataset for subsequent graphing. Its `subsets` and `total` options allow useful variations on the default. You might still need to do some further work to get a good graph, but the overall labor is nevertheless likely to be much reduced. The method is widely applicable in so far as any calculation can be represented by a program as yielding r-class or e-class results.

6 Acknowledgments

Vince Wiggins planted the immediate seed for this column with a single cogent remark. Martin Weiss has been an energetic proponent of `statsby` on Statalist.

7 References

Cox, N. J. 2001. Speaking Stata: How to repeat yourself without going mad. *Stata Journal* 1: 86–97.

———. 2002. Speaking Stata: How to move step by: step. *Stata Journal* 2: 86–102.

———. 2003. Speaking Stata: Problems with tables, Part I. *Stata Journal* 3: 309–324.

———. 2008. Speaking Stata: Between tables and graphs. *Stata Journal* 8: 269–289.

———. 2009. Speaking Stata: Creating and varying box plots. *Stata Journal* 9: 478–496.

Newson, R. 1999. dm65: A program for saving a model fit as a dataset. *Stata Technical Bulletin* 49: 2–6. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 19–23. College Station, TX: Stata Press.

———. 2000. dm65.1: Update to a program for saving a model fit as a dataset. *Stata Technical Bulletin* 58: 2. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, p. 7. College Station, TX: Stata Press.

———. 2003. Confidence intervals and p-values for delivery to the end user. *Stata Journal* 3: 245–269.

About the author

Nicholas Cox is a statistically minded geographer at Durham University. He contributes talks, postings, FAQs, and programs to the Stata user community. He has also coauthored 15 commands in official Stata. He wrote several inserts in the *Stata Technical Bulletin* and is an editor of the *Stata Journal*.

The Stata Journal (2010)
10, Number 1, pp. 152–156

Stata tip 83: Merging multilingual datasets

Devra L. Golbe
Hunter College, CUNY
New York
dgolbe@hunter.cuny.edu

`merge` is one of Stata's most important commands. Without specific instructions to the contrary, `merge` holds the master data file inviolate. Its variables are neither replaced nor updated. Variable and value labels are retained. All these properties and more are well documented. What is not so well documented is how `merge` interacts with Stata's multiple language support (`label language`), added in Stata 8.1 and described in Weesie (2005). In essence, Stata users must pay careful attention to which languages are defined and current when merging files.

The language feature is useful not only for multiple "real" languages (e.g., English and French) but also for using different sets of labels for different purposes, such as short labels ("Mines") and long ("Non-Metallic and Industrial Metal Mining") in one data file. `merge` may generate unexpected results if attention is not paid to the language definitions and, in particular, the current language in each file. Multilingual datasets to be merged should be defined with common languages and each should have the same language set as the current language.

I illustrate using `auto.dta`. Starting with `autotech.dta`, create a new dichotomous variable, `guzzler`, defined as mpg < 25, label the variable and its values in English (en) and French (fr), and save to a file called `tech.dta`. The tabulations below display variable and value labels:

```
. label language en
. tabulate guzzler
```

Gas Guzzler	Freq.	Percent	Cum.
No	19	25.68	25.68
Yes	55	74.32	100.00
Total	74	100.00	

```
. label language fr
. tabulate guzzler
```

Verte	Freq.	Percent	Cum.
Oui	19	25.68	25.68
Non	55	74.32	100.00
Total	74	100.00	

From `auto.dta`, create an English-labeled file, `origin.dta`. Now merge in `tech.dta` and tabulate `guzzler` and `foreign`:

```
. tabulate guzzler foreign
```

Verte	Car type: Domestic	Foreign	Total
Oui	8	11	19
Non	44	11	55
Total	52	22	74

`foreign` is labeled in English, but `guzzler` is labeled in French. How did that happen? The `label language` and `labelbook` commands can clarify:

```
. label language
Language for variable and value labels
    In this dataset, value and variable labels have been defined in only one
    language:  en
  (output omitted)
. labelbook
-------------------------------------------------------------------------------
value label origin
-------------------------------------------------------------------------------
  (output omitted)
  definition
           0   Domestic
           1   Foreign
   variables:  foreign
-------------------------------------------------------------------------------
value label yesno_en
-------------------------------------------------------------------------------
  (output omitted)
  definition
           0   No
           1   Yes
   variables:
-------------------------------------------------------------------------------
value label yesno_fr
-------------------------------------------------------------------------------
  (output omitted)
  definition
           0   Oui
           1   Non
   variables:  guzzler
  (output omitted)
```

On reflection, this is what we might have expected. The master dataset is inviolate in the sense that its language—English (and only English)—is preserved. The using dataset has two languages, but only the labels from the current language (French) are attached to the single language (English) in the master dataset.

