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, 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.

The Stata Journal is indexed and abstracted by CompuMath Citation Index, Current Contents/Social and Behavioral Sciences, RePEc: Research Papers in Economics, Science Citation Index Expanded (also known as SciSearch), Scopus, and Social Sciences Citation Index.

For more information on the Stata Journal, including information for authors, see the webpage http://www.stata-journal.com
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 listed below include both a printed and an electronic copy unless otherwise mentioned.

<table>
<thead>
<tr>
<th></th>
<th>U.S. and Canada</th>
<th>Elsewhere</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Printed &amp; electronic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-year subscription</td>
<td>$115</td>
<td>$145</td>
</tr>
<tr>
<td>2-year subscription</td>
<td>$210</td>
<td>$270</td>
</tr>
<tr>
<td>3-year subscription</td>
<td>$285</td>
<td>$375</td>
</tr>
<tr>
<td>1-year student subscription</td>
<td>$85</td>
<td>$115</td>
</tr>
<tr>
<td>1-year institutional subscription</td>
<td>$345</td>
<td>$375</td>
</tr>
<tr>
<td>2-year institutional subscription</td>
<td>$625</td>
<td>$685</td>
</tr>
<tr>
<td>3-year institutional subscription</td>
<td>$875</td>
<td>$965</td>
</tr>
<tr>
<td><strong>Electronic only</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-year subscription</td>
<td>$85</td>
<td>$85</td>
</tr>
<tr>
<td>2-year subscription</td>
<td>$155</td>
<td>$155</td>
</tr>
<tr>
<td>3-year subscription</td>
<td>$215</td>
<td>$215</td>
</tr>
<tr>
<td>1-year student subscription</td>
<td>$55</td>
<td>$55</td>
</tr>
</tbody>
</table>

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.

Copyright © 2014 by StataCorp LP

Copyright Statement: The Stata Journal and the contents of the supporting files (programs, datasets, and help files) are copyright © by StataCorp LP. The contents of the supporting files (programs, datasets, and help files) may be copied or reproduced by any means whatsoever, in whole or in part, as long as any copy or reproduction includes attribution to both (1) the author and (2) the Stata Journal.

The articles appearing in the Stata Journal may be copied or reproduced as printed copies, in whole or in part, as long as any copy or reproduction includes attribution to both (1) the author and (2) the Stata Journal.

Written permission must be obtained from StataCorp if you wish to make electronic copies of the insertions. This precludes placing electronic copies of the Stata Journal, in whole or in part, on publicly accessible websites, file servers, or other locations where the copy may be accessed by anyone other than the subscriber.

Users of any of the software, ideas, data, or other materials published in the Stata Journal or the supporting files understand that such use is made without warranty of any kind, by either the Stata Journal, the author, or StataCorp. In particular, there is no warranty of fitness of purpose or merchantability, nor for special, incidental, or consequential damages such as loss of profits. The purpose of the Stata Journal is to promote free communication among Stata users.

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

The Stata Journal Editors’ Prize 2014: Roger Newson

Plotting regression coefficients and other estimates

Tools for checking calibration of a Cox model in external validation: Approach based on individual event probabilities

Estimation of multiprocess survival models with cmp

dhreg, xtdhreg, and bootdhreg: Commands to implement double-hurdle regression

The chi-squared goodness-of-fit test for count-data models

txttool: Utilities for text analysis in Stata

iop: Estimating ex-ante inequality of opportunity

femlogit—Implementation of the multinomial logit model with fixed effects

Analysis of partially observed clustered data using generalized estimating equations and multiple imputation

Lee (2009) treatment-effect bounds for nonrandom sample selection

General-to-specific modeling in Stata

Robust data-driven inference in the regression-discontinuity design

adjcatlogit, ccrlogit, and ucrlogit: Fitting ordinal logistic regression models

Collecting and organizing Stata graphs

Speaking Stata: Design plots for graphical summary of a response given factors

Notes and Comments

Stata tip 121: Box plots side by side

Software Updates
The Stata Journal Editors’ Prize 2014: Roger Newson

1 Prize announcement

The editors of the Stata Journal are delighted to announce the award of the Editors’ Prize for 2014 to Roger Newson. The aim of the prize is to reward contributions to the Stata community for one or more outstanding articles published in the Journal in the previous three calendar years. For the original announcement of the prize and its precise terms of reference, see Newton and Cox (2012), which is accessible at the following website: http://www.stata-journal.com/sjpdf.html?articlenum=gn0052. The prize recognizes the outstanding article on “Attributable and unattributable risks and fractions and other scenario comparisons” (Newson 2013a) and three other articles by Newson that appeared within the time frame: “From resultssets to resultstablestables in Stata” (Newson 2012a), “Sensible parameters for univariate and multivariate splines” (Newson 2012b), and “Bonferroni and Holm approximations for Šidák and Holland–Copenhaver $q$-values” (Newson 2013b).

Roger Benedict Newson was born in Nottingham in 1955 and grew up there. He gained a bachelor’s degree from the University of Nottingham, a master’s degree from the University of Reading, and a doctoral degree from the University of Sussex. Morphing from a biologist to a statistician, Newson has combined work as a programmer, a statistician, and a lecturer, focusing mostly on biological and medical problems. He is currently a research associate in the Department of Primary Care and Public Health.
at Imperial College London. He has long been a stalwart of the Stata user community, as the author of many frequently downloaded programs, as a frequent speaker at Stata users group meetings (at every London meeting from 2000, in particular), and as a frequent contributor to Statalist. He has been an associate editor of the Stata Journal since 2003 and has organized Stata Users Group meetings in London with Stephen Jenkins in 2009, 2011, and 2013.

Newson’s article (2013a) “Attributable and unattributable risks and fractions and other scenario comparisons” is an excellent article backed up with excellent software. The margins command introduced in Stata 11 and its related commands pcompare and contrast added huge power to users’ ability to interpret their estimation results. Regression coefficients, especially those for nonlinear models such as logit and probit, are often difficult to explain to nonspecialists. Results are much more easily understandable if they are expressed using an interpretable metric, say, changes or differences in the outcome of interest, such as a probability for a binary outcome or a mean for a continuous outcome. The great power of margins and its related commands is also their potential weakness: going beyond very basic options can be difficult for users. The manual entries are long and complicated, too. Newson’s article goes beyond those manual entries.

The article has several notable features. Drawing on his deep understanding of the statistics involved, Newson first provides intuitive background to the problems that need to be solved and how he attacks them: estimation of “scenario means and comparisons”, as he puts it. He explains the relevant methods and formulas and introduces his useful wrapper commands—margpred, marglmean, regpar, punaf, and punafcc—for margins and its related commands. Several worked examples illustrate the analytical tools and commands. One important aspect of the calculations of the estimates is their sampling variability, and here the nonlinearities raise issues as well. A further feature of Newson’s calculations is careful attention to the automation of symmetric confidence intervals for transformed parameters and asymmetric confidence intervals for the various types of statistics, here making deft use of nlcom in the course of normalizing and variance-stabilizing transformations. As in all of Newson’s programs, careful attention is paid to detail. All the standard options and extras that you might expect are present, for example, handling of weights, a portfolio of saved results, posting of results, and calculations for subgroups.

Newson considers postestimation calculations not only for standard regression models but also for case–control or survival-data models. In addition, he shows how his approach may be used to compare outcomes when different models are applied to the same scenario, as when standardizing statistics from different subpopulations to a common distribution (for example, by sex and age).

Newson refers to his work in the article as relevant to “applied scientists, especially in the public health sector” (2013a, page 672). Although his work is directed at that audience and terms such as attributable and unattributable risks and fractions may be unfamiliar outside that disciplinary context, Newson is here being too modest. For example, important special cases of his regpar and punaf commands can produce what
economists would call “average partial effects” and “relative probability ratios”. These are produced with appropriate choices of the \texttt{atzero()} option. In these and other ways, Newson’s work can be interesting and useful beyond his apparent target readership.

We turn now to the other articles cited in recommendation of Newson for the prize. In “From resultssets to resultstables in Stata” (Newson 2012a), Newson focuses on “delivery” aspects, rather than transformation of estimates as in the first article (Newson 2013a). This article takes forward earlier work for which he is deservedly renowned, specifically, Newson (2003a). His earlier programs provide “resultssets”, sets of estimation results that are themselves Stata datasets. The issue addressed is how to convert such output into appropriately formatted tables for users who may be using some flavor of \TeX{} or Microsoft Word. Together, those programs encompass most of the Stata user community.

Newson (2012a) provides and illustrates the \texttt{listtab} package (which supersedes his earlier \texttt{listtex} package). \texttt{listtab} calls several other utility packages that Newson has written and that are available from SSC. \texttt{listtab} inputs a list of variables and outputs them as a table in one of several formats, including \TeX{}, \LaTeX{}, HTML, Microsoft Rich Text Format, or possibly future XML-based formats. This work represents a substantial contribution to, and enhancement of, the medley of output formatting packages available to the Stata user community.

“Sensible parameters for univariate and multivariate splines” (Newson 2012b) returns the emphasis to providing tools that assist end-users’ interpretations of estimation results. Splines of various kinds are commonly used in regression modeling to allow for nonlinear relationships between predictors and outcome, but the estimated parameters are tricky to interpret. Newson describes the update of his \texttt{bspline} package for creating spline variables (originally published in Newson [2000a]) and the associated utility programs \texttt{frencurv} and a new \texttt{flexcurv}. He provides extensive illustration using worked examples. There is also helpful technical discussion of different types of splines, useful for those who wish to make connections between the functionality built into Stata and more complicated specifications.

“Bonferroni and Holm approximations for Šidák and Holland–Copenhaver $q$-values” (Newson 2013b) is a note building on an earlier article on frequentist $q$-values for multiple-test procedures (Newson 2010b), with its associated \texttt{qqvalue} package. Newson identifies a potential problem: there may be issues of precision, especially with $q$-values corresponding to very small $p$-values, and proposes a remedy, which he also discusses in relation to computing $q$-values and discovery sets. This topic is relatively specialized, but the style is all about getting every detail right, thoroughly in the best traditions of Stata.

In summary, we salute Roger Newson for outstanding contributions to the Stata community, based on excellent programs greatly extending the functionality available to users explained in excellent accompanying articles in the \textit{Stata Journal}, all within a laudable emphasis on providing tools delivering interpretable findings to end users. In every case, he takes a focused approach based on sound general statistical principles, and he shows extraordinary care and attention to detail.
As editors, we are indebted to the awardee for biographical material and to a necessarily anonymous nominator for a most helpful appreciation of Newson’s work. Below, we give references to the most notable of Newson’s publications in the *Stata Technical Bulletin* and *Stata Journal*. Additionally, his software updates flagged in the *Stata Journal* can be found in Stata by typing `search newson, author`.

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

## 2 References


Plotting regression coefficients and other estimates

Ben Jann
University of Bern
Bern, Switzerland
jann@soz.unibe.ch

Abstract. Graphical display of regression results has become increasingly popular in presentations and in scientific literature because graphs are often much easier to read than tables. Such plots can be produced in Stata by the \texttt{marginsplot} command (see \cite{Rmarginsplot}). However, while \texttt{marginsplot} is versatile and flexible, it has two major limitations: it can only process results left behind by \texttt{margins} (see \cite{Rmargins}), and it can handle only one set of results at a time. In this article, I introduce a new command called \texttt{coefplot} that overcomes these limitations. It plots results from any estimation command and combines results from several models into one graph. The default behavior of \texttt{coefplot} is to plot markers for coefficients and horizontal spikes for confidence intervals. However, \texttt{coefplot} can also produce other types of graphs. I illustrate the capabilities of \texttt{coefplot} by using a series of examples.

Keywords: gr0059, coefplot, marginsplot, margins, regression plot, coefficients plot, ropeladder plot

1 Introduction

Tabulating regression coefficients has long been the preferred way of communicating results from statistical models. However, researchers now increasingly use graphs to present regression results, for several reasons. On the one hand, interpretation of regression tables can be challenging, especially if there are interaction effects, categorical variables, or nonlinear functional forms. Moreover, in nonlinear models, the original regression coefficients are often not the primary interest of researchers. For example, in logistic regression, the raw coefficients represent effects on log odds. However, most people would be more comfortable with effects expressed on the probability scale. Because probability effects are not constant in such a model, it can be helpful, for example, to plot effect functions. On the other hand, and more fundamentally, researchers have recognized that displaying results in the form of graphs can be much more effective than tabulation, especially in presentations and lectures, but also in written work. This is because the “reexpression of data in pictorial form capitalizes upon one of the most highly developed human information processing capabilities—the ability to recognize, classify, and remember visual patterns” \cite{Lewandowsky1989}.

Tables are well-suited as a look-up source for specific values, but it is difficult to interpret results presented as numbers in tables. Graphs generally do a much better
job of “revealing patterns, trends, and relative quantities” (Jacoby 1997, 7) because graphs translate differences among numbers into spatial distances, thereby emphasizing the main features of the data and abstracting from irrelevant details. Pictorial representations of information also seem to be easier to remember (Lewandowsky and Spence 1989).1

Graphics are present in many scientific fields. Most prominently, graphs are used to depict univariate distributions (for example, histograms), bivariate distributions (for example, scatterplots), or changes over time (line diagrams). They are used to analyze data—for example, to get a quick overview of important features of the data or evaluate assumptions imposed by statistical models—or to present results (Healy and Moody 2014).

One type of presentation plot that has become popular recently, sometimes called a ropeladder plot, displays regression coefficients or other statistics of interest against a common scale, using markers for point estimates and spikes for confidence intervals (for examples, see Kastellec and Leoni [2007], Harrell [2001], Cleveland [1994, 217–220], Cleveland and McGill [1985], Dice and Leraas [1936], Gosset [Student, pseud.] [1927], and Chapin [1924]). It can be very effective to present statistical results in this way because evaluating the position of points along a common scale and judging the length of lines are two of the most powerful perceptional capabilities of humans (Cleveland and McGill 1985). Furthermore, ropeladder plots provide an immediate and accurate impression of the statistical precision of results, much preferred over p-values and significance stars in regression tables.

Unfortunately, creating such graphs in Stata is tedious, hindering their more widespread use (although, see Newson [2003]). The coefficients and variances have to be gathered from the e() returns, the confidence intervals have to be computed, and the results have to be appropriately stored as variables in the dataset. Then, a suitable variable for the category axis must be generated and coefficient labels must be defined. Finally, a complicated graph command has to be issued to plot the coefficients and confidence intervals.

This task has been greatly simplified with the introduction of marginsplot (see [R] marginsplot) in Stata 12. It is now possible to plot coefficients and confidence intervals with just a few lines of code. For example, consider the following linear regression model (see [R] regress):

---

1. For a brief review of the literature on the merits of graphical displays over tabular representations, see Gelman, Pasarica, and Dodhia (2002). For results on graphical perceptions and general principles on designing effective graphics, see the works by Chambers et al. (1983), Lewandowsky and Spence (1989), and Cleveland (1993, 1994). As a rich source of inspiration, consider Tufte (2001) and Wainer (1997).
. sysuse auto
(1978 Automobile Data)
. regress price mpg trunk length turn

Source | SS      df  MS
--------+---------+---------+---------+---------+---------+---------+---------+---------+
Model   | 159570047 4  39892511.8   F(  4,   69) = 5.79   Prob > F = 0.0004
Residual| 475495349 69  6891236.94   R-squared = 0.2513
        |         |         | Adj R-squared = 0.2079
Total   | 635065396 73  8699525.97   Root MSE = 2625.1

price       Coef.  Std. Err.  t    P>|t|      [95% Conf. Interval]
mpg  |  -186.8417   88.17601  -2.12  0.038   -362.748  -10.93533
trunk  |   -12.72642  104.8785  -0.12  0.904    -221.9534  196.5005
length  |    54.55294  35.56248   1.53  0.130   -16.39227 125.4981
turn  |  -200.3248  140.0166  -1.43  0.157  -479.6502  79.00066
_cons |    8009.893   6205.538   1.29  0.201   -4369.817 20389.6

To plot the regression coefficients (which, in this case, are equal to the average marginal effects), we could type

. margins, dydx(*)
(output omitted)
. marginsplot, horizontal xline(0)
>  yscale(reverse) recast(scatter)
(output omitted)

marginsplot is a versatile command that can do so much, especially when plotting predictive margins, the area of application that marginsplot was primarily designed for. However, marginsplot can only process results left behind by margins (see [R] margins), and it has some other limitations.

To overcome these limitations, I wrote a new command called coefplot. On the following pages, I illustrate the scope and usage of coefplot through a series of examples. For a systematic overview of the syntax and options, type help coefplot after you install the command in Stata.

2 Scope of coefplot

c coefplot is a tool to graph results from estimation commands in Stata, comparable to commands such as outreg (Gallup 2012) or estout (Jann 2007) for tables. Some
of `coefplot`’s functionality overlaps with the possibilities offered by `marginsplot`, but `coefplot` goes much further.

- **marginsplot** can only process the results left behind by the `margins` command. `coefplot`, however, can be applied to the results of any estimation command in Stata that posts its results in `e()` (as most estimation commands do, including `margins` if specified with the `post` option) and can even be used to plot results that have been collected manually using the `matrix` commands (see section 7.6).

- **marginsplot** can only process the results from one call to `margins`. As with tables, however, it is often desirable to combine results from several model specifications or estimation techniques into one graph. With `coefplot`, multiple results can be freely combined and arranged in one graph, including the possibility to distribute results across subgraphs.

- **marginsplot** draws confidence intervals for only one confidence level. Given the criticism of a strict interpretation of significance tests and confidence intervals, it seems advisable to display multiple confidence intervals using varying levels. `coefplot` offers such functionality.

- Finally, good graphs need good labels. `coefplot` offers various options to label coefficients, equations, and subgraphs, include labels for groups of estimates, or to insert subheadings to structure the display.

The main purpose of `coefplot` is to plot point estimates of coefficients along with confidence intervals. By default, `coefplot` draws a ropeladder plot using markers for point estimates and spikes for confidence intervals and by arranging the estimates along a categorical axis providing labels for the different coefficients. Depending on context, however, it can also be sensible to draw different types of graphs. For example, one can use bars for point estimates and capped spikes for confidence intervals or display estimates as connected lines along a continuous axis, which is all supported by `coefplot`.

Figure 1 provides a “tour d’horizon” of `coefplot`, illustrating its scope. Graph A displays a standard ropeladder plot containing regression coefficients from two subgroups for two different dependent variables. In sections 3 and 4, I discuss the basic usage of `coefplot` and explain how to create such a plot.

Graph B is a variant of a standard ropeladder plot, in which subgraphs are drawn by the coefficient instead of by models (see section 7.4). Graph B also illustrates the use of group labels (note the label “Subgroup results” below “Domestic” and “Foreign”), as discussed in section 5.

Graph C is an extreme example of using multiple confidence intervals. The graph contains 50 confidence intervals for each coefficient from levels 1% to 99% using varying line widths and color intensities. Of course, it is also possible to include just a few confidence intervals, say, the 99% and 95% confidence intervals, as discussed in section 6.
Figure 1. Examples of graphs produced by coefplot
Graph D is a plot in which the values of estimates are displayed as marker labels, as discussed in section 7.3. Furthermore, graph D also illustrates the automatic wrapping of long labels (see section 5).

Graph E is a bar plot with capped spikes for confidence intervals, a graph type that is appropriate if the estimates to be plotted are proportions (see section 7.2).

Finally, graph F is yet another graph type, suitable for plotting effect contours in which the categorical axis containing coefficient labels has been replaced with a continuous axis. Such graphs can be created by providing plot positions through the \texttt{at()} option, as discussed in section 7.5.

In the remainder of this article, I discuss how to use \texttt{coefplot} to produce rope-ladder plots (sections 3 and 4), introduce the various options for labeling the categorical axis (section 5), and illustrate the use of multiple confidence intervals (section 6). In section 7, I will cover more-advanced topics, such as using the \texttt{recast()} option, adding marker labels, arranging subgraphs by coefficients, using a continuous axis, and plotting results from matrices.

### 3 Plotting a single model

The syntax to produce a plot of the coefficients of one model is

\begin{verbatim}
\texttt{coefplot [ name ] [ , options ]}
\end{verbatim}

where \texttt{name} is the name of a stored model (see \texttt{[R] estimates}), or \texttt{.} or empty string denoting the active model. For details about \texttt{coefplot} syntax, see \texttt{help coefplot}.

For example, to plot point estimates and 95% confidence intervals for the most recent model, type

\begin{verbatim}
. sysuse auto, clear
(1978 Automobile Data)
. regress price mpg trunk length turn
(output omitted)
. coefplot, drop(_cons) xline(0)
\end{verbatim}

Option \texttt{drop(_cons)} was added to remove the constant and \texttt{xline(0)} was used to draw a reference line at 0 so that we can better see which coefficients are significantly different from 0.
**Plotting regression coefficients**

`coefplot` can graph results from almost any estimation command. For example, to plot coefficients from a logit model (see [R] `logit`), type

```
. sysuse auto, clear
(1978 Automobile Data)
. logit foreign mpg trunk length turn
(output omitted)
. coefplot, drop(_cons) xline(0)
> xtitle(Log odds)
```

With logit models, one is often interested in odds ratios instead of the raw coefficients. To plot odds ratios instead of log odds, use the `eform` option, which causes `coefplot` to compute exponents of coefficients and confidence intervals (using endpoint transformation).

```
. coefplot, drop(_cons) xline(1) eform
> xtitle(Odds ratio)
```

Furthermore, if you want to plot average marginal effects instead of log odds or odds ratios, you can apply `margins` (see [R] `margins`).
. sysuse auto, clear  
(1978 Automobile Data)  
. logit foreign mpg trunk length turn  
(output omitted)  
. margins, dydx(*) post  
(output omitted)  
. coefplot, xline(0)  
> xtitle(Average marginal effects)

It is essential to specify the `post` option with `margins` so that it posts its results in `e()`, which is where `coefplot` collects the results to display. If you do not specify the `post` option, then `margins` leaves `e()` unchanged and `coefplot` uses the raw coefficients from the logit model that still reside in `e()`.

4 Plotting multiple models

To include results from several commands in one graph, one can save the results from each command by using `estimates store` (see [R] `estimates`) and then provide the names of the stored estimation sets to `coefplot`. There are three alternatives for including multiple results in the graph. First, one can include models as different plots in the same graph. By “plot”, I mean a set of markers and confidence spikes using the same plot style. Second, one can create separate subgraphs, with each subgraph containing one or more plots. Third, one can append multiple models into the same plot.

4.1 Models as plots

The syntax to include multiple models as separate plots is

\[ \text{coefplot} \{ [()] \text{name}[ , \text{plotopts}] \} \  \{[()] \text{name}, \text{plotopts} \} ... \} \  , \text{globalopts} \]

where `name` is again the name of a stored model, or . or empty string denoting the active model. `plotopts` are options that apply to a single plot. They specify the information to be collected, affect the rendition of the plot, and provide a label for the plot in the legend. `globalopts` are options that apply to the overall graph, such as titles or axis labels, but may also contain any options allowed as plot options to provide defaults for the single plots. For details about `coefplot` syntax, see `help coefplot`. 
A basic example is as follows:

```stata
. sysuse auto, clear
(1978 Automobile Data)
. regress price mpg trunk length turn
    if foreign==0
(output omitted)
. estimates store D
. regress price mpg trunk length turn
    if foreign==1
(output omitted)
. estimates store F
. coefplot D F, drop(_cons) xline(0)
```

To specify separate options for the individual plots, enclose the models and their options in parentheses. For example, to add a label for each plot in the legend, to use alternative plot styles, and to change the marker symbol, type

```stata
. coefplot (D, label(Domestic Cars)
    pstyle(p3))
>       (F, label(Foreign Cars)
>          pstyle(p4)),
>       msymbol($) drop(_cons) xline(0)
```

In the example, `msymbol()` was specified as a global option so that the same symbol is used in both plots. To use different symbols, include an individual `msymbol()` option for each plot.