We could, of course, redefine our languages and reattach the appropriate labels. However, if we plan to merge our master file with a multilingual dataset, a better strategy is to prepare the master file by defining the same two languages and then merge. It is crucial that the current languages are the same in both files. To illustrate, add French labels to `origin.dta`. Then consider what happens if the current languages differ. Suppose that the current language in the master file (`origin.dta`) is French, but the current language in the using file (`tech.dta`) is English. Then the labels from the current language in the using file (English) are attached in the current language of the master file (French), and the French labels (noncurrent) from the using file are not attached at all:

```
. label language fr
(fr already current language)
. tabulate guzzler foreign
```

Gas Guzzler	Origine USA	Origine Autre	Total
No	8	11	19
Yes	44	11	55
Total	52	22	74

```
. label language en
. tabulate guzzler foreign
```

guzzler	Car type Domestic	Car type Foreign	Total
0	8	11	19
1	44	11	55
Total	52	22	74

Although we could of course correct the labels afterward, it is easier to make sure that the files are consistent before the merge. If we set the current language to English (or French) in both files before merging, we see that the labels are properly attached:

```
. label language en
(en already current language)
. tabulate guzzler foreign
```

Gas Guzzler	Car type Domestic	Car type Foreign	Total
No	8	11	19
Yes	44	11	55
Total	52	22	74

```
.  label language fr
```

```
. tabulate guzzler foreign
```

Verte	Origine USA	Autre	Total
Oui	8	11	19
Non	44	11	55
Total	52	22	74

Whether or not the labels are properly attached, `merge` does preserve them. But a little planning in advance will ensure that they are attached in the way you expect.

In passing, it is worth noting that the situation is more complex if there are no languages defined in the master file. In that case, issuing a `label language` statement changes the behavior of `merge`. First, let's see what happens if we ignore the language of the master file. Thus we start with the original `autotech.dta` and merge in the multilingual `origin.dta`. We see that two languages have been defined, and the labels are properly attached:

```
. webuse autotech, clear
(1978 Automobile Data)
. merge 1:1 make using origin, nogenerate
  (output omitted)
. tabulate foreign
```

Car type	Freq.	Percent	Cum.
Domestic	52	70.27	70.27
Foreign	22	29.73	100.00
Total	74	100.00	

```
. label language fr
. tabulate foreign
```

Origine	Freq.	Percent	Cum.
USA	52	70.27	70.27
Autre	22	29.73	100.00
Total	74	100.00	

If before merging, however, we check that the master file's only language is the default, we get different results:

```
. webuse autotech, clear
(1978 Automobile Data)
. label language
Language for variable and value labels
    In this dataset, value and variable labels have been defined in only one
    language:  default
  (output omitted)
```

```
. merge 1:1 make using origin, nogenerate
  (output omitted)
. label language
Language for variable and value labels
    In this dataset, value and variable labels have been defined in only one
    language:  default
  (output omitted)
```

In this case, only the default language is defined. Tabulation of `foreign` indicates that labels are attached from only the current language in the using file. The reason is that issuing the `label language` command sets the characteristics that define the current language and all available languages. `merge` then declines to overwrite those characteristics with characteristics from the using dataset. If they are as yet undefined, however, those characteristics are taken from the using language.

Reference

Weesie, J. 2005. Multilingual datasets. *Stata Journal* 5: 162–187.

The Stata Journal (2010)
10, Number 1, pp. 157–159

Stata tip 84: Summing missings

Nicholas J. Cox
Department of Geography
Durham University
Durham City, UK
n.j.cox@durham.ac.uk

Consider this pretend dataset and the result of a mundane `collapse`:

```
. list, sepby(g)
```

	g	y
1.	1	1
2.	1	2
3.	1	3
4.	1	4
5.	2	5
6.	2	6
7.	2	7
8.	3	8
9.	3	.
10.	4	.

```
. collapse (sum) y, by(g)
. list
```

	g	y
1.	1	10
2.	2	18
3.	3	8
4.	4	0

Here we have a grouping variable, `g`, and a response, `y`. When we `collapse, by(g)` to produce a reduced dataset of the sums of `y`, many users are surprised at the results. Clearly $1+2+3+4=10$ and $5+6+7=18$, so no surprises there, but the other results need more comment.

When producing sums, Stata follows two specific rules:

1. Initialize the result at 0 and add pertinent numbers one by one until done. The result is the sum.

2. Ignore missings.

With these two rules, it should not surprise you that the sum $8 + .$ is reported as 8 (the missing is ignored) or that the sum of `.` is reported as 0 (the missing is ignored, and we just see the result of initializing, namely, 0).

Nevertheless, many users see this reasoning as perverse. A counterargument runs like this: If we know that all values in a group are missing, then we know nothing about their sum, which should also be recorded as missing. Users with this view need to know how to get what they want. They should also want to know more about why Stata treats missings in this way.