`coefplot` offsets the plot positions of the coefficients so that the confidence spikes do not overlap. To deactivate the automatic offsets, one can specify the global option `nooffsets`. Alternatively, one can specify custom offsets by using the `offset()` option (if `offset()` is specified for at least one plot, automatic offsets are disabled). The spacing between coefficients is one unit, so usually offsets between $-0.5$ and $0.5$ make sense. For example, to use smaller offsets than the default, type
4.2 Subgraphs

The syntax to create subgraphs is

```
coefplot plotlist[, subgropts] || plotlist, subgropts || ...
```

where `plotlist` is a list of plots as in section 4.1, and `subgropts` are options that apply to a single subgraph. For details about `coefplot` syntax, see `help coefplot`.

An example with one model per subgraph is

```
. sysuse auto, clear
(1978 Automobile Data)
. regress price mpg trunk length turn
> if foreign==0
(output omitted)
. estimates store D
. regress price mpg trunk length turn
> if foreign==1
(output omitted)
. estimates store F
. coefplot D, bylabel(Domestic Cars)
> || F, bylabel(Foreign Cars)
> ||, drop(_cons) xline(0)
```
Plotting regression coefficients

An example with multiple models per subgraph is

```
. regress weight mpg trunk length turn
> if foreign==0
(output omitted)
. estimates store wD
. regress weight mpg trunk length turn
> if foreign==1
(output omitted)
. estimates store wF
. coefplot
> (D, label(Domestic))
> (F, label(Foreign)), bylabel(Price)
> || wD wF, bylabel(Weight)
> ||, drop(_cons) xline(0)
> byopts(xrescale)
```

Option `byopts(xrescale)` was specified so that each subgraph can have its own scale.

In the example above, plot labels for the legend were set within the first subgraph. They could also have been specified within the second subgraph, because plot styles are recycled with each new subgraph and plot options are collected across subgraphs. To prevent recycling of plot styles, add the `norecycle` option, as follows:

```
. coefplot D F, bylabel(A)
> || wD wF, bylabel(B)
> ||, drop(_cons) xline(0)
> norecycle byopts(xrescale)
> legend(rows(1))
```

4.3 Appending models

The syntax to append models within the same plot is

```
coefplot (name[ , modelopts ] \ [ name, modelopts \ ...] [ , plotopts ] [ ...])
```

where `name` is the name of a stored model, or `.`, or empty string denoting the active model, and `modelopts` are options that apply to a single model. For details about `coefplot` syntax, see `help coefplot`. 
For example, to draw a graph comparing bivariate and multivariate effects, type

```
.sysuse auto, clear
(1978 Automobile Data)
.regress price mpg trunk length turn
(output omitted)
.estimates store multivariate
.foreach var in mpg trunk length turn {
2. quietly regress price `var'
3. estimates store `var'
4. }
.coefplot (mpg \ trunk \ length \ turn, > label(bivariate)) (multivariate), > drop(_cons) xline(0)
```

### 4.4 How coefficients and equations are matched

The default for `coefplot` is to use the first (nonzero) equation from each model and match coefficients across models by their names (ignoring equation names). For example, `regress` returns one (unnamed) equation containing the regression coefficients whereas `tobit` (see `[R] tobit`) returns two equations, equation “model” containing the regression coefficients and equation “sigma” containing the standard error of the regression. Hence, the default for `coefplot` is to match the regression coefficients from the two models and ignore equation “sigma” from the tobit model.

```
.sysuse auto, clear
(1978 Automobile Data)
.foreach v of var price mpg trunk length turn {
2. quietly summarize `v'
3. quietly replace `v' = (`v' - r(mean)) / r(sd)
4. }
.regress price mpg trunk length turn
(output omitted)
.estimate store regress
.tobit price mpg trunk length turn, > ll(-.5)
(output omitted)
.estimate store tobit
.coefplot regress tobit, xline(0)
```

To include the second equation from the tobit model, you can add option `keep(*)` (indicating that all equations are to be kept). However, as soon as more than one equation is collected per model, equation names start to matter and coefficients will be matched within equations. Therefore, you may want to assign the equation name...
"model" to the results from `regress` so that the coefficients from the two models are matched into the same equation.

```
. coefplot (regress, asequation(model))
  > (tobit, keep(*:)),
  > xline(0)
```

Alternatively, you could also use `eqrename(_, = model)` to rename equation "_" to "model" or `eqrename(model = _)` to rename equation "model" to "_".

The option `asequation()` can also be applied when you want to assign equations to results from `margins`. In the following example, I show how to plot log odds of a multinomial logit (see [R] `mlogit`) along with average marginal effects:

```
. sysuse auto, clear
(1978 Automobile Data)
. gen mpp = mpg/8
. label variable mpp "Miles per pint"
. mlogit rep78 mpp foreign if rep>=3
(output omitted)
. estimates store mlogit
. forvalues i = 3/5 {
  2. quietly margins, dydx(*)
  > predict(outcome(`i´)) post
  3. estimates store ame`i´
  4. quietly estimates restore mlogit
  5. }
. coefplot mlogit, keep(*) drop(_cons) omitted bylabel(Log Odds)
  > || (ame3, aseq(3) \ ame4, aseq(4) \ ame5, aseq(5)), bylabel(AME)
  > ||, xline(0) byopts(xrescale)
```
Finally, if you want to match coefficients that have different names in the input models, you can apply the `rename()` option. I use the following example to illustrate the effect of measurement error in regression models:

```
. drop _all
. matrix C = ( 1, .5, 0 \\ .5, 1, .3 \\ 0, .3, 1 )
. drawnorm x1 x2 x3, n(10000) corr(C) (obs 10000)
. generate y = 1 + x1 + x2 + x3 +
  > 5 * invnorm(uniform())
. regress y x1 x2 x3
  (output omitted)
. estimates store m1
. generate x1err = x1 +
  > 2 * invnorm(uniform())
. regress y x1err x2 x3
  (output omitted)
. estimates store m2
. coefplot (m1, label(Without error)) (m2, label(With error)),
  > xline(1) rename(x1err = x1)
```

We can see how measurement error on `x1` distorts all slope coefficients in the model, even for variable `x3` that is uncorrelated with `x1` (due to the indirect correlation through `x2`).

### 4.5 How coefficients are ordered

In general, coefficients are plotted in the same order (from top to bottom) as they appear in the input models. However, coefficients appearing only in later models are placed after coefficients from earlier models (with the exception of `_cons`, which is always placed last). To arrange the coefficients in a different order, you can use the `order()` option, as in the following example:
Plotting regression coefficients

. sysuse auto, clear
(1978 Automobile Data)
. regress price mpg length
(output omitted)
. estimate store m1
. regress price mpg trunk turn
(output omitted)
. estimate store m2
. regress price mpg trunk length turn
(output omitted)
. estimate store m3
. coefplot m1 || m2 || m3, xline(0)
> drop(_cons) byopts(row(1)) order(mpg trunk length)

Mileage (mpg)
−400 −200 0 200 400
Trunk space (cu. ft.)
−200 −100 0 100 200
Length (in.)
−400 −200 0 200 400
Turn Circle (ft.)
−400 −200 0 200 400

m1 m2 m3
Within order( ), you can use the * (any string) and ? (any nonzero character) wildcards. Furthermore, you can type . to insert gaps (but also see the section on headings and groups below).

In case of multiple equation models, the default is to order coefficients by equations. To reorder equations, to apply different orderings within equations, or to break equations apart, specify equation names within order( ), as in the following example:

. sysuse auto, clear
(1978 Automobile Data)
. mlogit rep78 headroom gear_ratio foreign
> if rep>3
(output omitted)
. coefplot, xline(0) keep(*:)
> order(4:foreign 5:foreign
> 4:gear* head* 5:gear* head*
> 4: 5:)

5 Labeling the categorical axis

coefplot looks for variables that correspond to the collected coefficient names and then uses their variable labels for the categorical axis. For factor variables, coefplot additionally takes value labels into account (the rule is to print the value label, if a value label is defined, and otherwise print the variable label or name along with the level). The following is an example with categorical variables and interaction terms:
To use coefficient names instead of variable labels, specify the `nolabels` option.

### 5.1 Custom coefficient labels

An easy way to provide labels for the coefficients is to define appropriate variable and value labels before applying `coefplot`; see [D] `label`. However, not all coefficients have corresponding variables (for example, `_cons`). To provide labels for such coefficients or to assign custom labels to coefficients without manipulating variable labels, use the `coeflabels()` option.
Plotting regression coefficients

`coeflabels()` has a `wrap()` and a `truncate()` suboption to deal with long labels. These suboptions apply to all coefficient labels, whether they are automatically generated or provided within `coeflabels()`. For example, to limit the line to 20 characters and wrap long labels to multiple lines, type

```
    . coefplot, xline(0) coeflabel(, wrap(20))
```

Multiline labels can also be created using compound double quotes, for example, `coeflabels(1.foreign = "Line 1" "Line 2")`. Such labels will not be altered by `wrap()` or `truncate()`.

5.2 Headings and groups

Sometimes it is useful to add headings between coefficients to better arrange a graph. This can be achieved by using the `headings()` option.

```
    . sysuse auto, clear
    (1978 Automobile Data)
    . keep if rep78>=3
    (10 observations deleted)
    . regress mpg headroom i.rep##i.foreign
       (output omitted)
    . coefplot, xline(0) omitted baselevels
    >     headings(1)
    >     3.rep78 = "(bf:Repair Record)"
    >     0.foreign = "(bf:Car Type)"
    >     3.rep78#0.foreign =
    >     "(bf:Interaction Effects)"
    >     drop(_cons)
```

In this example, `omit` requests to plot omitted coefficients and `baselevels` requests to plot base-level coefficients. Omitted coefficients and base-level coefficients are always equal to 0, but it can sometimes be helpful to include them in a graph for clarity. The `{bf}` tag changes text to bold; see [G-4] `text` for details on text in graphs.
In addition to headings, you can also define groups of coefficients and add group labels using the `groups()` option as follows:

```
> . coefplot, xline(0) omitted base
>     groups(.?rep78 =
>         ""{bf:Repair}" "{bf:Record}"")
>     ?.foreign = "{bf:Car Type}"
>     ?.rep78#?.foreign =
>         "{bf:Interaction Effects}"
>     drop(_cons)
```

5.3 Equation labels

Equation labels provide yet another layer of labels. The default is to place the equation labels on the right-hand side, similar to group labels.

```
> . sysuse auto, clear
(1978 Automobile Data)
>     gen mpp = mpg/8
>     mlogit rep78 mpp i.foreign if rep>=3
(Warning omitted)
>     coefplot, omitted keep(:*)
>         coeflabels(mpp = "Mileage")
>         eqlabels("Equation 1" "Equation 2"
>                 "Equation 3")
```
However, you can also set the equation labels as headings between equations by using the `asheadings` suboption as follows:

```
    . coefplot, omitted keep(\*)
    >     coeflabels(mpp = "Mileage")
    >     eqlabels("(bf:Equation 1)"
    >           "(bf:Equation 2)"
    >           "(bf:Equation 3)", asheadings)
```

In this case, the `headings()` option is not allowed.

### 5.4 Labels on opposite side

The default is to plot all labels on the left of the plot region. Use option `yscale(alt)` to move labels to the right (see [G-3] `twoway_options`).

```
    . sysuse auto, clear
    (1978 Automobile Data)
    . keep if rep78>=3
    (10 observations deleted)
    . regress mpg headroom i.rep##i.foreign
    (output omitted)
    . coefplot, xline(0) omitted baselevels
    >     headings(
    >           3.rep78 = "(bf:Repair Record)"
    >           0.foreign = "(bf:Car Type)"
    >           3.rep78#0.foreign = "(bf:Interaction Effects)"
    >     )
    >     drop(_cons) yscale(alt)
```
Group labels and equation labels are rendered as additional axes (axis 2 for group labels; axis 2 or 3 for equation labels, depending on whether groups were specified), so you have to use the `axis()` suboption to move these.

```
. coefplot, xline(0) omitted base
>    groups(?.
```

Moving group labels to the right can also be useful if you want to add an extra set of coefficient labels without actually forming groups. The following is an example in which `groups()` is used to add information on the sample sizes of factor levels:

```
. sysuse auto, clear
(1978 Automobile Data)
. keep if rep78>=3
(10 observations deleted)
. regress mpg i.rep i.foreign
(output omitted)
. coefplot, xline(0) omitted baselevels
>    groups(3.rep78 = "N = 30"
>             4.rep78 = "N = 18"
>             5.rep78 = "N = 11"
>             0.foreign = "N = 38"
>             1.foreign = "N = 21",
>             nogap angle(horizontal))
>    drop(_cons) yscale(alt axis(2))
```

6 Confidence intervals

The default for `coefplot` is to draw spikes for 95% confidence intervals (or as set by `set level`; see `[R] level`). To specify a different level or to include multiple confidence intervals, use the `levels()` option. Here is an example with 99.9%, 99%, and 95% confidence intervals:
```
. sysuse auto, clear
   (1978 Automobile Data)
. regress price mpg trunk length turn
   (output omitted)
. coefplot, drop(_cons) xline(0)
   > msymbol(s) mfcolor(white)
   > levels(99.9 99 95)
   > legend(order(1 "99.9" 2 "99" 3 "95")
   > row(1))
```

Line widths are (logarithmically) increased across the confidence intervals. To use different line widths, specify, for example, `ciopts(lwidth(*1 *2 *4))`.

To compute confidence intervals, `coefplot` collects the variances of the coefficients from the diagonal of `e(V)` and then, depending on whether degrees of freedom is available in scalar `e(df_r)` (or, for estimates from `[MI] intro`, in matrix `e(df_mi)`), applies the standard formulas for confidence intervals on the basis of the t distribution or the normal distribution, respectively. If a model does not provide degrees of freedom but you want to compute confidence intervals by using the t distribution, you can provide the degrees of freedom through option `df()` (see the online help). If variances are stored in a matrix other than `e(V)`, use the `v()` option to provide the appropriate matrix name, or use option `se()` to provide custom standard errors (in which case variances from `e(V)` will be ignored). Likewise, if your estimation command provides precomputed confidence intervals, use the `ci()` option to include them in the plot. For example, to plot the normal-approximation, percentile, and bias-corrected confidence intervals that are provided in `e(ci_normal)`, `e(ci_percentile)`, and `e(ci_bc)` by the bootstrap method, you could type

```
. regress price mpg trunk length turn,
   > vce(bootstrap)
   (output omitted)
. coefplot
   > ci(ci_normal) label(normal)
   > ci(ci_percentile) label(percentile)
   > ci(ci_bc) label(bc)
   > drop(_cons) xline(0) legend(row(1))
```
In addition to `levels()` and `ci()`, you can also use option `cismooth` to add smoothed confidence intervals.\(^2\) By default, `cismooth` generates confidence intervals for 50 equally spaced levels \(1, 3, \ldots, 99\) with graduated color intensities and varying line widths, as illustrated in the following example:

```
. coefplot, drop(_cons) xline(0)
>   cismooth grid(none)
```

The smoothed confidence intervals are produced independently from `levels()` and `ci()` and are not affected by `ciopts()`. Their appearance, however, can be set by several suboptions (see the online help). If `cismooth` is specified together with `levels()` or `ci()`, then the smoothed confidence intervals are placed behind the confidence intervals from `levels()` or `ci()`.

### 7 Alternate plot types and advanced examples

#### 7.1 Vertical mode

By default, `coefplot` produces a horizontal graph with labels on the \(y\) axis and values on the \(x\) axis. To flip axes, specify the `vertical` option.

---

\(^2\) The `cismooth` option has been inspired by code by David B. Sparks to produce smoothed confidence interval plots in R (see [http://dsparks.wordpress.com/2011/02/21/choropleth-tutorial-and-regression-coefficient-plots/](http://dsparks.wordpress.com/2011/02/21/choropleth-tutorial-and-regression-coefficient-plots/)).
Plotting regression coefficients

```
. sysuse auto, clear
(1978 Automobile Data)
. regress price mpg trunk length turn
   (output omitted)
. coefplot, drop(_cons) vertical yline(0)
```

When changing from horizontal to vertical mode, options referring to specific axes must be adjusted. This is why `yline(0)` was used in the example instead of `xline(0)` to draw the 0 line.

### 7.2 Using the recast() option

To change the plot types used for markers and confidence intervals, you can use the `recast()` option. Available plot types for markers are standard twoway plots such as `scatter` (the default), `line`, `dot`, or `bar`. For confidence intervals, use range plots such as `rspike` (the default), `rline`, `rcap`, or `rbar`.

For example, to display confidence intervals using capped spikes, you could type

```
. sysuse auto, clear
(1978 Automobile Data)
. regress price mpg trunk length turn
   (output omitted)
. coefplot, drop(_cons) xline(0)
. ciopts(recast(rcap))
```

```
```

```
```
Furthermore, a bar chart of proportions with capped confidence spikes can be produced as follows:

```stata
. sysuse auto, clear
(1978 Automobile Data)
. proportion rep if foreign==0
(output omitted)
. estimates store domestic
. proportion rep if foreign==1
(output omitted)
. estimates store foreign
. coefplot domestic foreign,
> vertical recast(bar)
> barwidth(0.25) fcolor(*.5)
> ciopts(recast(rcap)) citop
> citype(logit)
> xtitle(Repair Record 1978) ytitle(Proportion)
```

In this example, the `citop` option was used to prevent the lower limits of the confidence intervals from being hidden behind the bars. Furthermore, the `citype(logit)` option was specified to compute confidence intervals using the logit transformation, as is appropriate for proportions (see [R] proportion).

### 7.3 Adding marker labels

To add the values of the coefficients as marker labels, use the `mlabel` option, possibly together with `format()` to set the display format.

```stata
. sysuse auto, clear
(1978 Automobile Data)
. keep if rep78>=3
(10 observations deleted)
. regress mpg headroom i.rep##i.foreign
(output omitted)
. coefplot, xline(0) mlabel format(%9.2g)
> mlabposition(12) mlabgap(*2)
```

Stata graphs do not support background colors for marker labels, which makes labels unreadable if you place them on top of the markers using `mlabposition(0)`. However, the following is a workaround:

```stata
. mlabposition(12) mlabgap(*2)
```
Plotting regression coefficients

```
. sysuse auto, clear
    (1978 Automobile Data)
. keep if rep78>3
    (10 observations deleted)
. regress mpg headroom i.rep##i.foreign
    (output omitted)
. mata: st_matrix("e(box)",
>     (st_matrix("e(b)") :- 2 \
>      st_matrix("e(b)") :+ 2))
. coefplot, xline(0) mlabel format(%9.2g)
>     mlabposition(0) msymbol(i)
>     ci(95 box) ciopts(recast(. rbar)
>     barwidth(. 0.35) fcolor(. white)
>     lwidth(. medium))

The trick is to add a second “confidence interval” that is a bar of fixed width. The
dot in each suboption within ciopts() specifies the “default” style (see [G-4] stylelists).

7.4 Arranging subgraphs by coefficients

It is often sensible to arrange coefficients in separate subgraphs with individual scales,
because the size of coefficients may vary considerably. For example, when comparing
results by subgroups or estimation techniques, the focus is usually more on differences
across models and less on differences within models, so it appears natural to use individ-
ual subgraphs for the different coefficients (see Gelman, Pasarica, and Dodhia [2002]).

Creating subgraphs by coefficients requires lengthy commands, because a separate
piece of subgraph syntax must be put together for each coefficient. To circumvent the
extra typing, you can use the bycoefs option. Technically, bycoefs flips coefficients
and subgraphs, that is, the coefficients are treated as subgraphs and what was specified
as subgraphs is treated as coefficients. This seems difficult to understand, but it should
come clear in the following example:
```

```
. sysuse auto, clear
    (1978 Automobile Data)
. forv i = 3/5 {
    2. quietly regress price mpg
>           headroom weight turn
>           if rep78==`i`
    3. estimate store rep78_`i`
4. }
. coefplot rep78_3 || rep78_4 || rep78_5,
>     drop(_cons) xline(0)
>     bycoefs byopts(xrescale)
```
If `bycoefs` is specified, options such as `headings()` and `groups()` apply to the elements on the categorical axis (instead of coefficients). To address the elements, use integer numbers 1, 2, 3, etc., as in the following example:

```stata
.sysuse auto, clear
(1978 Automobile Data)
.regress price mpg headroom weight turn
(output omitted)
.estimates store Total
.regress price mpg headroom weight turn
> if foreign==0
(output omitted)
.estimates store Domestic
.regress price mpg headroom weight turn
> if foreign==1
(output omitted)
.estimates store Foreign
.coefplot Domestic || Foreign || Total, drop(_cons) yline(0) vertical
> bycoefs byopts(yrescale)
> group(1 2 = "Subgroup results", nogap) ylabel(0, add)
```

Option `ylabel(0, add)` was added to ensure that the 0 baseline is included in each subgraph.

### 7.5 Using a continuous axis

Coefficients provided to `coefplot` may represent estimates along a continuous dimension. Examples are predictive margins or marginal effects computed over values of a continuous variable. In such a case, one can use the `at()` option to provide the plot positions to `coefplot`. Here is an example where predictive margins of `foreign` are computed by level of `mpg`, once from a bivariate model and once from a multivariate model:
. sysuse auto, clear
   (1978 Automobile Data)
. logit foreign mpg
   (output omitted)
. margins, at(mpg=(10(2)40)) post
   (output omitted)
. estimates store bivariate
. logit foreign mpg turn price
   (output omitted)
. margins, at(mpg=(10(2)40)) post
   (output omitted)
. estimates store multivariate
. coefplot bivariate multivariate, at
>   ytitle(Pr(foreign=1)) xtitle(Miles per Gallon)

`at()` causes `coefplot` to use a continuous axis with default labeling for the plotted
estimates instead of compiling a categorical axis. It also causes `coefplot` to switch to
vertical mode, because this is more common for such plots. Because no categorical axis
is constructed if `at()` is specified, options such as `order()`, `coeflabels()`, `headings()`,
and `groups()` are not allowed. Furthermore, continuous and categorical mode cannot
be mixed. That is, `at()` must be specified for all models or for none. In the example
above, `at` was used without argument. This is suitable for results provided by `margins`,
because `coefplot` contains special code to retrieve the plot positions in this case. See
the online help for alternative applications of `at()`.

`coefplot` does not change the plot type for markers and confidence intervals, and
hence still draws dots and spikes. Use option `recast()` to change this, for example, as
follows:

. coefplot (bivariate) (multivariate), at
   >   ytitle(Pr(foreign=1))
   >   xtitle(Miles per Gallon)
   >   recast(line) lwidt(*2)
   >   ciopts(recast(rline))
7.6 Plotting results from matrices

Finally, to plot results from a matrix (see [P] matrix) instead of the e() returns, use syntax

\[
\text{coefplot} \left( \text{matrix}(m\text{spec}), \text{modelopts} \right) [\ldots][][\ldots]
\]

where \text{mspec} is

- \text{name} \quad \text{get point estimates from first row of matrix \text{name}}
- \text{name[\#,\#]} \quad \text{get point estimates from row \# of matrix \text{name}}
- \text{name[\#,\#]} \quad \text{get point estimates from column \# of matrix \text{name}}

For details about coefplot syntax, see help coefplot.

In this case, names given in options such as \text{at()} or \text{ci()} will also be interpreted as matrix names. For example, to plot medians and their confidence intervals as computed by \text{centile} (see [R] centile), you could type

```
. sysuse auto, clear
(1978 Automobile Data)
. matrix res = J(3,3,.)
. matrix coln res = median ll95 ul95
. matrix rown res = mpg trunk turn
. local i 0
. foreach v of var mpg trunk turn {
2. local ++ i
3. quietly centile `v'
4. matrix res[`i',1] = r(c_1),
    > r(lb_1), r(ub_1)
5. }
. matrix list res
```

```
<table>
<thead>
<tr>
<th></th>
<th>median</th>
<th>ll95</th>
<th>ul95</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpg</td>
<td>20</td>
<td>19</td>
<td>22</td>
</tr>
<tr>
<td>trunk</td>
<td>14</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>turn</td>
<td>40</td>
<td>37.078729</td>
<td>42</td>
</tr>
</tbody>
</table>
```

\[
\text{coefplot matrix(res[.,1]), ci((res[.,2] res[.,3]))}
\]
A single `coefplot` command can contain both regular syntax and `matrix()` syntax. For example, to add means to the graph above, you could proceed as follows:

```
    . mean mpg trunk turn
    (output omitted)
    . estimates store mean
    . coefplot (matrix(res[,1]), label(median)
    >       ci((res[,2] res[,3])))
    >       (mean)
```

8 Acknowledgments

I thank Nick Cox, Marc Höglinger, and an anonymous reviewer for their helpful suggestions.

9 References


About the author

Ben Jann is professor of sociology at the University of Bern, Switzerland. His research interests include social-science methodology, statistics, social stratification, and labor market sociology. Recent publications include articles in *Sociological Methodology*, *Sociological Methods and Research*, *Public Opinion Quarterly*, and the American Sociological Review.
Tools for checking calibration of a Cox model in external validation: Approach based on individual event probabilities

Patrick Royston
MRC Clinical Trials Unit at University College London
London, UK
j.royston@ucl.ac.uk

Abstract. The Cox proportional hazards model has been used extensively in medicine over the last 40 years. A popular application is to develop a multivariable prediction model, often a prognostic model to predict the clinical outcome of patients with a particular disorder from “baseline” factors measured at some initial time point. For such a model to be useful in practice, it must be “validated”; that is, it must perform satisfactorily in an external sample of patients independent of the sample on which the model was originally developed. One key aspect of performance is calibration, which is the accuracy of prediction, particularly of survival (or equivalently, failure or event) probabilities at any time after the time origin. We believe systematic evaluation of the calibration of a Cox model has been largely ignored in the literature. In this article, we suggest an approach to assessing calibration using individual event probabilities estimated at different time points. We exemplify the method by detailed analysis of two datasets in the disease primary biliary cirrhosis; the datasets comprise a derivation and a validation dataset. We describe a new command, \texttt{stcoxcal}, that performs the necessary calculations. Results for \texttt{stcoxcal} can be displayed graphically, which makes it easier for users to picture calibration (or lack thereof) according to follow-up time.

Keywords: st0357, stcoxcal, Cox proportional hazards model, multivariable model, prognostic factors, external validation, calibration, survival probabilities

1 Introduction

Multivariable survival models are used in medicine, particularly as the basis of prognostic models in clinical and research practice and as risk models for population screening. Given certain “baseline” factors measured at some appropriate initial time point (denoted by $t = 0$), the models are used to predict the future clinical outcome of individuals with, for example, a particular condition such as cancer or heart disease or those at risk of such. Risk models have several applications, including selection of persons at high risk of needing preventive therapy, stratification of risk in clinical disease and audit studies, and personalized prediction of disease outcome.
Models developed for patients in a given sample may predict well within that dataset but fail to “generalize” (predict) well on samples from other patient populations. Assessing generalizability is the key component of external model validation. In statistical terms, external validation involves checking that outcome predictions from a model developed on a “derivation” sample are sufficiently accurate in an independent “validation” sample. See Altman and Royston (2000) and Altman et al. (2009) for a general background on model validation.

It is common to distinguish two aspects when “validating” a model: discrimination and calibration. Discrimination means the ability of a model to distinguish between outcomes of patients with different risks. Calibration means the accuracy of prediction, particularly of survival (or equivalently, failure or event) probabilities at any time after \( t = 0 \). The literature has paid much attention to measures of discrimination; popular examples include the work of Harrell et al. (1982) on the \( c \) index of concordance and that of Royston and Sauerbrei (2004) on the \( D \) statistic. (See Choodari-Oskooei, Royston, and Parmar [2012] for a detailed comparison between several approaches.) In contrast, little has been published on assessing calibration of models for time-to-event data; Harrell (2001) describes techniques based on the bootstrap for internal validation. Here we focus on tools for assessing calibration in external data.

We suggest an approach to assessing calibration of a Cox proportional hazards model using individual event probabilities at different time points. This may be seen as an extension of recent work (Royston and Altman 2013; Royston Forthcoming) addressing calibration in aggregates of patients at similar risk of an event (that is, in risk groups). The tools may also be used to check the calibration of a model on the same data it was developed on, that is, on the derivation dataset. See Royston and Altman (2013) for further considerations in validating a published Cox model.

The structure of this article is as follows. We first outline the framework for our survival modeling. We describe two datasets from the disease primary biliary cirrhosis (PBC) as well as the Cox model we are using as a running example. We discuss our motivation and approach to calibration in a more familiar logistic regression context, and we extend it to the Cox proportional hazards model framework. We present illustrative analyses using the PBC data. We then explain \texttt{stcoxcal}, a new tool that implements the analyses and graphs. We finish with some closing comments.

## 2 Proportional hazards models

Suppose we have a vector of explanatory variables \( \mathbf{x} = (x_1, \ldots, x_k) \). A Cox proportional hazards model with the parameter vector \( \beta \) incorporates multiplicative effects of \( \mathbf{x} \) on the baseline hazard function and is usually written as

\[
h(t; \mathbf{x}) = h_0(t) \exp(\mathbf{x}^\top \beta) \]   \hspace{1cm} (1)

where \( h(t; \mathbf{x}) \) is the hazard function, \( h_0(t) = h(t; \mathbf{0}) \) is the baseline hazard function, and \( t \) is the follow-up time. If we integrate (1), we obtain the cumulative hazard function, \( H(t; \mathbf{x}) = \int_0^t h(u; \mathbf{x}) du \). Taking logarithms, we get
\[
\ln H(t; \mathbf{x}) = \ln H_0(t) + \mathbf{x} \beta
\]  
(2)

Now let’s write \( \ln H(t; \mathbf{x}) = g\{S(t; \mathbf{x})\} \), where \( S(t; \mathbf{x}) \) is the survival function and \( g(u) = \ln(-\ln u) \) is the “link function”. Then we can write (2) as
\[
g\{S(t; \mathbf{x})\} = g\{S_0(t)\} + \mathbf{x} \beta
\]  
(3)

In the Cox model (1), the baseline hazard function and, hence, the baseline survival function in (3), \( S_0(t) = S(t; \mathbf{0}) \), are unspecified and are not estimated as part of the model.

3 Example datasets

3.1 Description and proportional hazards model

To illustrate, we assemble prognostic variables in common across two datasets relating to the disease PBC (usually known as cirrhosis of the liver). The first dataset on which we derived a model was used by Fleming and Harrington (1991) to exemplify certain aspects of survival analysis. The data comprise survival or censoring times of \( n = 418 \) patients (161 deaths) with PBC, 312 of whom entered a randomized controlled trial and the remaining 126 participated in a cohort study. Six prognostic factors with complete data were available for analysis.

The second dataset, which we used as a validation sample, comes from a randomized controlled trial of 248 patients with PBC (Christensen et al. 1985). After removing 41 cases (17%) with missing values or no patient follow-up, we had data on 207 patients (105 deaths) for analysis.

Three covariates were recorded in both datasets: age, bilirubin, and albumin. We applied \texttt{mfp} with Cox regression (the \texttt{stcox} command) to build the following proportional hazards model in the derivation dataset:

\[
h(t; \mathbf{x}) = h_0(t) \exp \{0.04085 \times \text{age} + 0.9405 \times \ln \text{(bilirubin)} - 0.09852 \times \text{albumin}\}
\]

The predictive ability is high for a survival model: Harrell’s \( c = 0.824 \) and Royston and Sauerbrei’s \( D = 2.27 \), for which \( R^2_D = 55\% \). Corresponding values in the validation dataset when applying the prognostic index (PI) \( \mathbf{x} \hat{\beta} \) from the derivation dataset are Harrell’s \( c = 0.785 \) and Royston and Sauerbrei’s \( D = 1.89 \), for which \( R^2_D = 46\% \). There appears to be some reduction in the discrimination of the model in the validation dataset.

The two datasets were combined for analysis. A binary variable \texttt{val} was created, taking the value zero in the derivation dataset and one in the validation dataset.
3.2 Preliminary analysis

Figure 1a shows Kaplan–Meier survival curves in the derivation and validation datasets. Survival (unadjusted for covariates) is clearly worse in the validation dataset. Figure 1b shows estimates of the baseline survival curve in each dataset, which are computed by centering the \( \text{PI} \) on its mean in the derivation dataset and offsetting the \( \text{PI} \) in Cox models separately in each dataset. Although adjusting the \( \text{PI} \) clearly brought the curves closer together (and, incidentally, altered their shape), the lower survival in the validation dataset persists. Figure 1b suggests that the model is imperfectly calibrated in the validation dataset, with a tendency to underpredict event probabilities. By fitting the following Cox model to the combined dataset,

\[
\text{. stcox val xb}
\]

we find that the adjusted hazard ratio for \( \text{val} \) is 1.38 [95% confidence interval (CI) 1.08, 1.77], which shows an increased adjusted hazard for \( \text{val} = 1 \), as expected. The unadjusted hazard ratio for \( \text{val} \) is 1.70. Next we investigate the calibration in more detail.

Figure 1. PBC datasets. a) Kaplan–Meier survival curves; b) baseline survival curves according to the \( \text{PI} \) centered on 0 in the derivation dataset.
Tools for checking calibration of a Cox model in external validation

We clarify the Stata-related details for obtaining the curves in figure 1b as follows. The three covariates are \( x_1 \) (age), \( x_2 \) (log bilirubin), and \( x_3 \) (albumin).

\[
\begin{align*}
&\text{stcox } x_1 \ x_2 \ x_3 \text{ if } \text{val}=0 \\
&\text{predict } xb, xb \quad \text{// note: predicts for all observations, including val=1} \\
&\text{summarize } xb \text{ if } \text{val}=0 \\
&\text{replace } xb = xb - r(\text{mean}) \\
&\text{stcox if } \text{val}=0, \text{ offset}(xb) \\
&\text{predict } s_0, \text{basesurv} \\
&\text{stcox if } \text{val}=1, \text{ offset}(xb) \\
&\text{predict } s_1, \text{basesurv} \\
&\text{line } s_0 \ s_1 \ _t, \text{ sort}
\end{align*}
\]

We centered \( xb \) to ensure that the baseline distribution function is meaningful. Thus \( xb = 0 \) represents a patient in the derivation dataset at “average risk” of dying.

4 Assessing calibration of logistic regression models

Let \( F(t; x) = 1 - S(t; x) \) be the failure (event) probability, that is, the chance of an event occurring in the interval \((0, t)\) for an individual with covariate vector \( x \). To motivate what follows, we first consider a logistic regression model. Now \( t \) plays no role, so the event probability, \( F(x) \), is a function of only the \( \Pi \), \( x\bar{\beta} \), and the baseline log odds of an event, \( \beta_0 = \logit\{F(0)\} \). Assessing model calibration means comparing the observed event probabilities with those predicted by the model. The observed event probability for an individual is taken as 1 if the individual experiences an event (outcome \( Y = 1 \)) and 0 otherwise (outcome \( Y = 0 \)). We write the \( \Pi \) as \( \Pi = \hat{\beta}_0 + x\hat{\beta} \). The predicted event probability is \( \hat{F}(x) = \logit^{-1}(\Pi) = (1 + \exp(-\Pi))^{-1} \). An auxiliary logistic regression model, which is linear in the \( \Pi \) (Miller, Hui, and Tierney 1991), may be used to check agreement between observed and predicted probabilities.

\[
\logit\{\Pr(Y = 1)\} = \gamma_0 + \gamma_1 \Pi
\]

If model (4) is fit to the same dataset as that used to estimate \( \hat{\beta}_0 \) and \( \hat{\beta} \), the estimates of \( \gamma_0 \) and \( \gamma_1 \) are identically 0 and 1, respectively, which is of no help. However, (4) may be used to investigate external validation when \( \hat{\beta}_0 \) and \( \hat{\beta} \) are estimates from a published report or other suitable source.

Consider the simplified auxiliary model

\[
\logit\{\Pr(Y = 1)\} = \gamma_0 + \Pi
\]

that is, with \( \gamma_1 \) constrained to 1, (4) with the \( \Pi \) offset from the linear predictor. The intercept \( \gamma_0 \) in (5) assesses calibration “in the large” (Harrell 2001) because it shifts the entire distribution function \( F(Y) \) by \( \gamma_0 \) on the logit scale.

We can quantify miscalibration easily by applying three hypothesis tests based on (4) and (5). For calibration in the large, we fit (5), estimate \( \gamma_0 \), and test \( \gamma_0 = 0 \). To check the regression on the \( \Pi \) (essentially, discrimination), we fit (4) and test \( \gamma_1 = 1 \). To perform an overall test of calibration, we use a joint test of \((\gamma_0, \gamma_1) = (0, 1)\) with 2
degrees of freedom. If we are concerned about type 1 error, a conservative approach is to perform the joint test first, then proceed to the separate tests of $\gamma_0$ and $\gamma_1$ only if the result of the joint test is significant. This is a closed-test procedure that maintains the familywise error rate.

Before we return to survival models, we note that calibration error in the validation dataset may be a more complex function of the PI than a straight line on the logistic scale. Thus (4) and (5) may be misspecified. A recommended graphical adjunct is to plot a scatterplot smooth of $Y$ or of residuals $Y - \hat{F}(x)$ on $\hat{F}(x)$, together with pointwise CIs. This can reveal subtle miscalibration. It can also be used as a graphical check of calibration on the derivation dataset.

5 Assessing calibration of Cox regression models

5.1 The baseline distribution function

The principle of checking calibration by comparing observed and predicted event probabilities can also be used with Cox models. Calibration in the survival context is intrinsically time dependent. It may be assessed overall and at several suitable values of $t$ up to the maximum event time.

This raises an important issue for the Cox model. In external validation, we wish to evaluate predicted event probabilities, $\hat{F}(t; x)$ for some $t$, in an independent dataset. To do this, we need to “export” the baseline distribution function, $F_0(t) = 1 - S(t; 0)$, for relevant values of $t$ from the derivation data to the validation data. However, the Cox model does not provide an estimate of the baseline distribution function. A simple solution to this is to model the baseline distribution function in the derivation dataset using a suitable class of approximating models. Often an adequate solution is to model the log baseline cumulative-hazard function, $\ln H_0(t) = \ln(-\ln S_0(t))$, as a second-degree fractional polynomial (FP2) in $t$ (Royston and Altman 2013; Royston Forthcoming).

Despite the assumptions of linear regression analysis not being met, the function to be approximated is very smooth, and it is satisfactory to estimate the parameters of the FP2 model $E[\ln H_0(t)] = \delta_0 + \delta_1 t^{p_1} + \delta_2 t^{p_2}$ by ordinary least squares. $p_1$ and $p_2$ may be estimated using the \texttt{fracpoly} or (in Stata 13.0 and above) the \texttt{fp} command. See Royston and Altman (1997) for further examples of the usefulness of FP functions for approximation of smooth functions.

The ordinary least-squares regression comprises two stages. First, after fitting the Cox model to the derivation dataset using \texttt{stcox}, we use the command \texttt{predict} \texttt{varname}, \texttt{basechazard} to estimate the baseline log cumulative-hazard function in the derivation dataset. We then regress \texttt{varname} on $t$ as previously described. The resulting FP2 function can be used to predict $\ln H_0(t)$ out of sample in the validation dataset. The out-of-sample prediction step cannot readily be done without the intermediate regression analysis.
5.2 Overall calibration

In principle, we can investigate the calibration for external validation by using Cox regression on the PI in the validation sample. As with logistic regression, we are interested in the parameters $\gamma_0$ and $\gamma_1$ in this global setting. However, the Cox model has no intercept, so we cannot estimate $\gamma_0$. Furthermore, by regressing on the PI in the validation dataset, we are reestimating the baseline distribution function. For a strict assessment of calibration, we wish to avoid such reestimation. We want to know whether the entire model (baseline included) fit on the derivation dataset still predicts accurately in the validation dataset.

We take a different approach to obtain a calibration model with a linear predictor of the form $\gamma_0 + \gamma_1 \text{PI}$. Instead of the PI, the covariate in the model for the $i$th patient at time $t$ is the estimated log cumulative-hazard function, $\ln \hat{H}(t; x_i) = \ln(-\ln\{1 - \hat{F}(t; x_i)\})$, which is the complementary log-log transformation of the predicted event probability, $\hat{F}(t; x_i)$. We obtain the predicted event probability by “importing” the baseline distribution function with an FP2 function estimated in the derivation data and applied out of sample to the validation data, as previously described. We have

$$\hat{F}(t; x_i) = 1 - \hat{S}_0(t)^{\exp(\text{PL})}$$

where

$$\ln \{-\ln \hat{S}_0(t)\} = \hat{\gamma}_0 + \hat{\gamma}_1 \text{PI} + \hat{\gamma}_2 \text{PL}$$

We thus have “expected” event probabilities at time $t$ at the individual level. How do we derive corresponding “observed” event probabilities? Pohar Perme and Andersen (2008) and Andersen and Pohar Perme (2010) help with this. Given a sample of $n$ individuals and a time point $t$ within the observed follow-up times, the method of pseudo-observations (which we call “pseudovalues”) provides values $\hat{F}_1(t), \ldots, \hat{F}_n(t)$, which are unbiased estimates of $F_1(t), \ldots, F_n(t)$. Right-censoring is taken into account. Parner and Andersen (2010) elegantly implemented the method in Stata as the stpsurv command. Note that the values $\hat{F}_1(t), \ldots, \hat{F}_n(t)$ are jackknife quantities and individually do not resemble recognizable event probabilities. For example, they are not necessarily confined to $(0, 1)$ and may even be negative or exceed 1. Their key property is their unbiasedness in expectation.

For any reasonable value of $t$, the Cox model is perfectly calibrated on the validation dataset if the following property holds:

$$E\{\hat{F}_1(t)\} = F(t; x_i)$$

Under these conditions, a generalized linear model (GLM) with responses $\hat{F}_1(t), \ldots, \hat{F}_n(t)$, linear predictor $\gamma_0 + \gamma_1 \ln \hat{H}(t; x_i)$, and complementary log-log link function $\ln \{-\ln (1 - x)\}$ should fit the validation data well.
5.3 Testing regimen: Single time point

The GLM with responses $\tilde{F}_i(t)$ and linear predictor $\gamma_0 + \gamma_1 \ln \tilde{H}(t; x_i)$ supports three tests at time $t$:

1. Intercept test. Constraining $\gamma_1 = 1$, we wish to know whether $\hat{\gamma}_0$ is consistent with 0. If it is not consistent with 0, we have a calibration error sometimes known as “miscalibration in the large” (Harrell 2001). The (adjusted) event rate in the validation data differs from that in the derivation data.

2. Slope test. Test of $\gamma_1 = 1$ with $\gamma_0$ estimated. For a correctly calibrated model, the estimate $\hat{\gamma}_1$ should be consistent with 1. If the test $p$-value is significant, the calibration failed. Because $\gamma_1 = 0$ implies a complete lack of model discrimination, the case $\hat{\gamma}_1 < 1$ may also indicate reduced discrimination in the validation dataset.

3. Joint test. The GLM can also furnish a joint test of $(\gamma_0, \gamma_1) = (0, 1)$ with 2 degrees of freedom. This examines the overall evidence for (linear) miscalibration.

One approach yielding a closed-test procedure is to perform the joint test (test 3) first and, if it is significant, to proceed to the intercept and slope tests. The last two tests should provide more information on the nature of the miscalibration.

5.4 Example

We illustrate the three tests for investigating calibration in the PBC validation dataset. We chose the time point $t = 7$ years. Table 1 shows Wald tests from the GLM reported by the stcoxcal command.

<table>
<thead>
<tr>
<th>Test</th>
<th>Wald $\chi^2$</th>
<th>Degrees of freedom</th>
<th>$p$-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Intercept</td>
<td>6.42</td>
<td>1</td>
<td>0.011</td>
</tr>
<tr>
<td>2. Slope</td>
<td>0.03</td>
<td>1</td>
<td>0.85</td>
</tr>
<tr>
<td>3. Joint</td>
<td>6.81</td>
<td>2</td>
<td>0.033</td>
</tr>
</tbody>
</table>

The estimate of the intercept is $\hat{\gamma}_0 = 0.44$ (standard error 0.17), suggesting that the mortality rate (relative hazard) at 7 years is a factor of about $\exp(0.44) = 1.55$ higher in the validation dataset. There is no evidence that $\gamma_1 \neq 1$ ($p = 0.85$). The joint test with 2 degrees of freedom provides some evidence ($p = 0.033$) that $(\gamma_0, \gamma_1) \neq (0, 1)$. Because $\hat{\gamma}_1$ is consistent with 1, the miscalibration seems to be entirely in the large. Notice that the joint test is less significant and has lower power in this situation. There is almost no contribution from $\gamma_1 \neq 1$ toward the 2 degrees of freedom $\chi^2$ statistic.
Tools for checking calibration of a Cox model in external validation

The `stcoxcal` command and its output with the results given in table 1 and with the various parameter estimates from the GLMs are shown below.

```
use pbc, clear
(PBC data, 3 sources)
istcox x1 x2 x3 if !val
(output omitted)
predict xb, xb
istcoxcal xb, val(val) times(7) test
(Std. Err. adjusted for 207 clusters in _id)
| _f  | Semirobust Coef. | Std. Err. | z  | P>|z|   | [95% Conf. Interval] |
|-----|------------------|-----------|----|-------|---------------------|
| 1._times | .4416587 | .174341 | 2.53 | 0.011 | .0999568 .7833607 |
| _clogF | .174341 | (offset) |

[Test 1: intercepts (gamma0) = 0 with slope (gamma1) constrained to 1]
1._times = 0
chi2( 1) = 6.42
Prob > chi2 = 0.0113
(Std. Err. adjusted for 207 clusters in _id)

| _f  | Semirobust Coef. | Std. Err. | z  | P>|z|   | [95% Conf. Interval] |
|-----|------------------|-----------|----|-------|---------------------|
| _clogF | .9585466 | .2217061 | 4.32 | 0.000 | .5240107 1.393083 |
| 1._times | .4192495 | .2173986 | 1.93 | 0.054 | -.0068439 .8453429 |

[Test 2: slope (gamma1) = 1 with constants (gamma0) estimated]
1._times = 0
_chi2( 1) = 0.03
Prob > _chi2 = 0.8517

[Test 3: joint test of slope (gamma1) = 1 and all constants (gamma0) = 0]
1._times = 0
(_clogF = 1
chi2( 2) = 6.81
Prob > _chi2 = 0.0333

The graph provided by `stcoxcal` is shown in figure 2. The dashed line shows the (rather jagged) running-line smooth of the pseudovalues at $t = 7$ years, estimated by `running`. The smoothed event probabilities are higher than predicted at all values of the predicted probability, demonstrating miscalibration in the large.
Figure 2. Smoothed pseudovalues (dashed lines) with pointwise 95% CI plotted against predicted event probabilities for the PBC data at $t = 7$ years. The solid line is the line of identity, denoting perfect calibration. Some miscalibration in the large is evident, with underprediction of event probabilities in the validation dataset.

5.5 Testing regimen: Multiple time points

In practice, we do not wish to limit calibration assessment to one time point. Rather, we want to assess it at several time points $t_1, \ldots, t_m$ spanning the follow-up period. We use `stpm2` to estimate the $m$ pseudovalues for every patient. The calibration analysis can now provide $m$ models $\gamma_{0j} + \gamma_{1j} \ln \hat{H}(t_j; \mathbf{x})$ ($j = 1, \ldots, m$), with 1 linear predictor for each time point.

Such models may be fit for all $m$ time points simultaneously after reshaping the data to “long” format structured according to the $m$ replicates. The `glm` command with `link(cloglog)` is used to fit models to the $mn$ pseudovalues $\hat{F}_i(t_j)$ on $\ln \hat{H}(t_j; \mathbf{x}_i)$ ($i = 1, \ldots, n; j = 1, \ldots, m$). The sandwich estimator is used for variance estimation of regression coefficients (Andersen and Pohar Perme 2010).

We do not want to fit such a complex model with as many as $m$ linear predictors, so we must simplify it. We start by investigating miscalibration in the large. We assume distinct intercept parameters $\gamma_{01}, \gamma_{02}, \ldots, \gamma_{0m}$ and a single slope $\gamma_1$, and we constrain $\gamma_1 = 1$. We implement this by using `glm` with $t$ as a factor variable and by applying the `offset()` option. Test 1, the test for zero intercepts, tests $\gamma_{01} = \gamma_{02} = \cdots = \gamma_{0m} = 0$ using a Wald test with $m$ degrees of freedom. Test 2, the slope test, is similar to the $m = 1$ case: we assume $\gamma_{11} = \gamma_{12} = \cdots = \gamma_{1m}$ and again include $t$ as a factor variable. We test whether the average value of $\gamma_1$ equals 1. Test 3, the joint test, is again performed with $\gamma_{01}, \gamma_{02}, \ldots, \gamma_{0m}$ and $\gamma_1$ fitted, and it has $m + 1$ degrees of freedom.
Tools for checking calibration of a Cox model in external validation

We can expand the model by testing for an interaction between \( \ln \hat{H}(t_j; \mathbf{x}) \) and the \( m \) time points. This allows us to investigate whether \( \gamma_{11} = \gamma_{12} = \cdots = \gamma_{1m} \), that is, whether \( \gamma_1 \) changes over time. A statistically significant test of the interaction suggests that miscalibration varies. For example, a Cox model in data with long-term follow-up could predict accurately in early follow-up but fail later by losing discrimination (reduction in \( \gamma_1 \)) or by changing miscalibration in the large (change in \( \gamma_0 \)).

The test of interaction has \( m - 1 \) degrees of freedom, with the null hypothesis being that \( \gamma_{11} = \gamma_{12} = \cdots = \gamma_{1m} \). This is calculated using the user-written command \texttt{stcoxcal} with the \texttt{test} option described in section 6.

5.6 Example

We extend the example with \( t_1 = 7 \) years by considering yearly intervals up to 9 years; that is, \( t_j = j \) (\( j = 1, \ldots, 9 \)). The results of the four calibration tests are shown in table 2.

Table 2. Tests of calibration in the PBC dataset over nine years of follow-up

<table>
<thead>
<tr>
<th>Test</th>
<th>Wald ( \chi^2 )</th>
<th>Degrees of freedom</th>
<th>( p )-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Intercept</td>
<td>16.49</td>
<td>9</td>
<td>0.057</td>
</tr>
<tr>
<td>2. Slope</td>
<td>0.42</td>
<td>1</td>
<td>0.52</td>
</tr>
<tr>
<td>3. Joint</td>
<td>18.00</td>
<td>10</td>
<td>0.055</td>
</tr>
<tr>
<td>4. Interaction</td>
<td>8.13</td>
<td>8</td>
<td>0.42</td>
</tr>
</tbody>
</table>

Test 1, the test of intercepts, shows some evidence against all intercepts being 0, but it is borderline (\( p = 0.057 \)). Test 2, the slope test of \( \gamma_1 = 1 \), is nonsignificant (\( p = 0.52 \)). Considered over all 9 time points, there is no evidence that \( \gamma_1 \neq 1 \). A borderline result is also obtained for the joint test. The interaction test does not suggest that \( \gamma_1 \) varies over time.

5.7 Plotting smoothed pseudovalues across time

Under perfect calibration, the vector \( \hat{\mathbf{F}}(t; \mathbf{x}) \) should accurately describe the expected pseudovalues across patients at any appropriate time \( t \). Because miscalibration may be time dependent, plotting a combined graph with the chosen values of \( t \) may help to clarify the pattern of miscalibration over time.
The results for the PBC data are shown in figure 3.

![Figure 3](image)

Figure 3. Smoothed pseudovalues (dashed lines) with pointwise 95% CI plotted against predicted event probabilities for the PBC data. Times represent each year over nine years of follow-up. The solid line is the line of identity, denoting perfect calibration.

The underestimation of event probabilities in the validation dataset is visible at most of the nine time points.

For a more detailed view, it is sometimes helpful to smooth and plot the residuals, that is, the pseudovalues minus the predicted event probabilities. Under perfect calibration, there should be no important biases, trends, or patterns among the smoothed residuals. The corresponding plot requires the residuals option of stcoxcal. Figure 4 shows this type of plot for the PBC data. Essentially, the same message emerges as from figure 3.
Tools for checking calibration of a Cox model in external validation

Figure 4. Smoothed pseudo-value residuals (dashed lines) with pointwise 95% CI plotted against predicted event probabilities for the PBC data. The horizontal line at $y = 0$ denotes perfect calibration. The plot uses the residuals option of stcoxcal.

6 Implementation

The methods, calculations, and graphs described above are implemented with the command stcoxcal.

6.1 Syntax