Consider a related problem: What is the sum of an empty set? Stata's answer to the problem is best understood by extending the problem. Imagine combining that empty set with any nonempty set, say, 1, 2, 3. What is the sum of the combined set? The Stata answer is that as

$$\text{sum of empty set} + \text{sum of } 1, 2, 3 = \text{sum of combined set } 1, 2, 3$$

so also the sum of an empty set must be regarded as 0. Sums are defined not by what they are but by how they behave. Insisting that the sum of an empty set must be missing prevents you from ever combining that result helpfully with anything else.

This problem, which was dubbed "related", is really the same problem in Stata's mind. Because missings are ignored, it is as if they do not exist. Thus a set of values that is all missing is equivalent to an empty set.

If you look carefully, you will find this behavior elsewhere in Stata, so it should be less of a surprise. `summarize` (see [R] **summarize**) will also report the sum of missings as zero:

```
. sysuse auto, clear
(1978 Automobile Data)
. summmarize rep78 if rep78 == .
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       rep78 |          0
. return list
scalars:
                  r(N) =  0
              r(sum_w) =  0
                r(sum) =  0
```

The sum (and sum of weights) is not explicit in the output from `summarize`, but it is calculated on the side and put in r-class results.

`egen` (see [D] **egen**) functions `total()` and `rowtotal()` by default yield 0 for the total of missings. However, as from Stata 10.1, they now have a new `missing` option. With this option, if all arguments are missing, then the corresponding result will be set to missing.

Mata has an explicit variant on empty sets, e.g.,

```
. mata: sum(J(0,0,.))
  0
. mata: sum(J(0,0,42))
  0
```

Whenever this behavior does not meet your approval, you need a work-around. Returning to the first example, one strategy is to include in the `collapse` enough information to identify groups that were all missing. There are several ways to do this. Here is one:

```
. collapse (sum) y (min) min=y, by(g)
. list
```

	g	y	min
1.	1	10	1
2.	2	18	5
3.	3	8	8
4.	4	0	.

```
. replace y = . if missing(min)
(1 real change made, 1 to missing)
. list
```

	g	y	min
1.	1	10	1
2.	2	18	5
3.	3	8	8
4.	4	.	.

The idea here is that the minimum is missing if and only if all values are missing. Using `missing(min)` as a test also catches any situation in which the smallest missing value is one of `.a`–`.z`.

Missings pose problems for any mathematical or statistical software, and it is difficult to design rules that match all users' intuitions. Indeed, some intuitions are changed by becoming accustomed to certain rules. Stata's philosophy of ignoring missings where possible contrasts with software in which missings are regarded as highly infectious, affecting whatever operations they enter. Here is not the place to debate the merits and demerits of different choices in detail but rather to underline that Stata does attempt to follow rules consistently, with the consequence that you need a work-around whenever you disagree with the outcome.

The Stata Journal (2010)
10, Number 1, pp. 160–163

Stata tip 85: Looping over nonintegers

Nicholas J. Cox
Department of Geography
Durham University
Durham, UK
n.j.cox@durham.ac.uk

A loop over integers is the most common kind of loop over numbers in Stata programming, as indeed in programming generally. `forvalues`, `foreach`, and `while` may be used for such loops. See their manual entries in the *Programming Reference Manual* for more details if desired. Cox (2002) gives a basic tutorial on `forvalues` and `foreach`. In this tip, I will focus on `forvalues`, but the main message here applies also to the other constructs.

Sometimes users want to loop over nonintegers. The help for `forvalues` contains an example:

```
. forvalues x = 31.3 31.6 : 38 {
  2.       count if var1 < `x´ & var2 < `x´
  3.       summarize myvar if var1 < `x´
  4. }
```

It is perfectly legal to loop over such a list of numbers, because `forvalues` allows any arithmetic progression as lists, with either integer or noninteger constant difference between successive terms. However, such lists can cause problems. On grounds of precision, correctness, clarity, and ease of maintenance, the advice here is to use loops over integers whenever possible.