```
  stcoxcal xbetavar [if] [in], times(numlist) [nograph residuals
    saving(filename) test trend val(varname) graph two-way_options
    running_options]
```

You must have running and stpsurv installed before using stcoxcal. running is available from the Statistical Software Components archive (see [R] ssc). stpsurv can be installed from the Stata Journal archive with the command `net install st0202, from(http://www.stata-journal.com/software/sj10-3)`.
6.2 Description

stcoxcal is a tool for examining the (possibly time-dependent) calibration of a Cox model whose linear predictor (“prognostic index”) is supplied in xbeta.

A “well-calibrated” model is one that accurately predicts survival or event probabilities at all relevant follow-up times. A model that includes covariates whose effects change (for example, dwindle) over time is unlikely to be well calibrated. Such a model will give a more or less biased prediction of survival probabilities. stcoxcal is designed to detect and display the lack of calibration graphically. It also includes tests of good calibration and of time-dependent trends of miscalibration.

By default, stcoxcal examines calibration of a model on its “own” dataset. With the val() option, stcoxcal can be used to examine model calibration in an independent dataset (that is, for external validation).

6.3 Options

times(numlist) lists times at which model calibration is to be assessed. times() is required.

nograph suppresses the production of calibration plots.

residuals plots the smoothed residuals (difference between observed and predicted event probabilities) against the predicted event probabilities. The default is to plot smoothed observed against predicted event probabilities.

saving(filename) saves five variables in the validation dataset to file filename:

_id observation number in the original data
_times integer scores (levels) 1, 2, ... of times specified in times()
_f pseudovalues for event probabilities
_F event probabilities predicted from a Cox model
_clogF complementary log-log transformation of _F

These variables can be used by an expert to create plots and to further analyze model calibration. The data are held in long format, with a complete set of values for each level of _times.

test tests whether the slope (on the log cumulative-hazard scale) of the regression of pseudovalues for event probabilities on predicted event probabilities over all time points in times() equals one. A nonsignificant p-value suggests good overall calibration, sometimes called “calibration in the large”. test also tests the interaction between the slopes and the times specified in times(). A significant p-value suggests that calibration changes over time. Typically, calibration declines as follow-up time increases.

trend tests whether the slope (on the log cumulative-hazard scale) of the regression of pseudovalues for event probabilities on predicted event probabilities over all time
Tools for checking calibration of a Cox model in external validation

points in \texttt{times()} equals one (same as for \texttt{test}). \texttt{trend} also tests the linear interaction between the slopes and the integer scores for the times specified in \texttt{times()}. This may be more powerful than the interaction test provided by \texttt{test}.

\texttt{val(varname)} is for use in external validation. \texttt{varname} is a binary variable coded zero to define the “model derivation” dataset and any other nonmissing value to define the “model validation” dataset. Predictions of event probabilities at different times from the derivation dataset are made in the validation dataset via the linear predictor and a smoothed version of the baseline cumulative-hazard function in the derivation dataset. Royston and Altman (2013) call this “strict” calibration.

\texttt{graph twoway_options} are options of \texttt{graph twoway}. These may be used to customize the appearance of the calibration plots.

\texttt{running_options} are options of \texttt{running}. These may be used to customize the smoothing of pseudovalues. The most relevant option is likely to be \texttt{span(#)}. See \texttt{help running} for further information.

6.4 Remarks

Note that \texttt{stcoxcal} computes the baseline survival and cumulative hazard functions internally. As a preliminary, \texttt{stcoxcal} centers the prognostic index supplied in \texttt{xbetavar} on zero. If \texttt{val(varname)} is provided, the mean of \texttt{xbetavar} in the subset defined by \texttt{varname} = 0 is subtracted from all values of \texttt{xbetavar}. Otherwise, centering takes place over the estimation sample. Next, a Cox model is fit with no covariates and with \texttt{xbetavar} offset from the linear predictor. Again, this is done either in the \texttt{varname} = 0 subset or in the estimation sample. Finally, the baseline cumulative hazard function is predicted and smoothed for use with the calibration method described in section 5.

Because \texttt{xbetavar} or indeed the original covariates are not refitted to the validation data, \texttt{stcoxcal} can be used in “partial validation” mode. The prognostic index is created from a derivation model fit elsewhere and imported for application in the available validation dataset. Validation is partial because the baseline cumulative hazard and survival functions are estimated by \texttt{stcoxcal} on the validation data, whereas \texttt{xbetavar} is calculated by the user on the validation data from regression coefficients estimated externally. Although imperfect, partial validation nevertheless allows a useful evaluation of the predictive accuracy of a predefined model when the baseline distribution function is (perforce) tailored to the validation data.
6.5 Examples

```stata
webuse brcancer, clear
stset rectime, failure(censrec) scale(365.24)
fp generate x1^(-2 -0.5)
fp generate x6^(0.5), scale
stcox x1_1 x1_2 x4a x4b x5e x6_1 hormon
predict xb, xb
stcoxcal xb, times(1(1)6) test
stcoxcal xb, times(1(1)6) trend
set seed 3143
generate byte random_half = (runiform() < 0.5)
stcox x1_1 x1_2 x4a x4b x5e x6_1 hormon if random_half==0
predict xb2, xb
stcoxcal xb2, val(random_half) times(1(1)6) test
stcox x1 x4a x4b x5 x6 hormon
predict xb3, xb
stcoxcal xb3, times(1(1)6) test
```

6.6 Stored results

`stcoxcal` stores the following in `r()`: 

Scalars

- `r(gamma1)`: estimate of $\gamma_1$ with $\gamma_0$ estimate
- `r(gamma1_se)`: Std. Err. of `gamma1`
- `r(P0)`: p-value for test 1, of $\gamma_0 = 0$ given $\gamma_1 = 1$
- `r(P1)`: p-value for test 2, of $\gamma_1 = 1$ with $\gamma_0$ estimated
- `r(P01)`: p-value for test 3, joint test of $(\gamma_0, \gamma_1) = (0, 1)$
- `r(Pint)`: p-value for test 4, of interaction of $\gamma_1$ with time

Macros

- `r(fp_pwrs)`: powers of $t$ in FP2 model for $\ln H_0(t)$

7 Comments

Currently, `stcoxcal` works only with “plain” Cox models, that is, Cox models without stratification factors (the `strata()` option in `stcox`) and without time-dependent regression coefficients (`tvc()` not allowed) or time-varying covariates.

The user can choose the time points at which to assess calibration. Results will somewhat vary with different choices, so it may be advisable to perform a sensitivity analysis. For example, when one uses the $m = 5$ time points of 2, 4, 6, 8, and 10 years with the PBC data, test 1 is significant at $p = 0.040$ and test 3 at $p = 0.045$. This result should change the interpretation only slightly. The estimates of $\gamma_j$ ($j = 1, \ldots, 5$) with $\gamma_1 = 1$ are 0.32, 0.06, 0.28, 0.40, and 0.24, again suggesting some miscalibration in the large.

If a formula for the baseline survival function calculated on the derivation data were provided, `stcoxcal` could be slightly extended to handle validation of a published model even without the raw derivation data. This would require investigators proposing a Cox
Tools for checking calibration of a Cox model in external validation

model to publish an FP-based expression for the baseline log cumulative-hazard function. Unfortunately, in practice, the baseline survival function or a suitable transformation of it is never reported, so implementing the extension is not currently worthwhile. Nevertheless, as described in section 6.4, partial validation of an externally derived prognostic index is an option in such cases.

Calibration of Cox proportional hazards models has been substantially neglected in the literature. We hope that our suggestions as well as the new command stcoxcal for assessing calibration of these models will help analysts with external validation of time-to-event models.

8 References


### About the author

Patrick Royston is a medical statistician with more than 30 years of experience, with a strong interest in biostatistical methods and in statistical computing and algorithms. He works largely in methodological issues in the design and analysis of clinical trials and observational studies. He is currently focusing on alternative outcome measures in trials with a time-to-event outcome; on problems of model building and validation with survival data, including prognostic factor studies and treatment-covariate interactions; on parametric modeling of survival data; and on novel clinical trial designs.
Estimation of multiprocess survival models with 
cmp

Tamás Bartus  
Corvinus University of Budapest  
Hungarian Demographic Research Institute  
Budapest, Hungary  
tamas.bartus@uni-corvinus.hu

David Roodman  
Freelance Public Policy Consultant  
Washington, DC  
david@davidroodman.com

Abstract. Multilevel multiprocess hazard models are routinely used by demographers to control for endogeneity and selection effects. These models consist of multilevel proportional hazards equations, and possibly probit equations, with correlated random effects. Although Stata currently lacks a specialized command for fitting systems of multilevel proportional hazards models, systems of seemingly unrelated lognormal survival models can be fit with the user-written cmp command (Roodman 2011, Stata Journal 11: 159–206). In this article, we describe multiprocess survival models and demonstrate theoretical and practical aspects of estimation. We also illustrate the application of the cmp command using examples related to demographic research. The examples use a dataset shipped with the statistical software aML.

Keywords: st0358, survival analysis, multilevel analysis, multilevel multiprocess hazard model, simultaneous equations, SUR estimation, cmp

1 Introduction

Multilevel multiprocess hazard models are routinely used by demographers to adjust regression estimates for endogeneity and selection effects. Originally, multilevel multiprocess models were developed as systems of proportional hazards models with correlated individual-level random effects (Lillard 1993). The multilevel multiequation modeling framework also accommodates the joint estimation of hazard and probit equations to account for the endogeneity of dummy explanatory variables that appear in the hazard equation of primary interest (Lillard, Brien, and Waite 1995; Impicciatore and Billari 2012). The joint estimation accounts for the correlation of the random effects and allows researchers to control for the effects of unobserved personality traits.

To date, no official Stata commands are devoted to estimating systems of survival models. Therefore, it is not surprising that multiprocess hazard models are fit using other statistical packages, including aML (Lillard and Panis 2003) and MLwiN (Rasbash et al. [2012]; see also Leckie and Charlton [2013]). With the recently introduced gsem command, Stata can now fit systems of survival models with correlated random effects. In this article, we demonstrate how Stata users can fit multiprocess models with the user-written cmp command (Roodman 2011). cmp is a flexible tool to estimate systems of equations with various link functions and with normally distributed
and correlated errors. Although most of the bivariate models fit in the article could also be fit with \texttt{gsem}, \texttt{cmp} offers two advantages over \texttt{gsem} for survival modeling. First, \texttt{cmp} is structured to allow cross-equation correlations in modeling errors, even when equations have probit, interval, or other kinds of censoring. Doing the same with \texttt{gsem} is cumbersome because it requires a user to create latent variables, impose constraints upon them, and mathematically transform the results for intuitive interpretation. Second, \texttt{cmp} naturally deals with the truncated outcomes, which is necessary when using multispell survival data.

In this article, we explain how \texttt{cmp} can be used to fit systems of lognormal survival models or systems that may also include probit models. We also show how recent additions to the \texttt{cmp} package enable researchers to fit systems of multilevel models with random effects. This article is organized as follows. In section 2, we describe multiprocess hazard models, as developed by Lillard (1993). In section 3, we describe the \texttt{cmp} compatible models that we label multiprocess survival models. We give the syntax for estimation in section 4. We then present examples in section 5 using a dataset shipped with \texttt{aML}, and we conclude in section 6.

2 Multiprocess hazard models

2.1 Motivation

Multiprocess modeling was motivated by the insight that explanatory variables are often endogenous because of selection mechanisms. Suppose a researcher examines the impact of children on marital stability. Estimates of ordinary survival models of the hazard of divorce are likely to be biased because of the presence of two forms of endogeneity (Lillard and Waite 1993). First, having children is the outcome of a process of timing of births. Second, the latter process may depend on the latent propensity to end the marriage: if couples expect their marriage to be short-lived, they may postpone the first (or higher-order) births. Therefore, the latent expectation of marriage dissolution creates a spurious negative relationship between the number of children and the hazard of marital dissolution. If children negatively affect the hazard of separation, the relationship between children and marital dissolution might be both positive and negative.

The classic method of eliminating the endogeneity bias is to estimate systems of equations with joint normally distributed disturbances (Heckman 1978). Let $y_1^*$ and $y_2^*$ denote the endogenous latent variables under study; for instance, the former might denote the hazard of conception, and the latter might denote the hazard of marital dissolution. The dependence of each latent variable on the other, as well as on other explanatory variables, is described with the following structural equations

\begin{align*}
y_1^* &= \alpha_1'X_1 + \lambda_1 y_2^* + \epsilon_1 \\
y_2^* &= \alpha_2'X_2 + \lambda_2 y_1^* + \epsilon_2
\end{align*}
where \( \mathbf{X}_1 \) and \( \mathbf{X}_2 \) are vectors of observed variables, and \( \mathbf{\alpha}_1 \) and \( \mathbf{\alpha}_2 \) are vectors of coefficients. Observed realizations of the latent variables, like marital status or the number of children, may be included in the explanatory variables.

Endogeneity arises from the presence of latent variables on the right-hand side of the structural equations. Hence, one should estimate the system of reduced-form equations. Estimation must consider that the residuals in the reduced-form equations are probably correlated, even when the disturbances in the structural equations are independent of each other. If the latter error terms were normally distributed, the following system, supplemented with appropriate link functions, must be estimated:

\[
\begin{align*}
\mathbf{y}^*_1 &= \beta_{11}' \mathbf{X}_1 + \beta_{12}' \mathbf{X}_2 + v_1 \\
\mathbf{y}^*_2 &= \beta_{21}' \mathbf{X}_1 + \beta_{22}' \mathbf{X}_2 + v_2 \\
\begin{bmatrix}
v_1 \\
v_2
\end{bmatrix} &\sim \mathcal{N} \left( \mathbf{0}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\
\sigma_{12} & \sigma_2^2 \end{bmatrix} \right)
\end{align*}
\]

The estimation of the variance–covariance matrix of the residuals also makes it easier to interpret the results and separate the causal and selection effects (Heckman 1978). The reduced-form coefficients depend on the corresponding structural parameters and the selection parameters \( \lambda_1 \) and \( \lambda_2 \). The estimation of the covariance matrix of the reduced-form residuals enables one to estimate the selection effect, because the elements of this matrix depend on the selection parameters and not on the structural parameters. One can then use the available estimates of the selection effects to identify the structural parameters. For instance, suppose that \( \lambda_2 \) were constrained to be 0 and the variable \( x_j \) appears in both \( \mathbf{X}_1 \) and \( \mathbf{X}_2 \). Here \( \beta_{21} = \alpha_2 \) and \( \beta_{22} = 0 \), which implies that \( \beta_{2j} = \alpha_{2j} \). Hence, the reduced-form coefficient of \( x_j \) in the first equation, \( \beta_{1j} = \alpha_{1j} + \lambda_1 \beta_{2j} \), and the estimated covariance of the residuals is \( \lambda_1 \sigma_2^2 \). Therefore, the structural coefficient can be recovered as

\[
\alpha_{1j} = \beta_{1j} - \lambda \beta_{2j} = \beta_{1j} - \frac{\sigma_{12}}{\sigma_2^2} \beta_{2j}
\]  

### 2.2 Multilevel multiprocess hazard models

The estimation strategy outlined above cannot be applied to survival analysis without further modifications or extensions. In the popular proportional hazards models, the log of the hazard rate equals the linear combination of variables and coefficients. This model can be restated as a latent-variable model in which the random component of the latent variable follows an exponential distribution. Without a widely accepted multivariate exponential distribution, the correlation of the underlying residuals cannot be modeled, and the seemingly unrelated estimation strategy seems to be infeasible.

To solve this problem, Lillard (1993) suggested the joint estimation of hazard models including normally distributed and possibly correlated random effects. Including jointly normally distributed random effects allows one to adjust estimates for the correlation of the total underlying residuals, and it allows one to estimate the covariance matrix of residuals and, hence, the selection effects. Note that the assumption of joint normality applies only to the random effects. Note also that identifying the random effects requires
repeated occurrences of outcomes. This data requirement is fortunately easy to meet because demographic events such as marriage, divorce, and giving birth are recurrent.

The resulting model, often labeled as a multilevel multiprocess model, can be stated as follows

\[ y_{1j}^* = \beta_1^1 X_{1j} + u_1 + v_{1j} \]
\[ y_{2j}^* = \beta_2^2 X_{2j} + u_2 + v_{2j} \]
\[ \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \sigma_{1,1}^2 & \sigma_{1,2}^2 \\ \sigma_{2,1}^2 & \sigma_{2,2}^2 \end{bmatrix}\right) \]

where \( j \) indexes the recurrent observations, and for simplicity, \( X_{1j} \) and \( X_{2j} \) encompass all explanatory variables in the equations. The subscript \( j \) expresses that the latent outcomes as well as the explanatory variables may change over time, in general, and over spells, in particular. All variances and covariances are indexed by 2, indicating that the matrix expresses these covariances among level-2 residuals.

The model must be supplemented with appropriate link functions to reflect the incomplete observability of \( y_{1j}^* \) and \( y_{2j}^* \). In addition to the link function of proportional hazards models, other link functions can be accommodated. For instance, the multilevel hazard model of marital dissolution, which includes premarital cohabitation as an explanatory variable, may be fit jointly with a probit model of premarital cohabitation to control for a possible selection effect that arises because individuals who are less willing or less able to live in marriages may choose cohabitation or are at an above-average risk of divorce upon marriage (Lillard, Brien, and Waite 1995).

### 2.3 Fitting multilevel multiprocess models with gsem

To date, no official Stata command is explicitly devoted to estimating systems of hazard equations with correlated random effects. The official \texttt{gsem} command, however, could potentially fit such models. Because the focus of our article is not \texttt{gsem}, we just outline the procedure. The starting point is the equivalence of exponential hazard models and Poisson models (Holford 1980): if survival time \( t \) follows an exponential distribution with parameter \( h \), then the expected number of failures follows a Poisson distribution with parameter \( ht \). Relying on this result, Skrondal and Rabe-Hesketh (2004) show that exponential hazard models can be restated as Poisson regression models, in which the dependent variable is the failure-indicator variable and the explanatory variables include the natural log of the duration of the current spell. These results suggest that a user can use \texttt{gsem} to estimate (2) when the user specifies the Poisson family. The user should also include correlated latent variables in the equations to represent the correlation of the random effects (the \( u \)'s) across the equations.
3 Multiprocess modeling with cmp

3.1 The multiprocess lognormal survival model

In this article, we argue that the user-written \texttt{cmp} command (Roodman 2011) allows one to fit systems of lognormal survival models with jointly normally distributed error terms. The \texttt{cmp} command supports interval regression models, even ones with truncated dependent variables. Note that the lognormal survival model is just an interval-censored regression of log failure-times. Note also that the lognormal model is formulated exclusively in the accelerated failure-time metric, so it cannot be formulated in the proportional hazards metric (Cleves et al. 2010). In short, multiprocess modeling boils down to fitting lognormal survival models jointly with other lognormal survival models or with probit models for endogenous regressors.

The seemingly unrelated system of log failure-times for \( P \) interdependent survival processes (that is, the multiprocess lognormal survival model) can be defined as

\[
\begin{align*}
\log T_1 &= \beta_1' X_{1j} + \varepsilon_{1j} \\
\vdots \\
\log T_P &= \beta_P' X_{Pj} + \varepsilon_{Pj}
\end{align*}
\]

\[\varepsilon \sim \mathcal{N}\left(0, \begin{bmatrix} \sigma_1^2 & \ldots & \sigma_1^P \\ \vdots & \ddots & \vdots \\ \sigma_P^1 & \ldots & \sigma_P^2 \end{bmatrix}\right)\quad (3)
\]

Subscript \( j \) expresses that the explanatory variables and the residuals change over time and over the observation periods indexed by \( j \). The process-specific equations are seemingly unrelated because the process-specific error terms can be correlated.

Unlike the classic multilevel multiprocess models, the current model does not include individual-specific random effects. Once the process-specific errors are assumed to be normal, they can easily be modeled as intercorrelated (the normal distribution easily generalizes to multiple dimensions). Therefore, it also becomes less essential to introduce individual-level random effects when we want cross-equation correlation. Nevertheless, we can add higher-level residuals to the structural equations, which are also assumed to be jointly normally distributed.
\[
\log T_{1j} = \beta'_1 X_{1j} + u_1 + \varepsilon_{1j} \\
\ldots \\
\log T_{Pj} = \beta'_p X_{Pj} + u_P + \varepsilon_{Pj} \\
\varepsilon \sim \mathcal{N}(0, \Sigma_1) \\
u \sim \mathcal{N}(0, \Sigma_2)
\]

\[
\Sigma_1 = \begin{bmatrix}
\sigma^2_{1,1} & \ldots & \sigma_{1P,1} \\
\vdots & \ddots & \vdots \\
\sigma_{1P,1} & \ldots & \sigma^2_{P,1}
\end{bmatrix}
\]

\[
\Sigma_2 = \begin{bmatrix}
\sigma^2_{1,2} & \ldots & \sigma_{1P,2} \\
\vdots & \ddots & \vdots \\
\sigma_{1P,2} & \ldots & \sigma^2_{P,2}
\end{bmatrix}
\] (4)

### 3.2 Maximum likelihood estimation

We can estimate the parameters of (2) and (3) using maximum likelihood. We must write the formula for the likelihood, which we do at the individual level because the random effects are correlated (identical) across the observations that constitute each individual’s history.

We consider multispell data. Multiprocess modeling often requires multispell data because qualitatively different events rarely happen simultaneously. That is, the exogenous or endogenous events throughout individual \(i\)’s life divide it into \(J\) episodes or spells. (For simplicity, subscript \(i\) is omitted.) For each process \(p\), the outcome of the process in episode \(j\) is characterized by two variables: the time variable \(t_{pj}\) and the failure-indicator variable \(y_{pj}\). The former measures the time to the occurrence of the event (or censoring). Termination of the process \(p\) in episode \(j\) is indicated by \(y_{pj} = 1\); censoring is indicated by 0 values.

We begin with the log likelihood for a process with no random effect. We can construct the log-likelihood contribution of any individual as follows. Log time to event is either observed or censored. With the exception of the first spell, log time to event is left-truncated because waiting times in spell \(j\) must be larger than the time elapsed until the beginning of that spell. The linear combination of the explanatory variables and the parameters is denoted by \(\theta_j = \beta' X_j\). Explanatory variables might change over time but are assumed to be constant within each spell \(j\). The log likelihood for individual \(i\) is given by

\[
\log L = \sum_{j=1}^{J} \left\{ y_j \log f_j + (1 - y_j) \log S_j - \log S_j^0 \right\}
\] (5)
Multiprocess survival models with cmp

where

\[ f_j = \phi(\log t_j - \theta_j; \sigma) \]
\[ S_j = \Phi\left(\frac{\theta_j - \log t_j}{\sigma}\right) \]
\[ S_0^j = \Phi\left(\frac{\theta_j - \log t_{j-1}}{\sigma}\right) \]

where \( \phi(\cdot) \) and \( \Phi(\cdot) \) are the normal probability density and cumulative density functions and \( t_0 = 0 \). (See, for instance, Klein and Moeschberger [2003].) Following Roodman (2011), (4) can be rewritten as

\[ \log L_i = \sum_{j=1}^{J} \log \int_{A_j}^{B_j} f(\varepsilon) d\varepsilon \]

where

\[ f(\varepsilon) = \phi(\varepsilon; \sigma) \]
\[ A_j = \log t_j - \theta_j \]
\[ B_j = y_j(\log t_j - \theta_j) + (1 - y_j) \infty \]
\[ C_j = \log t_{j-1} - \theta_j \]

The interpretation of the first integrand requires the conventions that \( \int_{a}^{a} f(\varepsilon) d\varepsilon = f(\varepsilon) \), and \( 0 \infty = 0 \).

The generalization of the log-likelihood expression to multiprocess models is straightforward. For simplicity, we consider only two simultaneous processes. The log-likelihood expression involves the two-dimensional integral

\[ \log L_i = \sum_{j=1}^{J} \log \int_{A_{1j}}^{B_{1j}} \int_{A_{2j}}^{B_{2j}} f(\varepsilon_{1j}, \varepsilon_{2j}) d\varepsilon_{2j} d\varepsilon_{1j} \]

where

\[ A_{pj} = \log t_{pj} - \theta_{pj} \]
\[ B_{pj} = y_{pj}(\log t_{pj} - \theta_{pj}) + (1 - y_{pj}) \infty \]
\[ C_{pj} = \log t_{p(j-1)} - \theta_{pj} \]

The log-likelihood formula extends to \( P \) processes analogously with order-\( P \) integrals. Adding an equation to model a dummy endogenous variable introduces further complications, though the principles remain the same. Users can refer to Roodman (2011) for more details on formulation and practical computation of the integrals.
To add a random effect to the two-process model, as an example, we redefine $\theta_{pj} = \beta_{pj} ' X_{pj} + u_p$, where $u$ is the random effect defined earlier. Because $u$ is unobserved, to compute the individual-level likelihood, we must integrate it out.

$$\log L_i = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \sum_{j=1}^{J} \log \frac{B_{A_{ij}} B_{A_{ij}}}{C_{1j} C_{2j}} f(\varepsilon_{1j}, \varepsilon_{2j}) d\varepsilon_{2j} d\varepsilon_{1j} \phi\{(u_1, u_2)'; \Sigma_2\} du_2 du_1$$

In general, the outer integrals can only be computed using numerical methods. The dominant methods are adaptive quadrature and simulation. Roughly speaking, adaptive quadrature computes the integrand at 6–24 points and averages the results with special weights (Rabe-Hesketh, Skrondal, and Pickles 2002). Simulation computes the integrand at 50, 100, or 500 points and takes the simple average of the results (Haan and Uhlendorff 2006; Train 2009). The cmp command offers both methods as part of a multilevel modeling extension that was added following Roodman (2011).

4 Syntax

The description of the syntax is restricted to components of the cmp command that are specific to multiprocess survival modeling. For the full syntax of cmp, consult Roodman (2011) and the help file for cmp.

4.1 Single-equation survival models

We begin with the syntax for ordinary (or single-process single-level) lognormal survival models. Lognormal survival models are interval-censored regression models of log failure-times. We can fit interval-censored regression models with cmp as follows. Like the official intreg command, cmp expects two dependent variables that indicate the lower and upper bounds of failure-time. For observed failures (or uncensored observations), these limits are the same and are equal to the observed log failure-time. For observed failures (or uncensored observations), these limits are the same and are equal to the observed log failure-time. For censored durations, the lower limit is the log time of the interview and the upper limit is infinity, meaning that the event will occur somewhere in the future. We let $lo$ and $hi$ denote the lower- and upper-limit variables, respectively. An interval regression model of log failure-times is estimated with cmp using the following syntax

```
        cmp ( [label: ] lo hi = indepvars, [noconstant] ) [ if ] [ in ] [ weight ],

        indicators(7) [ options ]
```

where label is used instead of lo in the output, the mandatory indicators(7) tells cmp that the equation to be estimated is an interval-censored regression, and options is just shorthand for any other options incorporated in cmp. The indicators(7) option can be replaced by indicators($\$cmp\_intreg$) if the cmp setup command is issued at the beginning of the session.
Multiprocess survival models with cmp

When using multispell data, researchers should account for the left-truncation of the outcome as well as the interdependence of residuals within individuals. Suppose the variable \( id \) identifies the individuals. Let \( durvar \) be the variable that records the log of entry time, which is the log duration of a state measured at the beginning of the spell. To account for the truncation of survival time as well as the dependence of residuals within individuals, the basic syntax is extended as follows:

\[
\text{cmp } \left( \text{label: } \text{lo hi = indepvars, trunckpoints(durvar .) [noconstant]} \right) \left[ \text{if} \right] \left[ \text{in} \right] \left[ \text{weight} \right], \text{indicators(7) vce(cluster id) [options]} \right)
\]

Another way to account for the correlation of residuals within individuals’ histories is to use a multilevel survival model. To accommodate the individual-level random effect, the basic syntax is modified as follows:

\[
\text{cmp } \left( \text{label: } \text{lo hi = indepvars, noconstant || id: } \text{lo hi = indepvars, noconstant} \right) \left[ \text{if} \right] \left[ \text{in} \right] \left[ \text{weight} \right], \text{indicators(7) [options]} \right)
\]

4.2 Syntax for multiprocess models

Next, we will fit systems of lognormal survival models. For simplicity, the exposition considers two processes. We let \( lo_1 \) and \( hi_1 \) denote the respective lower and upper limits for the first process, and we let \( lo_2 \) and \( hi_2 \) denote the respective lower and upper limits for the second process. Multiprocess modeling requires multispell data; we denote the respective entry times for the processes by \( durvar_1 \) and \( durvar_2 \). The syntax is

\[
\text{cmp } \left( [\text{label: } \text{lo hi = indepvars}, \text{trunckpoints(durvar_1 .}) \left[ \text{noconstant} \right] \right) \left[ \text{if} \right] \left[ \text{in} \right] \left[ \text{weight} \right], \text{indicators(7 7) vce(cluster id) [options]} \right)
\]

Note that the \text{indicators()} option now has two arguments, one for each equation. \text{indicators(7 7)} means that both equations are interval-censored. For clarity, you may instead type \text{indicators($cmp\_intreg$ $cmp\_intreg$)}, provided that you issued the \text{cmp setup} command at the beginning of the session.

\text{cmp} also allows users to fit lognormal survival models jointly with probit models. This is useful if the survival models of primary interest include endogenous dummy variables. For simplicity, we consider one survival process and one probit equation. We let \( lo \) and \( hi \) denote the respective lower and upper limits for the survival process. The endogenous dummy variable is denoted by \( dvar \). The syntax is
The 4 (alternatively, $\text{cmp probit}$) requests a probit model for the second equation.

Heckman-type modeling to control for sample selection bias is also possible. Suppose that the survival model applies to a subsample that is identified by the indicator variable \textit{sample}. Survival estimates can be adjusted for sample selection bias by using the following syntax:

\begin{verbatim}
\text{cmp ( label: } \texttt{lo hi = dvar indepvars.1, truncpoints(durvar .) [noconstant]})
  (dvar = indepvars.2, [noconstant]) [if] [in] [weight], indicators(7 4)
  vce(cluster id) [options]
\end{verbatim}

The second probit equation applies to the sample defined by the optional \texttt{if}, \texttt{in}, and \texttt{weight} syntax elements. The survival model, however, is fit using observations where \textit{sample} equals 1.

5 Examples

5.1 Introduction: The research problem and the dataset

Our examples consider the relationship between education and the timing of births. Evidence suggests that highly educated women who postpone the transition to motherhood space the first and the second births closer together. We use a sample dataset that comes with the statistical software aML (Lillard and Panis 2003). The dataset contains information on marital births and marriage durations for American women. The slightly modified and Stata-compatible version is obtained as follows:

\begin{verbatim}
. use http://web.uni-corvinus.hu/bartus/stata/divorce.dta
(Data on marriages (source: divorce4.raw, shipped with aML))
\end{verbatim}

The data have a multilevel structure; conception episodes are nested within marriages, and marriages are nested within individuals. Marriages within individuals are identified with the variable \texttt{marnum}, and conception episodes within marriages are identified with the variable \texttt{numkids}, which measures the number of kids at the beginning of conception episodes. Each row records the duration of a conception episode; the duration is the difference between two variables, \texttt{time} and \texttt{mardur}. \texttt{mardur} measures the duration of the marriage at the beginning of each conception episode, and \texttt{time} measures the date of separation (or interview date).
We use data on only the first two conception episodes within the first marriages. For convenience, we recode the `numkids` variable to indicate parity or birth order. The Stata commands are as follows:

```
. keep if marnum==1 & numkids<2  
   (4210 observations deleted)
. replace numkids = numkids+1  
   (5446 real changes made)
```

Next, we generate the limit variables `lo` and `hi`. For observed failures (or uncensored observations), these limits are the same and are equal to the observed log failure-time. For censored durations, the lower limit is the log time of the interview and the upper limit is infinity, meaning that the event will occur in the future. In our example, failure time is the time to conception, which is the difference between `time` and `mardur`. The Stata commands are as follows:

```
. generate lo = ln(time-mardur)
. generate hi = cond(birth==1, lo, .)  
   (1857 missing values generated)
```

We want to know the effect of education on the spacing of second births. To find our answer, we regress the log of waiting time to the second birth on education and other control variables in the sample of mothers of one child. We use the key explanatory variable `hereduc`, which is a categorical variable with three categories: primary, secondary, and higher education. (Actually, these variables are computed from years of schooling.) We then use secondary education as a reference category. We do this with the help of the factor-variable notation `ib2.hereduc`, which forces Stata to treat the second category as a reference. For simplicity, we use only the age at the beginning of the conception spell (that is, the age when the first child was born) and its square as control variables. We centered the age around 30 to minimize the correlation between age and age-squared. After centering the age variable manually, the age-squared variable is created with the help of the factor-variable notation. Note that the `numkids==2` condition identifies mothers of one child.
Mixed-process regression

| Coef.  | Std. Err. | z     | P>|z|  | [95% Conf. Interval] |
|--------|-----------|-------|------|----------------------|
| birth2 |           |       |      |                      |
| hereduc|           |       |      |                      |
| <12 years | -.025006 | .0566602 | -0.44 | 0.659 | -.1360579 .0860459 |
| 16+ years | -.2767705 | .0861557 | -3.21 | 0.001 | -.4456325 -.1079085 |
| age    | .041695   | .0056703 | 7.35  | 0.000 | .0305815 .0528086 |
| c.age#c.age | -.0012928 | .0004843 | -2.67 | 0.008 | -.0022421 -.0003436 |
| _cons  | 1.782701  | .0491509 | 36.27 | 0.000 | 1.686367 1.879035 |

/lnsig_1 | .101572 | .0193556 | 5.25  | 0.000 | .0636357 .1395082 |

/sig_1   | 1.10691 | .0214249 | 5.25  | 0.000 | 1.065704 1.149708 |

The third level of the hereduc variable has a negative and significant coefficient. This suggests that the duration of the second conception episode is shorter among highly educated women than among women with lower education. In other words, highly educated women appear to space the first and second births closer together than women with secondary education.

Next, we control for several forms of endogeneity and sample selection to check whether the estimate of $-0.277$ is robust.

### 5.2 Multilevel modeling of recurrent events

Our first concern with the previous outcome is that it might result from the following selection effect: education negatively affects the transition to first birth, hence, education is positively correlated with unobserved causes of fertility in samples of mothers (Kravdal 2007). Therefore, the comparison of the fertility outcomes across educational categories measures not only the true effect of education but also the effect of unobserved preferences or personality traits (Kravdal 2001). We can control for this selection effect by modeling the parity-specific transitions jointly.

One way to model the parity-specific transitions jointly is multilevel modeling. We consider the waiting times to only the first two births. Note that the origin for the second birth is set when the first birth happens. For simplicity, we consider the first and second transitions. The dataset is already in long format and ready for multilevel analysis: episodes within the same person appear in different records. The unobserved person-specific characteristics that affect the transition to births are captured with a
random effect at the level of individuals. The fixed part of the model is extended so that the effects of education and age will be conditional on the number of children previously born.

The new syntax element | id: specifies the random intercepts at the level of individuals. By default, **cmp** uses adaptive quadrature with 12 integration points when fitting the model. One can change the default behavior to simulation by using the **redraws()** option; see the **cmp** help file for details. To easily compare this model with the previous model, we use the second birth as the reference category so that the main effects of education and age are indeed effects conditional on already having a child.

```stata
.cmp (birth: lo hi = ib2.numkids##(ib2.hereduc c.age##c.age) || id:), > indicators(7)
(output omitted)
Mixed-process multilevel regression
Number of obs = 5446
LR chi2(9) = 374.78
Log likelihood = -8025.5353 Prob > chi2 = 0.0000

| Coef. | Std. Err. | z    | P>|z|  | [95% Conf. Interval] |
|-------|-----------|------|------|------------------------|
birth  |           |      |      |                        |
1.numkids | -.080781  | .1041972 | -0.78 | 0.438       | -.2850038  | .1234418 |
hereduc |           |      |      |                        |
<12 years | .0857769  | .0915342 | 0.94  | 0.349       | -.0936268  | .2651807  |
16+ years | -.2325267 | .1386793 | -1.68 | 0.094       | -.5043692  | .0392439  |
age     | .0321114  | .0093012 | 3.45  | 0.001       | .0138814   | .0503413  |
c.age#c.age | -.0010855 | .0008032 | -1.35 | 0.177       | -.0026597  | .0004887  |
numkids# |           |      |      |                        |
hereduc |           |      |      |                        |
1#<12 years | .3124741  | .114096  | 2.74  | 0.006       | .088852    | .5360961  |
1#16+ years | .494463   | .1744228 | 2.83  | 0.005       | .1526007   | .8363253  |
numkids#c.age | .0890989  | .014383  | 6.19  | 0.000       | .0609089   | .117289   |
1 |           |      |      |                        |
numkids# |           |      |      |                        |
c.age#c.age | .003612   | .0012324 | 2.93  | 0.003       | .0011965   | .0060274  |
1 |           |      |      |                        |
_cons   | 2.061027  | .0791532 | 26.04 | 0.000       | 1.90589    | 2.216165  |
/lnsig_1_1 | -.1440281 | .058177  | -2.48 | 0.013       | -.2580528  | -.0300033 |
/lnsig_1  | .2537051  | .0222605 | 11.40 | 0.000       | .2100753   | .2973349  |

Random-effects Parameters
<table>
<thead>
<tr>
<th>Estimate</th>
<th>Std. Err.</th>
<th>[95% Conf. Interval]</th>
</tr>
</thead>
</table>
Level: id Standard deviations
_cons     | .8658635  | .0503733  | .7725544  | .9704424  |
Level: Residuals Standard deviation
| 1.288792  | .0286892  | 1.233771  | 1.346266  |
```
The standard deviation of the individual-level random effect is about two-thirds of
the standard deviation of the residual. Thus the interclass correlation is about 0.31.
The timing of second births is, therefore, not interdependent of the timing of first births.
The estimated effect of higher education is negative and has a similar magnitude as in
the previous example, but it lacks statistical significance. Now we have no evidence
to conclude that higher-educated women would space the first and second births closer
together.

5.3 Simultaneous equations for recurrent events

An alternative modeling strategy is to fit the two survival models jointly as seemingly
unrelated equations. Estimation requires a wide data structure. After reloading the
data and selecting the relevant cases, we drop the unnecessary variables and transform
the data into wide format. Then, we place educational levels and age at the beginning
of the conception episode in parity-specific global macros. The commands are

```
use http://web.uni-corvinus.hu/bartus/stata/divorce, clear
(Data on marriages (source: divorce4.raw, shipped with aML))
keep if marnum==1 & numkids<2
(4210 observations deleted)
replace numkids = numkids+1
(5446 real changes made)
gen lo = ln(time-mardur)
gen hi = cond(birth==1, lo, .)
(1857 missing values generated)
replace age = age + mardur - 30
(5446 real changes made)
drop mardur time sep
reshape wide birth age lo hi, i(id) j(numkids)
(output omitted)
```
Two survival models are fit jointly as follows. The command lists two equations, labeled *birth2* and *birth1*. The mandatory `indicators(7 7)` option tells *cmp* that both equations are interval-censored regression models.

```plaintext
    . cmp (birth2: lo2 hi2 = ib2.hereduc c.age2##c.age2) > (birth1: lo1 hi1 = ib2.hereduc c.age1##c.age1), indicators(7 7)
```

```
Mixed-process regression
Number of obs = 3325
LR chi2(8) = 299.69
Log likelihood = -7874.9256 Prob > chi2 = 0.0000

                  Coef. Std. Err.      z    P>|z|     [95% Conf. Interval]
------------------ ------------- ------------- -------- -------- ------------------
birth2
    hereduc
       <12 years  .0093223 .0577222  0.16  0.872    -.1038111   .1224557
       16+ years -.2500341 .0867196 -2.88 0.004    -.4200015   -.0800668
    age2        .0378136 .0058145  6.50 0.000   .0264175   .0492098
   c.age2#c.age2 -.0012528 .0004896 -2.56 0.011   -.0022124   -.0002932
    _cons       1.82599 .0510606 35.76 0.000   1.725913   1.926067
birth1
    hereduc
       <12 years  .4280959 .0722000  5.93 0.000   .2865864   .5696053
       16+ years .2532307 .1111068  2.28 0.023   .0354653   .4709962
    age1        .1354200 .0123288 10.98 0.000   .1112560   .1595841
   c.age1#c.age1 .0027916 .0010290  2.71 0.007    .0007749   .0048084
    _cons       2.122898 .0751981 28.23 0.000   1.975513   2.270283
/lnsig_1      .1070316 .0197926  5.41 0.000    .0682389   .1458243
/lnsig_2      .5709637 .0163749 34.87 0.000    .5388695   .6030578
/atanhrho_12  .1329566 .0398857  3.33 0.001    .0547820   .2113132
    sig_1       1.112969 .0220286  5.07 0.000    1.069529   1.156409
    sig_2       1.769972 .0289831  60.80 0.000    1.714068   1.826569
    rho_12      1.352178 .0391889  34.31 0.000    1.270948   1.433409
```

To interpret the findings, we explore a similarity between the joint modeling of survival processes and the Heckman-type selection modeling. The first-birth equation implicitly defines the probability of being a mother at time \( t \) as \( P(\theta_s + \sigma \varepsilon_1 \leq \log t) = \Phi((\log t - \theta_s)/\sigma) \), where \( \theta_s \) denotes the linear combination of explanatory variables and coefficients appearing in the survival model of first births. The probit equation of a Heckman model defines the probability of being a mother as \( \Phi(\theta_p) \), where \( \theta_p \) denotes the linear combination of explanatory variables and coefficients of the probit equation. Hence, the coefficients of the first-birth equation in the joint survival model divided by the estimated standard-deviation coefficient \( \sigma \) implicitly define the coefficients of the probit selection equation. More precisely, because \( \theta_p \) corresponds to \(-\theta_s\), the survival coefficients define the probability of not being in the sample. This is not surprising,
because a large coefficient in the survival model corresponds to a small hazard of becoming a mother. To summarize, the first-birth equation in the above joint survival model has the same function as the probit model of being a mother in the Heckman selection model, except the survival equation of first births models the probability of not being in the sample, that is, of being childless.

Our primary interest is to explain why women with higher education space births closer together (see section 5.1). In the joint model, higher education in the second conception equation has a significant negative effect. The difference between the naïve estimate of $-0.277$, reported in section 5.1, and the joint estimate of $-0.250$ is the selection effect. The selection effect is small and negative ($-0.027$). The selection effect is negative because the cross-equation correlation of the residuals is positive, and higher education has a negative effect on the implicit sample inclusion probability because it has a positive effect on the waiting time to first births.

The joint estimation of survival models on sequentially constructed samples is a computationally attractive alternative to a Heckman-type selection modeling. Nevertheless, the `cmp` command accommodates the Heckman model (see section 4.2). In our example, the estimation of the survival model of the timing of second births requires the sample of mothers, while the estimation of becoming a mother uses the sample of all women. In our dataset, this condition can be expressed as $(\text{birth1}==1)$. The appropriate syntax is