The precision problem is exactly that explained elsewhere (Cox 2006; Gould 2006; Linhart 2008). Stata necessarily works at machine level in binary, and so it does no calculations in decimal. Rather, it works with the best possible binary approximations of decimals and then converts to decimal digits for display. Not surprisingly, users often think in terms of decimals that they want to use in their calculations or display in their results. Commonly, the conversions required work well and are not detectable, but occasionally users can get surprising results. Here is a simple example:

```
. forvalues i = 0.0(0.05)0.15 {
  2. display `i´
  3. }
0
.05
.1
```

The user evidently expects `display` of 0, .05, .1, and .15 in turn, but the loop ends without displaying .15. Why is that? First, let us fix the loop by looping over integers and doing the noninteger arithmetic inside the loop. That is the most important trick for attacking this kind of problem.


```
. forvalues i = 0/3 {
  2. display `i´ * .05
  3. }
0
.05
.1
.15
```

Why did that work as desired, but not the previous loop? The default format for `display` is hiding from us the approximations that are being used, which are necessary because most multiples of 1/10 cannot be held as exact binary numbers. A format with many more decimal places reveals the problem:

```
. forvalues i = 0/3 {
  2. display %20.18f `i´ * .05
  3. }
0.000000000000000000
0.050000000000000003
0.100000000000000006
0.150000000000000022
```

The result 0.150000000000000022 is a smidgen too far as far as the first loop is concerned. Otherwise put, the loop terminates because Stata's approximation to $0.05 + 0.05 + 0.05$ is a smidgen more than its approximation to 0.15:

```
. display %20.18f 0.05 + 0.05 + 0.05
0.150000000000000022
. display %20.18f 0.15
0.149999999999999994
```

A little more cryptic, but closer to the way that Stata actually works, is a display in hexadecimal:

```
. display %21x 0.05 + 0.05 + 0.05
+1.3333333333334X-003
. display %21x 0.15
+1.3333333333333X-003
```

The difference really is very small, but it is enough to undermine the intention behind the original loop.

Another trick that is sometimes useful to ensure desired results is the formatting of numerical values as desired. Leading zeros are often needed, and for that we just need to insist on an appropriate format:

(*Continued on next page*)

```
. forvalues i = 1/20 {
  2. local j : display %02.0f `i´
  3. display "`j´"
  4. }
01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
```

A subtlety to notice here is the pair of double quotes flagging to `display` that the macro `j` is to be treated as a string. Omitting the double quotes as in `display ‘j’` would cause the formatting to be undone, because the default (numeric) display format would produce a display that started 1, 2, 3, and so forth. The syntax here for producing the local `j` is called an extended macro function and is documented in [P] **macro**.

If all that was desired was the display just seen, then the loop could be simplified to contain a single statement in its body, namely,

```
. display %02.0f `i´
```

However, knowing how to produce another macro with this technique has other benefits. One fairly common example is for cycling over filenames. Smart users know that it is a good idea to use a sequence of filenames such as `data01` through `data20`. Naming this way ensures that files will be listed by operating system commands in logical order; otherwise, the order would be `data1`, `data11`, and so forth. But those smart users then need Stata to reproduce the leading zero in any cycle over files. The loop above could easily be modified to solve that kind of problem by including a command such as

```
. use data`j´
```

Correctness, clarity, and ease of maintenance were also mentioned as advantages of looping over integers. Style preferences enter here, and programmers' experiences vary, but on balance fewer coding errors and clearer code overall seem likely to result from the approach here. Moreover, noninteger steps, such as .05 within the very first example, are rarely handed down from high as the only possibilities. There is a marked advantage to changing just a single constant rather than a series of values from problem to problem.

References

Cox, N. J. 2002. Speaking Stata: How to face lists with fortitude. *Stata Journal* 2: 202–222.

———. 2006. Stata tip 33: Sweet sixteen: Hexadecimal formats and precision problems. *Stata Journal* 6: 282–283.

Gould, W. 2006. Mata Matters: Precision. *Stata Journal* 6: 550–560.

Linhart, J. M. 2008. Mata Matters: Overflow, underflow and the IEEE floating-point format. *Stata Journal* 8: 255–268.

The Stata Journal (2010)
10, Number 1, p. 164

Software Updates

gr0009_1: Speaking Stata: Graphing model diagnostics. N. J. Cox. *Stata Journal* 4: 449–475.

Various updates have been made to programs and documentation. The most important are, first, a new command `rbinplot` for plotting means or medians of residuals by bins; second, new options for smoothing using restricted cubic splines where appropriate, for users of Stata 10 and later; third, documentation of `anova` examples using the new syntax introduced in Stata 11.

gr0021_1: Speaking Stata: Smoothing in various directions. N. J. Cox. *Stata Journal* 5: 574–593.

Various updates have been made to programs and documentation. An option to carry out smoothing using restricted cubic splines has been added to `doublesm` and `diagsm`. References to earlier work have been added to the help for `doublesm` and `polarsm`.

gr41_4: Distribution function plots. N. J. Cox. *Stata Journal* 5: 471; *Stata Journal* 3: 449; *Stata Journal* 3: 211; *Stata Technical Bulletin* 51: 12–16. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 108–112.

A `reverse(ge)` option added to the `distplot` program specifies the plotting of probabilities or frequencies of being greater than or equal to any data value. This allows plotting of such probabilities or frequencies on a logarithmic scale, as all calculated quantities will be positive.