```
. cmp (birth2: lo2 hi2 = $birth2) (sel1: birth1 = $birth1),
   > indicators("(birth1==1)*7" 4)
```

5.4 Simultaneous equations for different processes

We now turn to the joint model of the timing of second births and the timing of marital dissolutions. The motivation is that the timing of births depends on the quality of the marriage. Because match quality is unobserved, we must control for this omitted variable bias by fitting our survival model jointly with another survival model of marital dissolution. For simplicity, suppose that marital stability depends on education, age at the beginning of the conception episode, and the duration of the marriage. Given this specification, the timing of births should also depend on the duration of the marriage.

The example presented in this section also illustrates the use of the `cmp` command with multispell datasets. So far, we have used single-spell data. We turn to multispell data to accommodate for duration dependence, that is, the effect of the duration of the marriage on the timing of births and divorce. We split conception episode durations into smaller intervals by using the `stsplit` command. We use only first marriages and women who are at the risk of a second delivery. We use the following Stata commands to load the data and to create the multispell data structure, as well as the time-dependent variables for marriage duration and age at the beginning of a spell:
. use http://web.uni-corvinus.hu/bartus/stata/divorce, clear
(Data on marriages (source: divorce4.raw, shipped with aML))
. keep if marnum==1 & numkids==1
(7535 observations deleted)
. generate dur = time-mardur
. generate double id2 = _n
. stset dur, fail(sep==1) id(id)
(output omitted)
. stsplit bdur, at(1 2 5 10)
(4201 observations (episodes) created)
. replace mardur = mardur + _t0
(4201 real changes made)
. replace birth = 0 if sep==.
(2454 real changes made)
. replace sep = 0 if sep==.
(4201 real changes made)
. replace age = age + mardur - 30
(6322 real changes made)

Note that both marriage duration (mardur) and age are measured at the beginning of a spell and not at the time when events happen. Originally, the age variable measures age at the beginning of the marriage. The last command changes this variable into age at the beginning of a spell, centered around 30 years.

Next, we generate the dependent variables. We study two parallel processes (the timing of marriage dissolution and the timing of births), therefore, we have to generate two pairs of limit variables. The respective limit variable names for marriage dissolution and birth processes will begin with letters m and b. The Stata codes to create the dependent variables are

. generate mlo = ln(mardur+dur)
. generate mhi = cond(sep==1, mlo, .)
(6076 missing values generated)
. generate blo = ln(bdur+dur)
. generate bhi = cond(birth==1, blo, .)
(4854 missing values generated)

The only new explanatory variable is mardur, that is, the duration of the marriage at the beginning of the spell. The regression equations and the joint model are defined as follows. The new syntax elements include the request of clustered standard errors and the truncation option within both models. The latter accounts for the fact that times to event are left-truncated in multispell datasets, with the exception of the first spell.
To interpret the results, recall that the birth equation is not a structural equation but a reduced-form equation (see section 2). The structural effect of higher education must be recovered using (2). Because the covariance is the product of the displayed correlation and standard deviations, the structural coefficient in question can be estimated as

$$\hat{\alpha}_{1j} = \hat{\beta}_{1j} - \frac{\sigma_1 r_{12}}{\sigma_2} \hat{\beta}_{2j} = -0.292 - \frac{0.969(-0.513)}{0.926} 0.363 = -0.097$$
To assess the significance of this estimate, it is more useful to do the calculation with the official \texttt{nlcom} command. The resulting linear combination, labeled \texttt{nl1}, is as follows:

\begin{verbatim}
    . nlcom _b[birth:3.hereduc] - _b[divorce:3.hereduc]*
      > tanh(_b[atanhrho_12:_cons])*exp(_b[lnsig_1:_cons])/exp(_b[lnsig_2:_cons])
    \end{verbatim}

\texttt{nl1}: \_b[birth:3.hereduc] - \_b[divorce:3.hereduc]*
\begin{verbatim}
      > tanh(_b[atanhrho_12:_cons])*exp(_b[lnsig_1:_cons])/exp(_b[lnsig_2:_cons])
\end{verbatim}

| Coef.     | Std. Err. | z   | P>|z|  | [95% Conf. Interval] |
|-----------|-----------|-----|------|---------------------|
| \texttt{nl1} | -.0977424 | .1021125 | -0.96 | 0.338 | -.2978793 .1023945 |

The resulting nonlinear combination lacks statistical significance. Thus there is no evidence that education would have a direct effect on the timing of births. The negative net association between education and the time to second births is indirect; it is mediated through the latent satisfaction with marriage. The positive coefficient of higher education in the divorce equation suggests that highly educated women tend to live in relatively stable marriages. The negative correlation of the residuals suggests that women who are satisfied with their marriage tend to give birth to the second child earlier.

### 6 Conclusion

Seemingly unrelated systems of multilevel proportional hazards equations, often labeled multilevel multiprocess models, are routinely fit by demographers to adjust regression estimates for endogeneity and selection effects. In this article, we make a first step toward estimating systems of survival equations with Stata. We argue that systems of lognormal survival models can easily be fit with the user-written \texttt{cmp} command. After discussing the difference between multilevel multiprocess hazard models and multiprocess lognormal survival models, we demonstrate both the theoretical and the practical aspects of fitting models of the latter kind. Our exposition is restricted to the joint estimation of survival models, or the joint estimation of a survival and a probit model. We show how to fit these models and how to interpret the coefficients of interest. However, we do not consider systems including more than two equations and equations with error components.

Although no official Stata command is explicitly designed to estimate multiprocess survival models, the new \texttt{gsem} command in Stata 13 seems to enable users to fit systems of proportional hazards models with correlated random effects. Continuous-time exponential regression models can be restated as Poisson models (Skrondal and Rabe-Hesketh 2004), and \texttt{gsem} supports the Poisson distribution. However, \texttt{gsem} cannot handle the left-truncation of survival times, so it cannot be used to fit systems of continuous-time survival models on multispell data. In addition, \texttt{cmp} works in Stata 10.1 and later. In sum, the \texttt{cmp} command is a useful tool for survival modeling in Stata.
7 Acknowledgments

We thank an anonymous reviewer for helpful comments and suggestions.

Bartus received financial support from the TÁMOP project 4.2.1/B-09/1/KMR-2010-0005 of the Corvinus University of Budapest, as well as from the Hungarian Scientific Research Fund (OTKA) within the research project Mapping Family Transitions: Causes, Consequences, Complexities, and Context (grant number K109397).

8 References


Multiprocess survival models with cmp


About the authors

Tamás Bartus is an associate professor of sociology at the Corvinus University of Budapest and a senior research fellow at the Hungarian Demographic Research Institute in Budapest, Hungary.

David Roodman is a freelance public policy consultant in Washington, DC.

Appendix. Multiprocess modeling with gsem

This appendix illustrates the capabilities of the `gsem` command. We thank the anonymous reviewer for the syntax and the permission to include the syntax in this appendix.

➤ Example 1. Introduction: The research problem and the dataset

```
. cmp (birth2: lo hi = ib2.hereduc c.age2##c.age2) if numkids==2, indicators(7)
   . gsem (lo <- ib2.hereduc c.age2##c.age2 if numkids==2,
   > family(normal, udepvar(hi)))
```

➤ Example 2. Multilevel modeling of recurrent events

```
. cmp (birth: lo hi = ib2.numkids##(ib2.hereduc c.age2##c.age2) || id:),
   > indicators(7)
   . gsem (lo <- ib2.numkids##(ib2.hereduc c.age2##c.age2) M[id],
   > family(normal, udepvar(hi)))
```
Example 3. Simultaneous equations for recurrent events

```stata
. cmp (birth2: lo2 hi2 = ib2.hereduc c.age2##c.age2)
> (birth1: lo1 hi1 = ib2.hereduc c.age1##c.age1), indicators(7 7)
. gsem (lo2 <- ib2.hereduc c.age2##c.age2 M@1), family(normal, udepvar(hi2))
> (lo1 <- ib2.hereduc c.age1##c.age1 M, family(normal, udepvar(hi1)))
```

Example 4. Simultaneous equations for different processes (this example cannot be replicated with gsem because it does not support truncated outcomes)

```stata
. cmp (birth: blo bhi = ib2.hereduc c.age##c.age mardur, truncpoints(ln(bdur) .))
> (divorce: mlo mhi = ib2.hereduc mardur, truncpoints(ln(mardur) .)),
> vce(cluster id) indicators(7 7)
```
dhreg, xtdhreg, and bootdhreg: Commands to implement double-hurdle regression

Christoph Engel
Max Planck Institute for Research on Collective Goods
Bonn, Germany
engel@coll.mpg.de

Peter G. Moffatt
School of Economics
University of East Anglia
Norwich, UK
P.Moffatt@uea.ac.uk

Abstract. The dhreg command implements maximum likelihood estimation of the double-hurdle model for continuously distributed outcomes. The command includes the option to fit a p-tobit model, that is, a model that estimates only an intercept for the hurdle equation. The bootdhreg command (the bootstrap version of dhreg) may be convenient if the data-generating process is more complicated or if heteroskedasticity is suspected. The xtdhreg command is a random-effects version of dhreg applicable to panel data. However, this estimator differs from standard random-effects estimators in the sense that the outcome of the first hurdle applies to the complete set of observations for a given subject instead of applying at the level of individual observations. Command options include estimation of a correlation parameter capturing dependence between the two hurdles.

Keywords: st0359, dhreg, xtdhreg, bootdhreg, hurdle model, double-hurdle model, random-effects double-hurdle model, tobit, p-tobit, inverse Mills ratio, bootstrapping

1 Introduction

The double-hurdle model, introduced by Cragg (1971), embodies the idea that an individual’s decision on the extent of participation in an activity is the result of two processes: the first hurdle, determining whether the individual is a zero type, and the second hurdle, determining the extent of participation given that the individual is not a zero type. A key feature of the model is that there are two types of zero observations: an individual can be a zero type, and the outcome will always be zero whatever his or her circumstances at the time of the decision; alternatively, the individual might not be a zero type, but his or her current circumstances might dictate that the outcome is zero—this sort of zero is usually classified as a censored zero after (Tobin 1958).

In addition to naturally incorporating the zero type, the double-hurdle model allows estimation of the proportion of the population that is of the zero type. Better still, it allows the probability of a subject’s being zero type to depend on the subject’s characteristics.
The double-hurdle model has previously been applied in a variety of contexts. Jones (1989) applied it to cigarette consumption by individuals, with the justification that a proportion of the population would never smoke whatever circumstances they found themselves in. Burton, Tomlinson, and Young (1994) applied the model to meat consumption by single-adult households, with the justification that a proportion of the population of single adults must be vegetarian and therefore destined to record zero consumption of meat. The model has also been applied in models of loan default (Dionne, Artis, and Guillén 1996; Moffatt 2005), where it is assumed that a proportion of borrowers would, out of principle, never default on a loan.

The practice of fitting hurdle models is well developed in the context of count-data outcomes. See, for example, the user-written Stata commands ztpnm and hnbclg. McDowell (2003) has provided advice from the Stata help desk on the required programming. The practice is less developed in the context of continuous outcomes, the case of interest here. After our paper was accepted for publication, the dblhurdle command and the accompanying article by Garcia (2013) became available. Unlike our commands, the dblhurdle command offers weighted estimation. By contrast, our dhreg command allows the capture of possible correlation of the error terms between the hurdle equation and the equation for choices that pass the hurdle by using the inverse Mills’s ratio.

We also extend the hurdle framework to panel data (again going beyond dblhurdle). The panel-hurdle model has been applied to household milk consumption by Dong and Kaiser (2008). From that starting point, we extend the panel-hurdle model in an important way: we assume a nonzero correlation between the individual-specific error terms in the two hurdles. Hence, we achieve superior efficiency by fitting the panel-hurdle model with dependence. In a double-hurdle model with just one observation per subject, there are problems identifying this correlation (Smith 2003). In contrast, with panel data, we shall see that the parameter can be estimated with reasonably high precision.

The panel-hurdle model is potentially important in experimental economics, in which there are several natural applications, including public goods experiments and dictator games. Sometimes, each subject engages in only one task; however, it is more common for each subject to engage in a sequence of tasks throughout an experiment. The extension of the model to panel data requires care because the outcome of the first hurdle—that is, the determination of whether a respondent is of the zero type—must apply to that respondent for every period. Switching in and out of the zero type is ruled out. In contrast, the outcome of the second hurdle—that is, the amount consumed or contributed in any period—is determined at the level of individual observations. In principle, respondents classified as nonparticipants must necessarily consume or contribute zero in every period. We also offer a bootstrap version of the estimator, again going beyond the dblhurdle command. Bootstrapping can, for instance, be helpful if choices are nested in individuals who are, in turn, nested in higher-level units.
Commands to implement double-hurdle regression

In section 2, we cover theoretical aspects of the double-hurdle model, specifying the likelihood function for each model. In section 3, we do the same for the panel-hurdle model. We then introduce the user-written Stata commands and syntax in section 4. We demonstrate the commands using a simulated dataset in section 5.

2 Double-hurdle and related models

When referring to examples in the description of models, we will use the term “contribution” to represent the outcome variable; the term “consumption” could also be used.

2.1 Tobit

A natural starting point is the tobit model (Tobin 1958) because the hurdle model is an extension of it. Tobit-type models, or censored regression models, are required when the dependent variable is censored, that is, when there is an accumulation of observations at the limits of the range of the variable. The lower limit of the range is usually zero, and censoring is usually zero censoring, although sometimes, we are required to deal with upper censoring, where there is an accumulation of observations at the maximum. In other contexts, there is censoring from below but at a cutoff point different from zero. The software handles all of these. Yet when introducing the theory, we shall focus on lower censoring at zero.

We start with a linear equation in which the dependent variable is a latent (unobserved) variable, \( y^*_i \), representing the desired contribution of subject \( i \):

\[
\begin{align*}
y^*_i &= x_i^\prime \beta + \varepsilon_i \\
\varepsilon_i &\sim N(0, \sigma^2)
\end{align*}
\]

The desired contribution is assumed to be a linear function of the observed subject characteristics and treatment variables contained in the vector \( x_i \), plus a normally distributed random error. The important feature of \( y^*_i \) is that it can be negative: subjects are permitted to desire to contribute a negative amount. Of course, if a subject does desire to contribute a negative amount, most experimental designs would constrain the subject to contribute zero; if the subject desires to contribute any positive amount, this amount will be his or her actual observed contribution. This amounts to what is known as a censoring rule:

\[
\begin{align*}
y_i &= y^*_i \text{ if } y^*_i > 0 \\
y_i &= 0 \text{ if } y^*_i \leq 0
\end{align*}
\]

\( y_i \) is the observed contribution of subject \( i \). Therefore, the censoring rule shows the relationship between desired and actual contributions.

1. An interesting new development is the emergence of take games, dictator games in which some treatments allow dictators to take money away from the recipient, that is, to give less than zero (see List [2007]; Bardsley [2008]).
In the situation where we have lower censoring at zero, there are two regimes of behavior: zero observations and positive observations. The sample log-likelihood function is constructed by combining contributions for each regime as follows:

\[
\log L = \sum_{i=1}^{n} \left[ I_{y_i=0} \ln \left\{ \Phi \left( -\frac{x_i^\prime \beta}{\sigma} \right) \right\} + I_{y_i>0} \ln \left\{ \frac{1}{\sigma} \phi \left( \frac{y_i - x_i^\prime \beta}{\sigma} \right) \right\} \right]
\]  

(2)

\( I \) is the indicator function, taking the value one if the subscripted expression is true, and zero otherwise. \( \log L \) is maximized with respect to the parameters contained in the vector \( \beta \) and the standard deviation parameter \( \sigma \).

### 2.2 \( p \)-tobit

The overrestrictive feature of the tobit model described in section 2.1 is that it allows only one type of zero observation, and the implicit assumption is that zeros arise because of subject circumstances and treatments. The obvious way to relax this is to assume the existence of an additional class of subjects who would never contribute under any circumstances.

In the first instance, let us assume that the proportion of the population who are potential contributors is \( p \), so that the proportion of the population who would never contribute is \( 1 - p \). For the former group, the tobit model applies, while for the latter group, the contribution is automatically zero. This assumption leads to the \( p \)-tobit model, originally proposed by Deaton and Irish (1984) in the context of household consumption decisions, where they were essentially allowing for a class of abstinent consumers for each good modeled. The log-likelihood function for the \( p \)-tobit model is\(^2\)

\[
\log L = \sum_{i=0}^{n} \ln \left\{ 1 - p \Phi \left( \frac{x_i^\prime \beta}{\sigma} \right) \right\} + \sum_{i=0}^{n} \ln \left\{ p \frac{1}{\sigma} \phi \left( \frac{y_i - x_i^\prime \beta}{\sigma} \right) \right\}
\]

(3)

Maximizing (3) returns an estimate of the parameter \( p \), in addition to those of \( \beta \) and \( \sigma \) obtained under tobit.

### 2.3 Double hurdle

Because the class of subjects who would never contribute may be the focus of the analysis, it is desirable to investigate which types of subjects are most likely to appear in this class. With this in mind, we assume that the probability of a subject’s being in the said class depends on a set of subject characteristics. In other words, we shall generalize the \( p \)-tobit model of section 2.2 by allowing the parameter \( p \) to vary according to subject characteristics. This generalization leads us to the double-hurdle model.

\(^2\) For readers unfamiliar with the structure of log-likelihoods such as (2) and (3), a useful basic principle is that because of the symmetry of the normal distribution, \( \Phi(-z) = 1 - \Phi(z) \).
Commands to implement double-hurdle regression

As the model name suggests, subjects must cross two hurdles to contribute. The first hurdle needs to be crossed to be a potential contributor. Given that the subject is a potential contributor, his or her current circumstances and treatment in the experiment dictate whether he or she contributes—this is the second hurdle.

The double-hurdle model contains two equations and can be given the interpretation of a combined probit and tobit estimator. We write

\[
\begin{align*}
d_i^* &= z_i'\alpha + \varepsilon_{1,i} \\
y_i^{**} &= x_i'\beta + \varepsilon_{2,i} \\
(\varepsilon_{1,i}, \varepsilon_{2,i}) &\sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & \sigma^2 \end{pmatrix}\right)
\end{align*}
\]

(4)

The variance of \(\varepsilon_{1,i}\) is normalized to 1, as required for identification, because the outcome of the first hurdle is binary. The diagonality of the covariance matrix implies that the two error terms are assumed to be independently distributed.

The first hurdle is represented by

\[
\begin{align*}
d_i &= 1 \text{ if } d_i^* > 0 \\
d_i &= 0 \text{ if } d_i^* \leq 0
\end{align*}
\]

(5)

The first hurdle is thus assumed to be defined by the latent variable \(d_i^*\). The second hurdle closely resembles the tobit model (1):

\[
y_i^* = \max(y_i^{**}, 0)
\]

(6)

Finally, the observed variable, \(y_i\), is determined as

\[
y_i = d_i y_i^*
\]

(7)

The log-likelihood function for the double-hurdle model is

\[
\log L = \sum_0 \ln \left\{ 1 - \Phi(z_i'\alpha)\Phi\left(\frac{x_i'\beta}{\sigma}\right) \right\} + \sum_+ \ln \left\{ \Phi(z_i'\alpha)\frac{1}{\sigma}\phi\left(\frac{y_i - x_i'\beta}{\sigma}\right) \right\}
\]

(8)

When the lower hurdle is \(y_{\min} \neq 0\), the first term in (8) changes to \(\sum_0 \ln[1 - \Phi(z_i'\alpha)\Phi\{(x_i'\beta - y_{\min})/\sigma\}]\). When the hurdle is an upper hurdle, \(y_{\max}\), the first term becomes \(\sum_+ \ln[1 - \Phi(z_i'\alpha)\Phi\{(y_{\max} - x_i'\beta)/\sigma\}]\).

Figure 1 is useful for understanding the model defined in (4)–(7). The concentric ellipses are contours of the joint distribution of the latent variables \(d^*\) and \(y^{**}\). These ellipses are centered on the point \((z_i'\alpha, x_i'\beta)\) so that the whole distribution moves around with changes in the values taken by the explanatory variables. The likelihood component associated with noncontribution [that is, the first term in curly braces in (8)] is represented by the probability mass under the L-shaped region comprising the
northwest, southwest, and southeast quadrants of the graph; the likelihood component associated with a contribution [the second term in curly braces in (8)] is represented by a thin strip of the probability mass within the northeast quadrant at the value of the observed contribution (one such value is depicted in the diagram).

Finally, consider a double-hurdle model in which there are no explanatory variables in the first-hurdle equation. There is only an intercept, $\alpha_0$. The likelihood function becomes

$$\log L = \sum \ln \left( 1 - \Phi(\alpha_0) \Phi \left( \frac{x_i' \beta}{\sigma} \right) \right) + \sum \ln \left( \Phi(\alpha_0) \frac{1}{\sigma} \phi \left( \frac{y_i - x_i' \beta}{\sigma} \right) \right)$$

$\Phi(\alpha_0)$ is now a scalar. If we rename this scalar as $p$, we have the $p$-tobit model defined in (3).

This gives us a way of fitting the $p$-tobit model. We fit the double-hurdle model with no explanatory variables in the first hurdle. We then transform the estimate of the intercept parameter in the first hurdle, $\alpha_0$, using

$$p = \Phi(\alpha_0)$$

This gives the estimate of the parameter $p$ in the $p$-tobit model. The delta method is required to obtain a standard error for this estimate.
Commands to implement double-hurdle regression

2.4 The single-hurdle model

The single-hurdle model is a model that has the property of first-hurdle dominance (Jones 1989). This essentially requires that any individual who passes the first hurdle necessarily has a positive outcome. Hence, there is only one source of zeros, the zero type; censored zeros are ruled out.

The formal definition of the single-hurdle model is similar to that of the double-hurdle model given in section 2.3; the only difference is that (6) changes from a rule embodying zero censoring to one embodying zero truncation:

\[ y_i^* = y_i^{**} \text{ if } y_i^{**} > 0 \]
\[ y_i^* \text{ unobserved if } y_i^{**} \leq 0 \]

As will be mentioned in the next section, logical problems arise when we try to extend the single-hurdle model to the panel-data setting. For this reason, we do not pay close attention to this model.

3 Extension to panel data

3.1 The basic panel-hurdle model

Until this point in the article, we have been concerned with estimation with one cross-section of data. We now progress to panel data. The panel-hurdle model was developed by Dong and Kaiser (2008), who applied the model to household milk consumption. Here we assume that we have \( n \) subjects, each of whom participated in \( T \) tasks. We denote \( y_{it} \) as the decision (that is, the contribution) of subject \( i \) in task \( t \). The two hurdles are defined as follows:

First hurdle

\[ d_i^* = z_i' \alpha + \varepsilon_{1,i} \]
\[ d_i = 1 \text{ if } d_i^* > 0; d_i = 0 \text{ otherwise} \]
\[ \varepsilon_{1,i} \sim N(0, 1) \]

Second hurdle

\[ y_{it}^{**} = x_{it}' \beta + u_i + \varepsilon_{2,it} \]
\[ y_{it}^* = \max(y_{it}^{**}, 0) \]
\[ \left( \varepsilon_{1,i}, u_i, \varepsilon_{2,it} \right) \sim N \left( \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \sigma_u & 0 \\ \rho \sigma_u & \sigma_u^2 & 0 \\ 0 & 0 & \sigma^2 \end{pmatrix} \right) \]
The central feature of the panel-hurdle model is that the first hurdle has only one outcome per subject, and that outcome applies to all observations for that subject. For example, if subject \( i \) falls at the first hurdle \( (d_i = 0) \), then all observations on \( y \) for subject \( i \) must be 0 \( (y_{it} = 0, \ t = 1, \ldots, T) \). This feature is essential to capture the concept of the zero type. If a subject is a zero type, then the subject will necessarily contribute zero on every occasion on which he or she is observed.

Note that the second hurdle contains a subject-specific random-effects term \( (u_i) \) that allows between-subject heterogeneity and thereby within-subject dependence. In the specification of the joint distribution of the three stochastic terms, we have assumed that the correlation between \( \varepsilon_{1,i} \) and \( u_i \) is \( \rho \). In this section, we are considering the model with independence between the two hurdles, so we assume that \( \rho = 0 \). In section 3.4 (panel-hurdle model with dependence), we will allow \( \rho \) to be nonzero and consider the strategy for estimating this parameter.

### 3.2 The panel single-hurdle model

We introduced the single-hurdle model in section 2.4. This is a model satisfying first-hurdle dominance: passing the first hurdle necessarily implies a positive outcome. In the panel setting, first-hurdle dominance gives rise to a logical problem. If an individual passes the first hurdle, his or her outcomes would need to be positive in every period. We already know that if the individual falls at the first hurdle, the outcome is zero every time. Hence, first-hurdle dominance rules out a mixture of zero and positive outcomes for a given individual. This is clearly a serious problem because most panel-data sets would be expected to contain such mixtures. For this reason, we shall restrict attention to the framework of the panel double-hurdle model introduced in section 3.1, in which the zero censoring assumed in the second hurdle allows mixtures of zeros and positive observations for a given individual.

### 3.3 Construction of likelihood function

Conditional on \( d_i = 1 \) (and also on the heterogeneity term \( u_i \)), we obtain something very similar to the random-effects tobit likelihood:

\[
(L_i | d_i = 1, u_i) = \prod_{t=1}^{T} \left\{ \frac{1}{\sigma} \left[ \Phi \left( \frac{x_{it}'(\beta + u_i)}{\sigma} \right) \right]^{I(y_{it}=0)} \left( 1 - \Phi \left( \frac{y_{it} - x_{it}'\beta - u_i}{\sigma} \right) \right)^{I(y_{it}>0)} \right. \]

\[
(9)
\]
Commands to implement double-hurdle regression

Conditional on \( d_i = 0 \), the likelihood is trivial and depends on whether all observations are zero for subject \( i \):

\[
(L_i|d_i = 0) = 0 \text{ if } \sum_{t=1}^{T} y_{it} > 0
\]
\[
= 1 \text{ if } \sum_{t=1}^{T} y_{it} = 0
\]  

(10)

The likelihood (conditional on \( u_i \)) for subject \( i \) is then obtained as a weighted average of (9) and (10), with weights given by the probabilities \( P(d_i = 1) \) and \( P(d_i = 0) \), which are obtained from the first-hurdle equation.

\[
(L_i|u_i) = \Phi(z_i'\alpha)(L_i|d_i = 1, u_i) + (1 - \Phi(z_i'\alpha))(L_i|d_i = 0)
\]  

(11)

Finally, the marginal likelihood for subject \( i \) is obtained by integrating (11) over \( u \),

\[
L_i = \int_{-\infty}^{\infty} (L_i|u)f(u)du
\]

where \( f(u) \) is the normal \((0, \sigma_u^2)\) density function for \( u \).

The sample log-likelihood function is then given by

\[
\log L = \sum_{i=1}^{n} \ln L_i
\]

3.4 Panel-hurdle model with dependence

The model developed above assumes that there is no correlation between the error terms in the two hurdles. In this section, this assumption is relaxed.

Subject \( i \)'s idiosyncratic propensity to pass the first hurdle is represented by the error term \( \varepsilon_{1,i} \); \( i \)'s idiosyncratic propensity to contribute, conditional on passing the first hurdle, is represented by \( u_i \). It is between these two terms that we introduce a correlation:

\[
\text{corr}(\varepsilon_1, u) = \rho
\]

How is the correlation parameter \( \rho \) incorporated in estimation? Let us return to the first hurdle:

\[
d_i^a = z_i'\alpha + \varepsilon_{1,i}
\]
\[
d_i = 1 \text{ if } d_i^a > 0; \ d_i = 0 \text{ otherwise}
\]
\[
\varepsilon_{1,i} \sim N(0, 1)
\]
Because \(\text{corr}(\varepsilon_1, u) = \rho\), we can represent \(\varepsilon_1\) as

\[
\varepsilon_1 = \rho \frac{u}{\sigma_u} + \sqrt{1 - \rho^2} \xi
\]

where \(\xi \sim N(0, 1)\) and \(\xi \perp u\). The requirement for passing the first hurdle becomes

\[
d_i = 1 \text{ if } \xi > -\frac{z_i'\alpha + \rho \frac{u}{\sigma_u}}{\sqrt{1 - \rho^2}}
\]

from which the probability of passing the first hurdle becomes

\[
\Phi\left(\frac{z_i'\alpha + \rho \frac{u}{\sigma_u}}{\sqrt{1 - \rho^2}}\right)
\]

(12)

In estimation, the Halton draws used to represent realizations of \(u\) in the second hurdle also appear in the probability of passing the first hurdle in accordance with (12).

In the standard double-hurdle model with dependence (one observation per subject), there can be problems identifying the correlation coefficient \(\rho\) (Smith 2003). However, with panel data and use of the estimation approach outlined in this section, the parameter can be estimated precisely.

### 3.5 Two-step estimation of the dependence model

Heckman (1979) developed a procedure that treats correlation as an omitted variable problem. If the error terms are indeed correlated, the inverse Mills ratio from the first component must have explanatory power for the second component. Specifically, the coefficient of this additional regressor is precisely the covariance of the two error terms. Using this approach, we have a tractable way of fitting a double-hurdle model if the error terms for the \(d_i^*\) and the \(y_i^{**}\) are possibly correlated:

- Estimate the double-hurdle model assuming covariance to be zero.
- Generate the inverse Mills ratio from the first component. If the first component is estimated with probit, it is given by
  \[
  \frac{\phi(z_i'\alpha)}{\Phi(z_i'\alpha)}
  \]
- Refit the double-hurdle model with the first component as before, but with the second component (the \(y_i^{**}\) equation) estimated with the inverse Mills ratio as an additional explanatory variable. If the additional regressor turns out significant, this suggests that the two processes are indeed correlated.

If the problem at hand invites exclusion restrictions for the first hurdle, these restrictions identify the effect of the outer hurdle. Otherwise, as in a standard Heckman model, identification is through functional form only.
3.6 Bootstrap version of the model

The panel version of the double-hurdle model assumes a specific, randomly distributed error, uncorrelated with observables. Because there is no acknowledged fixed-effects version of the panel tobit model, a fortiori there is no reliable fixed-effects estimator for double-hurdle regression, which builds on tobit. As a safeguard, we offer a nonparametric version using bootstrapping, the `bootdhreg` estimator, which is the bootstrap version of the `dhreg` estimator. Users can match the panel structure of their data-generating process by sampling from participants as clusters. This command may also be convenient if there are reasons to doubt the normality assumption on which the maximum likelihood procedure of `dhreg` is based or if there are higher-level clusters. An illustration in the framework of the smoking example would be two samples taken from different subpopulations, say, the pupils of two different schools. Of course, the bootstrap clusters are less efficient than the random-effects model because they do not make any assumptions about covariance terms.

3.7 Estimation

Estimation of the panel-hurdle model is performed using the method of maximum simulated likelihood (Train 2009). This requires the use of Halton draws, which, when converted to normality, represent simulated realizations of the random-effects term $u$. In the model with dependence, in accordance with (12), the simulated values also appear in the probability of passing the first hurdle. Maximization of the simulated likelihood function is performed using the ml routine in Stata.

4 The dhreg, xtdhreg, and bootdhreg commands

4.1 Syntax

dhreg depvar indepvars [ if ] [ in ] [, up ptobit hd(varlist) millr ]

xtdhreg depvar indepvars [ if ] [ in ] [, up ptobit hd(varlist) uncorr trace
difficult constraints(numlist) ]

bootdhreg depvar indepvars [ if ] [ in ] [, up ptobit hd(varlist) millr
margins(string) seed(integer) reps(integer) strata(varlist) cluster(varlist)
capt maxiter(integer) ]
4.2 Options for dhreg

up specifies that the upper, not the lower, limit of the support of the dependent variable be treated as the hurdle.

ptobit estimates the equation for the outer hurdle with just the intercept.

hd(varlist) allows a set of explanatory variables for the outer hurdle that differs from the explanatory variables for the inner hurdle and those realizations of the dependent variable that surmount the hurdle.

millr estimates a second version of the model with the inverse Mills ratio controlling for potential correlation of the error terms.

4.3 Options for xtdhreg

up specifies that the upper, not the lower, limit of the support of the dependent variable be treated as the hurdle.

ptobit estimates the equation for the outer hurdle with just the intercept.

hd(varlist) allows a set of explanatory variables for the outer hurdle that differs from the explanatory variables for the inner hurdle and those realizations of the dependent variable that surmount the hurdle.

uncorr assumes that the error terms of the hurdle equation and of the main equation are uncorrelated.

trace displays coefficients from each iteration.

difficult uses an alternative, more calculation-intense algorithm for approximation (which may help if the model does not converge).

constraints(numlist) makes it possible for users to constrain the model.

4.4 Options for bootdhreg

up specifies that the upper, not the lower, limit of the support of the dependent variable be treated as the hurdle.

ptobit estimates the equation for the outer hurdle with just the intercept.

hd(varlist) allows a set of explanatory variables for the outer hurdle that differs from the explanatory variables for the inner hurdle and those realizations of the dependent variable that surmount the hurdle.

millr estimates a second version of the model with the inverse Mills ratio controlling for potential correlation of the error terms.

margins(string) calls for bootstrap estimates of marginal effects.
Commands to implement double-hurdle regression

**seed**(integer) fixes a seed for the randomization (as a default, one seed is implemented so that results can be replicated).

**reps**(integer) defines the number of bootstrap repetitions. The default is **reps**(50).

**strata**(varlist) orders the bootstrap to be stratified.

**cluster**(varlist) defines higher-order aggregates from which samples are drawn.

**capt** ignores repetitions that do not converge. It may be useful if, for instance, the program does not converge for some samples because of clustering. The bootstrap results are then taken from the remaining draws.

**maxiter**(integer) limits the number of iterations. **maxiter()** is also useful if the maximum likelihood routine has a hard time converging. **maxiter()** should be combined with **capt**. The default is **maxiter**(50).

5 Examples

Here is a stylized example for using the **dhreg** command. It works with simulated data with the following data-generating process:

\[
\begin{align*}
d_i^* &= \left\{ \begin{array}{ll} 1 & \text{if } -2 + 4 \times z_i + \varepsilon_{i1} > 0 \\ 0 & \text{otherwise} \end{array} \right. \\
y_{i**} &= 0.5 + 0.3 \times x_i + \varepsilon_{i2} \\
y_i^* &= \left\{ \begin{array}{ll} y_{i**} & \text{if } y_{i**} > 0 \\ 0 & \text{otherwise} \end{array} \right. \\
y_i &= d_i^* \times y_i^* \\
\varepsilon_{i1} &= 0.5 \times \varepsilon_{i2} + \sqrt{(1 - 0.5^2)} \times \eta_i \\
\varepsilon_{i2}, \eta_i &\sim N(0, 1) \\
\text{corr}(\varepsilon_{i1}, \varepsilon_{i2}) &= 0.5 \\
z_i, x_i &\sim U(0, 1)
\end{align*}
\]

In these data, the latent process defined by the first equation generates the first hurdle, which is determined by a constant, the (uniformly distributed) exogenous variable \( z \), and the error term for this process, \( \varepsilon_{i1} \). This error term has a correlation of 0.5 with the error term of the second process, \( \varepsilon_{i2} \); these correlated errors are simulated by a separate normal variate, \( \eta_i \). The latent process defined by the second equation generates the magnitude of those observations that pass both hurdles. This second process is determined by (uniformly distributed) exogenous variable \( x \) and the (normally distributed) error term, \( \varepsilon_{i2} \). Through the final equation (\( y_i = d_i^* \times y_i^* \)), the observed dependent variable, \( y_i \), is zero if either the first or the second hurdle is not passed and otherwise has the magnitude of the second latent variable, \( y_{i**} \). Figure 2 shows the resulting data. Nearly half fall at the first hurdle. A sizable portion falls at the second hurdle.
In the first step, we show that tobit is not appropriate for this data-generating process. The model does not pick up the effect of \( x \). The coefficient is insignificant and about half as large as the actual effect.

```
. tobit y x, ll(0)
Tobit regression
Number of obs = 1000
LR chi2(1) = 0.70
Prob > chi2 = 0.4015
Log likelihood = -1127.5531 Pseudo R2 = 0.0003

             Coef.  Std. Err.      t    P>|t|     [95% Conf. Interval]
------------- ------------- -------- -------- -------- ------------------------
      y       .1549814    .1846556    0.84    0.402      -.2073759    .5173388
        x       -.3249669    .1145513   -2.84    0.005     -.5497557    -.100178
_cons     -1.490576    .0589589   -1.374879 0.17388    -.5497557    -.100178
/sigma    1.374879    1.606274
```

In the next step, we present output from the double-hurdle model but still (wrongly, given our simulation) assume that error terms are uncorrelated. For expositional reasons, we had the effect of \( x \) designed to be small. It is now properly estimated, as is the effect of \( z \) and the estimate of the standard error. However, all coefficients are still slightly biased.
Commands to implement double-hurdle regression

.dhreg y x, hd(z)
(output omitted)

maximum likelihood estimates of double hurdle model

N = 1000
log likelihood = -947.24225
chi square hurdle equation = 81.817097
p hurdle equation = 1.495e-19
chi square above equation = 4.7952012
p above equation = .02853912
chi square overall = 90.440744
p overall = 2.296e-20

<table>
<thead>
<tr>
<th>coef</th>
<th>se</th>
<th>z</th>
<th>p</th>
<th>lower CI</th>
<th>upper CI</th>
</tr>
</thead>
<tbody>
<tr>
<td>hurdle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>3.877918</td>
<td>0.428722</td>
<td>9.04528</td>
<td>0</td>
<td>3.037636</td>
</tr>
<tr>
<td>_cons</td>
<td>-1.852446</td>
<td>0.152426</td>
<td>-12.15308</td>
<td>5.52e-34</td>
<td>-2.151195</td>
</tr>
<tr>
<td>above</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>0.3673263</td>
<td>0.1677446</td>
<td>2.189795</td>
<td>0.0285391</td>
<td>0.0385529</td>
</tr>
<tr>
<td>_cons</td>
<td>0.6856148</td>
<td>0.113751</td>
<td>6.027391</td>
<td>1.67e-09</td>
<td>0.462667</td>
</tr>
<tr>
<td>sigma</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_cons</td>
<td>0.9423028</td>
<td>0.0564864</td>
<td>16.68193</td>
<td>0</td>
<td>0.8315915</td>
</tr>
</tbody>
</table>

In the final step, we estimate the correlation using the inverse Mills ratio. The model rightly suggests that error terms are correlated (the coefficient of the inverse Mills ratio is highly significant). Coefficients are now almost in line with the data-generating process. Comparing the Wald tests, we see that the model has a considerably better fit.

.dhreg y x, hd(z) millr
(output omitted)

second stage results

N = 1000
log likelihood = -937.13325
chi square hurdle equation = 121.026
p hurdle equation = 3.772e-28
chi square above equation = 29.057293
p above equation = 4.901e-07
chi square overall = 127.67061
p overall = 1.891e-28

<table>
<thead>
<tr>
<th>coef</th>
<th>se</th>
<th>z</th>
<th>p</th>
<th>lower CI</th>
<th>upper CI</th>
</tr>
</thead>
<tbody>
<tr>
<td>hurdle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>4.049706</td>
<td>0.3681154</td>
<td>11.00118</td>
<td>0</td>
<td>3.328212</td>
</tr>
<tr>
<td>_cons</td>
<td>-2.002341</td>
<td>0.1479133</td>
<td>-13.53727</td>
<td>9.42e-42</td>
<td>-2.292246</td>
</tr>
<tr>
<td>above</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>0.3243296</td>
<td>0.1560948</td>
<td>2.077773</td>
<td>0.0377303</td>
<td>0.0183894</td>
</tr>
<tr>
<td>_mill</td>
<td>0.5337018</td>
<td>0.1108932</td>
<td>4.812754</td>
<td>1.49e-06</td>
<td>0.3183551</td>
</tr>
<tr>
<td>_cons</td>
<td>0.5127246</td>
<td>0.1099401</td>
<td>4.663672</td>
<td>3.11e-06</td>
<td>0.2972459</td>
</tr>
<tr>
<td>sigma</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_cons</td>
<td>0.8811395</td>
<td>0.0446539</td>
<td>19.73263</td>
<td>0</td>
<td>0.7936194</td>
</tr>
</tbody>
</table>
Our second example is also from simulated data, meant to illustrate the random effects and bootstrap estimators. The data-generating process is as follows:

\[
d_{i}^{*} = \begin{cases} 
1 & \text{if } -2 + 4 \times z_{i} + \varepsilon_{i1} > 0 \\
0 & \text{otherwise}
\end{cases}
\]

\[
y_{it}^{**} = 0.5 + 0.3 \times x_{it} + u_{i} + \varepsilon_{it2}
\]

\[
y_{it}^{*} = \begin{cases} 
y_{it}^{**} & \text{if } y_{it}^{**} > 0 \\
0 & \text{otherwise}
\end{cases}
\]

\[
y_{it} = d_{i}^{*} \times y_{it}^{*}
\]

\[
\varepsilon_{i1} = 0.9 \times u_{i} + \sqrt{1 - 0.9^2} \times \eta_{i}
\]

\[
\begin{pmatrix} \varepsilon_{it2} \\ u_{i} \end{pmatrix} \sim N \left( \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & \sigma^2 \end{pmatrix} \right)
\]

\[
\eta_{i} \sim N(0, \sigma^2)
\]

This dataset is a panel. Individuals \(i\) are observed at multiple points in time \(t\). Their choices are determined by the first hurdle, \(d_{i}^{*}\), which is assumed to be time invariant, and by the second hurdle, \(y_{it}^{*} = 0\), which is allowed to differ by individual and period. In the latent-variable defining choices, provided the first hurdle is passed, there is a random effect \(u_{i}\). The correlation of error terms involves this random effect and not the residual error \(\varepsilon_{it2}\).
The output shows that the estimator finds the effects of both explanatory variables, including the very small effect of \( x_{it} \), and the correlation of the error terms.

\[
\texttt{xtdhreg} \ y \ x, \ hd(z)
\]

\[
\begin{array}{l}
\text{Number of obs} = 10000 \\
\text{Wald chi2(1)} = 43.39 \\
\text{Prob > chi2} = 0.0000 \\
\end{array}
\]

|                | Coef. | Std. Err. | z     | P>|z| | [95% Conf. Interval] |
|----------------|-------|-----------|-------|-------|----------------------|
| hurdle         |       |           |       |       |                      |
| \_cons         | 4.736641 | .719114  | 6.59  | 0.000 | 3.327203 - 6.146079 |
| above          |       |           |       |       |                      |
| x \_cons       | .2585995 | .0530704 | 4.87  | 0.000 | .1545835 - .3626156 |
| sigma_u \_cons | 1.114397 | .0493997 | 22.56 | 0.000 | 1.017575 - 1.211219 |
| sigma_e \_cons | 1.00247  | .0121348 | 82.61 | 0.000 | .9788683 - 1.026254 |
| transformed-o \_cons | .7980986 | .5459896 | 1.46  | 0.144 | -.2720214 - 1.868219 |

generate estimate of correlation in error terms, with confidence interval

\[
\text{rho: tanh([transformed_rho]\_cons)}
\]

|                | Coef. | Std. Err. | z     | P>|z| | [95% Conf. Interval] |
|----------------|-------|-----------|-------|-------|----------------------|
| rho            | .6629725 | .3060095 | 2.17  | 0.030 | .0632049 - 1.26274 |

separate Wald tests for joint significance of all explanatory variables

\[
\begin{array}{l}
\text{chi square hurdle equation} = 43.385582 \\
\text{p hurdle equation} = 4.495e-11 \\
\text{chi square above equation} = 23.743844 \\
\text{p above equation} = 1.100e-06 \\
\text{chi square overall} = 67.117399 \\
\text{p overall} = 2.665e-15 \\
\end{array}
\]

In this case, we know this estimator to be appropriate. All normality assumptions are met, and the data are not nested. Therefore, there is no reason to resort to bootstrapping. Yet in real applications, it may be less obvious that the assumptions underlying a random-effects model are justified. If users instead or additionally run a bootstrap, the output looks as follows:
. bootdhreg y x, hd(z) cluster(i) capt
(output omitted)
maximum likelihood estimates of double hurdle model

N = 10000
log likelihood = -9860.9387
chi square hurdle equation = 9.3724107
p hurdle equation = .00220276
chi square main equation = 9.3724107
p main equation = .00220276
chi square overall = 570.78004
p overall = 1.14e-124
bootstrap results

<table>
<thead>
<tr>
<th></th>
<th>coef</th>
<th>se</th>
<th>p</th>
<th>lowciz</th>
<th>upciz</th>
<th>lowcip</th>
</tr>
</thead>
<tbody>
<tr>
<td>hurdle</td>
<td>z</td>
<td>4.218375</td>
<td>3.43218</td>
<td>.1095236</td>
<td>-2.508698</td>
<td>10.94545</td>
</tr>
<tr>
<td></td>
<td>_cons</td>
<td>-2.180051</td>
<td>1.124344</td>
<td>.0262535</td>
<td>-4.383766</td>
<td>.0236639</td>
</tr>
<tr>
<td>main</td>
<td>x</td>
<td>.243663</td>
<td>.0644732</td>
<td>.0000786</td>
<td>.1172956</td>
<td>.3700305</td>
</tr>
<tr>
<td></td>
<td>_cons</td>
<td>.851831</td>
<td>.3136393</td>
<td>.0033042</td>
<td>.2370979</td>
<td>1.466564</td>
</tr>
<tr>
<td>sigma</td>
<td>_cons</td>
<td>1.370452</td>
<td>.1429195</td>
<td>0</td>
<td>1.09033</td>
<td>1.650574</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>upcip</th>
</tr>
</thead>
<tbody>
<tr>
<td>hurdle</td>
<td>z</td>
</tr>
<tr>
<td></td>
<td>_cons</td>
</tr>
<tr>
<td>main</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>_cons</td>
</tr>
<tr>
<td>sigma</td>
<td>_cons</td>
</tr>
</tbody>
</table>

Bootstrap standard errors are larger than standard errors from the random-effects model. This is as expected. The bootstrap assumes less structure (there is no random effect), and it coarsens the data by the sampling process. The bootstrap routine resamples with replacement. Because the cluster(i) option is used, entire sets of observations per individual are sampled. With our seed (which users are free to change with the seed() option), some bootstrap samples do not converge, which is why the capt option is used. It confines the calculation of the bootstrap standard errors to those (multiple) instances that converge. As is standard in bootstrapping, coefficients are taken from using dhreg on the original data, but standard errors are taken from the bootstrap. The reported bootstrap standard error is the standard deviation of coefficients from all bootstrap instances. The resulting p-value and the lowciz and upciz confidence intervals assume that these results are normally distributed. The two final columns report a distribution-free estimate of the confidence interval. It results from the lower and the upper 2.5% of the empirical distribution of coefficients. If users want to rely on these estimates, they should check whether zero lies outside this interval. If
**Commands to implement double-hurdle regression**

users wish to allow for correlation of the error terms, they can combine `bootdhreg` with the `millr` option.

Additionally, coefficients from all bootstrap repetitions of the statistical model are stored in variables `res`. These data are useful if the user wants to run additional tests, such as a Wald-like test for a net effect. This can be done by generating a new variable that sums up the main effect and the interaction effect. With the `summarize` command, one generates the mean and the standard error of this new variable. \[
\min((1-\text{normal}(r(\text{mean})/r(\text{sd}))), (\text{normal}(r(\text{mean})/r(\text{sd})))) \]
generates the p-value.

We have enabled all estimators to take factor variables. Users can therefore rely on the `i.a##c.b` or on the `c.b##c.b` notation to generate interaction terms and other multiplicative terms, and they can use the `margins` command to derive model predictions, as we show in the example below. With the `margins` command, users can also calculate average marginal effects with the `dydx()` option. Yet for recovering the average marginal change in probability, which tends to be more interesting, the following, somewhat unintuitive, command must be used. It transforms the (average of the) linear marginal effect into a probability (because we use a probit specification for the second equation). In the example, a one-unit change in variable \(z\) (which would be a change from one extreme to the other, given the variable has range 0–1) increases the probability of the first hurdle being passed by almost 97%.

```plaintext
. margins, dydx(z) expression(normal(xb(hurdle)))
Average marginal effects               Number of obs = 1000
Model VCE     : OIM
Expression : normal(xb(hurdle))
dy/dx w.r.t. : z

| Delta-method | dy/dx | Std. Err. | z    | P>|z|  | [95% Conf. Interval] |
|--------------|-------|-----------|------|------|---------------------|
|              | z     | .9675177  | .0294781 | 32.82 | 0.000              | .9097416 1.025294 |
```

### 6 References


---

3. Because there is more than one equation in our model, through `predict(equation(eqname))`, users must specify the equation to which the coefficient in question refers. The hurdle equation is always denoted hurdle, while the equation estimating the dependent variable conditional on the first hurdle being passed is denoted above if a lower hurdle is estimated and below if an upper hurdle is estimated.
C. Engel and P. G. Moffatt


About the authors

Christoph Engel is a director of the Max Planck Institute for Research on Collective Goods, Bonn, Germany.

Peter Moffatt is a professor of econometrics in the School of Economics at the University of East Anglia, Norwich, UK.
The chi-squared goodness-of-fit test for count-data models

Miguel Manjón  
QURE-CREIP Department of Economics  
Rovira i Virgili University  
Reus, Spain  
miguel.manjon@urv.cat

Oscar Martínez  
QURE-CREIP Department of Economics  
Rovira i Virgili University  
Reus, Spain  
oscar.martinez@urv.cat

Abstract. In this article, we discuss the implementation of Andrews’s (1988a, Journal of Econometrics 37: 135–156; 1988b, Econometrica 56: 1419–1453) chi-squared goodness-of-fit test as a postestimation command. The new command chi2gof reports the test statistic, its degrees of freedom, and its p-value. chi2gof can be used after the poisson, nbreg, zip, and zinb commands.

Keywords: st0360, chi2gof, Andrews’s chi-squared goodness-of-fit test, m-tests, count-data models

1 Introduction

In empirical work, one often fits a model using alternative specifications and then concentrates on the coefficient estimates supported by a goodness-of-fit test (thus ignoring estimates not supported by the test). This practice reflects the use of the goodness-of-fit test to detect specification errors in a model. Goodness-of-fit tests can also be used in model comparison and selection. As Cameron and Trivedi (2013, 225) explain, “competing models [...] are compared and evaluated using model diagnostics and goodness-of-fit measures”.

One of these goodness-of-fit tests is the Pearson chi-squared test (see, for example, Cameron and Trivedi [2005, 266] for details). This is implemented in Stata as the postestimation command estat gof following use of the logit, logistic, probit, and poisson commands.1 By typing estat gof after logit, logistic, probit, or poisson, one obtains the $\chi^2$-statistic of the test and its p-value (as well as the number of observations and the number of covariate patterns). Alternatively, the group( #) option results in analogous output for the related Hosmer–Lemeshow test (see Hosmer and Lemeshow [1980]; and Hosmer, Lemeshow, and Sturdivant [2013]).

However, the Pearson and Hosmer–Lemeshow tests assume that the estimated coefficients are known. To control for the potential estimation error, Cameron and Trivedi (2010) suggest using the chi-squared diagnostic test developed by Andrews (1988a,b). This chi-squared goodness-of-fit test generalizes Pearson’s chi-squared test by comparing the sample relative frequencies of the dependent variable with the predicted fre-

---

1. Structural modeling (sem) and survey data (svy:) are other areas of application of this test that are supported by Stata.
frequencies from the model using a quadratic form and an estimate of the asymptotic
variance of the corresponding population moment condition. Unlike Pearson’s test (or
the Hosmer–Lemeshow test), the chi-squared goodness-of-fit test can be constructed
from any regular asymptotically normal estimator of the conditional expectation of the
dependent variable. However, this \( m \)-test is not yet available in Stata.\(^2\)

In this article, we discuss the implementation of the chi-squared goodness-of-fit test
in count-data models as a postestimation command. \texttt{chi2gof} reports the test statistic,
its degrees of freedom, and its \( p \)-value when used after the \texttt{poisson}, \texttt{nbreg}, \texttt{zip}, and
\texttt{zinb} commands. As an option, the command produces a table with the cells, absolute
frequencies, relative frequencies, predicted frequencies, and absolute differences between
actual and predicted frequencies.

2 Statistical basis for the chi-squared goodness-of-fit test

2.1 The chi-squared goodness-of-fit test

Let’s consider a model given by \( f(y|w, \theta) \), with the conditional density of the variable
of interest \( y \) given a set of covariates \( w \) and a vector of parameters \( \theta \).\(^3\) We are
particularly interested in the conditional density of the Poisson, the negative binomial
(NB), the zero-inflated Poisson (ZIP), and the zero-inflated negative binomial (ZINB)
models. Thus \( w = x \) in the Poisson and NB models and \( w = (x, z) \) in the inflated
versions (that is, \( z \) is the set of covariates used in the inflated part of the model). Also
let \( J \) be the number of (mutually exclusive) cells in which the range of the dependent
variable \( y_i \) is partitioned \((i = 1, \ldots, N)\). Finally, let \( d_{ij}(y_i) = 1(y_i \in j) \) be an indicator
variable that takes value 1 if observation \( i \) belongs to cell \( j \) and 0 otherwise.

If the model is correctly specified, then
\[
E\{d_{ij}(y_i) - p_{ij}(w_i, \theta)\} = 0
\] (1)

where \( p_{ij}(w_i, \theta) \) is the probability that observation \( i \) falls in cell \( j \) according to \( f(y|w, \theta) \).
In particular, stacking all \( J \) moments in vector notation, (1) becomes
\[
E\{d_i(y_i) - p_i(w_i, \theta)\} = 0
\]

Given a sample analog
\[
\hat{m}_N (\hat{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \left\{ d_i(y_i) - p_i \left( w_i, \hat{\theta} \right) \right\}
\]

2. According to Cameron and Trivedi (2010, 266), “\( m \)-tests such as conditional moment tests are tests
of whether moment conditions imposed by a model are satisfied” and “are a general specification
testing procedure that encompasses many common specification tests”.
3. This subsection is largely based on Greene (1994) and Cameron and Trivedi (2005, 2013).
The chi-squared goodness-of-fit test for count-data models

the chi-squared goodness-of-fit test statistic of Andrews (1988a,b) is

\[ N \tilde{m}_N^\prime \left( \hat{\theta} \right) \tilde{V}^{-1} \tilde{m}_N \left( \hat{\theta} \right) \]  

(2)

where \( \tilde{V} \) is a variance–covariance matrix given by \( \sqrt{N} \tilde{m}_N(\hat{\theta}) \rightarrow N(0, V) \).

Under the null hypothesis that the moment condition (1) holds, the chi-squared goodness-of-fit test statistic is asymptotically \( \chi^2 \) distributed with rank \( p \tilde{V} q \) degrees of freedom. However, \( \tilde{V} \) may not be of full rank. The rank is usually \( J - 1 \) because the sum of the probabilities over all \( J \) cells is 1. Moreover, the computation of this variance–covariance matrix is often complicated.

This is why, when the maximum likelihood (ML) estimation is used, it is the outer product of the gradient form of the test that is usually computed. This is \( N \) times the (uncentered) \( R^2 \) of the following auxiliary regression,

\[ 1 = \tilde{m}_i \delta + \tilde{s}_i \gamma + u_i \]

where 1 is a column vector of \( N \), \( \tilde{m}_i \) includes \( d_{ij}(y_i) - p_{ij}(w_i, \hat{\theta}^{ML}) \) for \( j = 1, \ldots, J - 1 \) (the last column of \( d_i - p_i \) has been dropped), and \( \tilde{s}_i = \left\{ \nabla \log f(y_i | w_i, \theta) \right\}_{\theta = \hat{\theta}^{ML}} \) is the matrix of contributions to the score evaluated at the ML estimate of \( \theta \). It is easy to see that the test statistic

\[ N \times R^2 = 1^\prime (\tilde{H}) (\tilde{H})^{-1} 1 \]

where \( \tilde{H}_i = [\tilde{m}_i, \tilde{s}_i] \) is the \( i \)th row of matrix \( \tilde{H} \). This asymptotically equivalent version of (2) is used in the \text{chi2gof} command. Under the null hypothesis of correct specification of the model, this statistic asymptotically follows a \( \chi^2 \) distribution with \( J - 1 \) degrees of freedom.

To conclude this section, we provide details of the computation of this test regarding both the predicted probabilities \( (p_{ij}) \) and the scores \( (\tilde{s}_i) \).

2.2 Predicted probabilities

Let \( \mu_i = e^{\alpha \beta} \) be the conditional expectation of the Poisson model. This model predicts that the probability that the variable of interest takes the value \( t \) is

\[ \Pr(y_i = t) = \frac{e^{\mu_i} \mu_i^t}{t!} = P_P(t) \]

Let \( \Gamma(\cdot) \) be the gamma function (see, for example, Cameron and Trivedi [2013, 505–506] for details). The predicted probabilities of the NB model with conditional variance \( \mu + \alpha \mu^2 \) are

\[ \Pr(y_i = t) = \frac{\Gamma(t + \alpha^{-1})}{\Gamma(t + 1) \Gamma(\alpha^{-1})} \left( \frac{1}{1 + \mu_i \alpha} \right)^{\alpha^{-1}} \left( \frac{\mu_i \alpha}{1 + \mu_i \alpha} \right)^t = P_{NB}(t) \]
Also let’s denote the distribution function used in the inflated versions of the Poisson and NB models by $\varphi$. Stata currently supports two functions—the logit and the probit. Thus

\[
\varphi_i = \varphi(z_i, \gamma) = \begin{cases} 
\Lambda(z_i, \gamma) = \frac{e^{z_i \gamma}}{1 + e^{z_i \gamma}} & \text{in the logit case} \\
\Phi(z_i, \gamma) = \int_{-\infty}^{z_i} \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} du & \text{in the probit case}
\end{cases}
\]

Finally, if we denote an indicator function that takes value 1 if the condition in brackets is true and 0 otherwise by $1(\cdot)$, then the predicted probabilities of the ZIP and NB regression models can be, respectively, expressed as follows:

\[
\Pr(y_i = t) = 1(t = 0) \varphi_i + (1 - \varphi_i) P_P(t)
\]

and

\[
\Pr(y_i = t) = 1(t = 0) \varphi_i + (1 - \varphi_i) P_{NB}(t)
\]

### 2.3 Scores

The individual contribution to the likelihood function in the models considered is

\[
f(y_i | x_i, \theta) = g(y_i) = \begin{cases} 
f(y_i | x_i, \beta) = P_P(y_i) & \text{in the Poisson model} \\
f(y_i | x_i, \beta, \alpha) = P_{NB}(y_i) & \text{in the NB model}
\end{cases}
\]

and

\[
f(y_i | x_i, z_i, \theta) = 1(y_i = 0) \varphi_i + (1 - \varphi_i) g(y_i) \text{ in the inflated versions}
\]

Thus the first derivative of the likelihood function in the Poisson model with respect to the parameters of interest, $\theta = \beta$, is

\[
s_i = \frac{\partial \log f(y_i | x_i, \beta)}{\partial \beta} = x_i (y_i - \mu_i)
\]

whereas the first derivative of the likelihood function in the NB model with respect to the parameters of interest, $\theta = (\beta, \alpha)$, is

\[
s_i = \begin{bmatrix} 
\frac{\partial \log f(y_i | x_i, \beta, \alpha)}{\partial \beta} \\
\frac{\partial \log f(y_i | x_i, \beta, \alpha)}{\partial \alpha}
\end{bmatrix} = \begin{bmatrix} 
x_i \left( \frac{y_i - \mu_i}{1 + \alpha \mu_i} \right) \\
\frac{1}{\alpha} \left\{ \log(1 + \mu_i) - \sum_{t=0}^{y_i-1} \frac{1}{1 + \alpha t} \right\} + \frac{y_i - \mu_i}{\alpha (1 + \mu_i)}
\end{bmatrix}
\]
The chi-squared goodness-of-fit test for count-data models

In the inflated versions of these models, \( f(y_i|x_i, z_i, \theta) = f(y_i|x_i, z_i, \beta, \gamma) \) for the Poisson and \( f(y_i|x_i, z_i, \theta) = f(y_i|x_i, z_i, \beta, \gamma, \alpha) \) for the NB. Therefore, the first derivative of the likelihood function with respect to the parameters of interest can be written as

\[
S_i = \begin{bmatrix}
\frac{\partial \log f(y_i|x_i, z_i, \theta)}{\partial \beta} \\
\frac{\partial \log f(y_i|x_i, z_i, \theta)}{\partial \gamma} \\
\frac{\partial \log f(y_i|x_i, z_i, \theta)}{\partial \alpha}
\end{bmatrix} = \begin{bmatrix}
\frac{(1-\alpha_i)}{f(y_i|x_i, z_i, \theta)} \frac{\partial g(y_i)}{\partial \beta} \\
\frac{1}{f(y_i|x_i, z_i, \theta)} \frac{\partial g(y_i)}{\partial \gamma} \\
\frac{(1-\alpha_i)}{f(y_i|x_i, z_i, \theta)} \frac{\partial g(y_i)}{\partial \alpha}
\end{bmatrix}
\]

where

\[
\frac{\partial g(y_i)}{\partial \beta} = \begin{cases} 
P_P(y_i) x_i (y_i - \mu_i) & \text{in the ZIP model} \\
P_{NB}(y_i) x_i \left( \frac{y_i - \mu_i}{1 + \alpha \mu_i} \right) & \text{in the ZINB model}
\end{cases}
\]

\[
\varphi' = \frac{\partial \varphi_i}{\partial \gamma} = \begin{cases} 
z_i \frac{e^{y_i\gamma}}{(1 + e^{y_i\gamma})^2} & \text{in the logit case} \\
z_i \frac{1}{\sqrt{2\pi}} e^{-\frac{(y_i)^2}{2}} & \text{in the probit case}
\end{cases}
\]

and, in the case of the ZINB model,

\[
\frac{\partial g(y_i)}{\partial \alpha} = P_{NB}(y_i) \left[ \frac{1}{\alpha^2} \left( \log(1 + \mu_i) - \sum_{t=0}^{y_i-1} \frac{1}{t + \alpha} \right) + \frac{y_i - \mu_i}{\alpha(1 + \mu_i)} \right]
\]

(Notice that this derivative is not needed in the ZIP model because \( \alpha = 0 \).)

3 The chi2gof command

3.1 Syntax

\texttt{chi2gof, \texttt{cells(numlist)}} [prcount table]

3.2 Options

\texttt{cells(numlist)} specifies a set of ascending integers greater than or equal to zero that determines the (mutually exclusive) cells in which the range of the dependent variable is partitioned to compute the test. \texttt{cells()} is required.

In principle, any partition of the dependent variable can be used (Andrews 1988b). For example, if 3 cells are chosen, the following partitions can be used: \{0, 1, 2, 3\}, \{4, 5\}, and \{6, 7, \ldots, \infty\}; \{0, 1\}, \{2, 3, 4, 5\}, and \{6, 7, \ldots, \infty\}; \{0, 1, 2, 3, 4, 5\}, \{6\}, and \{7, 8, \ldots, \infty\}; etc. Thus \texttt{chi2gof} allows partitions with both single-value elements (except for the last cell) and multiple-value elements. In the first case, \texttt{numlist} is the
number of cells chosen by the user; in the second case, `numlist` is a set of integers that corresponds to the upper limits of the intervals considered.

- Choosing the number of cells involves using partitions like $\{0\}$ and $\{1, 2, 3, \ldots, \infty\}$ when `cells(2)`; $\{0\}$, $\{1\}$, and $\{2, 3, \ldots, \infty\}$ when `cells(3)`; $\{0\}$, $\{1\}$, $\{2\}$, and $\{3, 4, \ldots, \infty\}$ when `cells(4)`; and so on. In general, for `cells(J)`, the partition that `chi2gof` uses is $\{0\}$, $\{1\}$, $\{2\}$, $\{J - 2\}$, and $\{J - 1, \ldots, \infty\}$.

- Choosing the upper limits of the intervals involves using partitions like $\{0, 1\}$, $\{2, 5\}$, and $\{6, \infty\}$ when `cells(1 5)`; $\{0, 3\}$, $\{4, 9\}$, and $\{10, \infty\}$ when `cells(3 4 9)`; and $\{0, 0\}$, $\{1, 1\}$, $\{2, 2\}$, $\{3, 3\}$, and $\{4, \infty\}$ when `cells(0 1 2 3)`. In general, for `cells(a_0, a_1, \ldots, a_{J - 2})`, the partition that `chi2gof` uses is $\{0, a_0\}$, $\{a_0 + 1, a_1\}$, $\ldots$, $\{a_{J - 3} + 1, a_{J - 2}\}$, and $\{a_{J - 2} + 1, \infty\}$.

Notice that `cells(0 1 2 \ldots J - 2)` is equivalent to `cells(J)`. Notice also that to construct the partition, one must select an integer 2 or more for the number of cells $J$. However, the chosen number should prevent cell frequencies from getting too small (Cameron and Trivedi 2005, 2013). Thus users should look at the distribution of the dependent variable to ensure that cells do not have zero or very few observations. Users should also try using alternative values around the number of cells initially chosen.

`prcount` calculates the probability that according to the model, a particular value of the dependent variable belongs to one of the defined cells. By default, the command calculates these predicted probabilities (or predicted frequencies) using the definition of the conditional density of the dependent variable (`direct`). These probabilities can also be computed using the command `prcounts` of Long and Freese (2001). Results are generally the same when using either command. However, differences occur when the number of counts is high, particularly if the ZINB model is used. In this case, an error message results stating “Missing values encountered when prcount option is used (try direct option)”.

`table` produces a table with the absolute and relative frequencies of each defined cells, the mean fitted value of the relative frequencies (that is, the mean value of the predicted probabilities for each individual of each of the defined cell), and the absolute differences between actual and predicted frequencies. This can be useful in assessing the adequacy of the partition of the dependent variable being used. This may help to detect cells with too few observations. Also the table may help identify the source of misspecification. In the Poisson model, for example, big absolute differences in the zero value may indicate overdispersion.

---

4. Notice also that the statistic may not be computed for the ZINB model if the $\alpha$ parameter is too small. If it is, an error message states that a Problem with alpha prevents estimation of predicted probabilities (alpha too small). In practice, this does not happen often because of the use of the `lngamma()` function. Ultimately, both error messages occur because of the large numbers that the `lngamma()` function generates (see section 2).
The chi-squared goodness-of-fit test for count-data models

Note that, as expression (2) shows, we can interpret the absolute differences between actual and predicted frequencies as the approximate contribution of each cell to the chi-squared goodness-of-fit test (the exact contribution being a quadratic form in $V$). Also each cell contributes the absolute differences between actual and predicted frequencies divided by the root of the predicted frequencies (the so-called Pearson residuals) to the chi-squared goodness-of-fit test when $V$ is a diagonal matrix of the predicted frequencies. In this case, the chi-squared goodness-of-fit test becomes Pearson’s chi-squared test. However, this is not the case in the count-data models considered here, nor is it in most regression applications (the multinomial logit model being an exception). This is why Pearson’s residuals are generally not useful when analyzing the chi-squared goodness-of-fit test.

3.3 Stored results

$chi2gof$ stores the following in $r()$:

Scalars

| $r(chi2gof)$ | chi-squared test statistic |
| $r(dof_ch2gof)$ | degrees of freedom |
| $r(p_ch2gof)$ | $p$-value |

4 Examples

In applications, the model should be suspected of being misspecified (that is, the model moment conditions are not satisfied) if the resulting test is statistically significant. Otherwise, there is no evidence of misspecification in the model. We illustrate this using the four examples below, in which we show the use of the new command and the interpretation of its output in different settings.

Given the illustrative purpose of this section, we closely follow the sources of the examples (Cameron and Trivedi 2010, 2013) when describing the data and discussing the possible misspecification of the proposed models. We contribute by merely analyzing the results of the chi-squared goodness-of-fit test. We do not address the reasons behind the possible misspecification of the models.

In the first example, we replicate results from chapter 5 of Cameron and Trivedi (2013). In the second example, we replicate and extend results reported in chapter 6 of Cameron and Trivedi (2013). In the third and fourth examples, we replicate and extend results from chapter 17 of Cameron and Trivedi (2010). For all examples, we report the output resulting from both the estimation command ($poisson$, $nbreg$, $zip$, or $znb$) and the new command ($chi2gof$). In the first and second examples, we also report a table with the cells, absolute frequencies, relative frequencies, predicted frequencies, and absolute differences between actual and predicted frequencies (option $table$).

Our results seem to confirm the original authors’ conclusions in respect to the poor fit of the ZIP and the ZINB models in the second example. In the third example, the NB2 model provides a similar fit (in terms of information criteria) to more complex
models such as the NB2 hurdle and the NB2 with a finite mixture. However, the chi-squared goodness-of-fit test suggests that the NB2 model is misspecified. In the fourth example, our results seem to confirm the authors’ doubts about the NB2 model being outperformed by its inflated version (ZINB).

4.1 Example 1

The first application we consider here is the analysis of the determinants of takeover bids done by Cameron and Trivedi (2013), which uses a sample of 126 U.S. firms taken over between 1978 and 1985. The dependent variable is the number of bids received by the firm after the initial tender offer (numbids), while covariates include defensive actions taken by the management of the firm (leglrest, realrest, finestrest, and whtknght), firm-specific characteristics (bidprem, insthold, size, and sizesq), and intervention by federal regulators (regulatn). The relation between the dependent and explanatory variables is fit using the Poisson regression model.

```
.infile docno weeks numbids takeover bidprem insthold size leglrest realrest
> finest regulatn whtknght sizesq constant using
> http://cameron.econ.ucdavis.edu/racd/racd5.asc
(126 observations read)
.poisson numbids leglrest realrest finest whtknght bidprem insthold size
> sizesq regulatn, nolog
```

```
Poisson regression
Number of obs = 126
LR chi2(9) = 33.25
Prob > chi2 = 0.0001
Log likelihood = -184.94833 Pseudo R2 = 0.0825
```

```
  | Coef.     Std. Err. | z    | P>|z|   | [95% Conf. Interval]
---|---------------------|------|--------|---------------------
numbids | 0.2601464          | 0.1509594 | 1.72 | 0.085  | -0.0357286 - 0.5560213
leglrest | -0.1956597         | 0.1926309 | -1.02 | 0.310  | -0.5732093 - 0.1818899
realrest | 0.0740301          | 0.1926309 | 0.38 | 0.732  | -0.03503452 - 0.0459453
finrest | 0.4813822          | 0.1588698 | 3.03 | 0.002  | 0.170003 - 0.7927613
whtknght | -0.6776968         | 0.3767372 | -1.80 | 0.072  | -1.416087 - 0.0608066
bidprem | 0.3619912          | 0.4243292 | -0.85 | 0.394  | -1.193661 - 0.0696788
insthold | 0.0075693          | 0.0031217 | 2.42 | 0.015  | -0.0031217 - 0.0136878
size | -0.0294392         | 0.1605682 | -0.18 | 0.985  | -0.344147 - 0.2852686
sizesq | 0.1785026          | 0.0600221 | 2.97 | 0.003  | 0.0608614 - 0.2961438
regulatn | -0.9860598         | 0.5339201 | 1.85 | 0.065  | -0.0604044 - 2.032524
_cons```

The chi-squared goodness-of-fit test for count-data models

```
.chi2gof, cells(6) table
Chi-square Goodness-of-Fit Test for Poisson Model:
Chi-square chi2(5) = 48.66
Prob>chi2 = 0.00

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>9</td>
<td>.0714</td>
<td>.2132</td>
<td>.1418</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>63</td>
<td>.5</td>
<td>.2977</td>
<td>.2023</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>31</td>
<td>.246</td>
<td>.2327</td>
<td>.0134</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>12</td>
<td>.0952</td>
<td>.1367</td>
<td>.0414</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>6</td>
<td>.0476</td>
<td>.068</td>
<td>.0204</td>
</tr>
<tr>
<td></td>
<td>5 or more</td>
<td>4</td>
<td>.0397</td>
<td>.0517</td>
<td>.012</td>
</tr>
</tbody>
</table>
```

From these results, reported on pages 185 and 195–196 of their book, Cameron and Trivedi (2013, 196) “[c]onclude that the Poisson is an inadequate fully parametric model, due to its inability to model the relatively few zeros in the sample”. The table with the absolute and relative frequencies of each defined cell, the mean fitted value of the relative frequencies, and the absolute differences between actual and predicted frequencies that we report using the option `table` clarifies this. It is also interesting to note that “none of the earlier diagnostics [they performed], such as residual analysis, detected this weakness of the Poisson estimates” (Cameron and Trivedi 2013, 196).

### 4.2 Example 2

The second application we consider is Cameron and Trivedi’s (2013) analysis of the determinants of the number of recreational boating trips to Lake Somerville, Texas, in 1980 (`trips`). Covariates include a subjective quality index of the facility (`so`), a dummy variable to indicate the practice of water-skiing at the lake (`ski`), the household income of the head of the group (`i`), a dummy variable to indicate whether the user paid a fee (`fc3`), dollar expenditure when visiting Lake Conroe (`c1`), dollar expenditure when visiting Lake Somerville (`c3`), and dollar expenditure when visiting Lake Houston (`c4`). In this analysis, they discuss different models (including finite mixtures and hurdle types of the Poisson and the NB models) and goodness-of-fit measures (the $G^2$ statistic, the pseudo-$R^2$, etc.). However, here we limit the reported results to the Poisson, NB2, ZIP, and ZINB estimates as well as to the chi-squared goodness-of-fit test.
. infile trips so ski i fc3 c1 c3 c4 using http://cameron.econ.ucdavis.edu/racd/racd6d2.asc, clear
(659 observations read)
. poisson trips so ski i fc3 c1 c3 c4, nolog
Poisson regression
Number of obs = 659
LR chi2(7) = 2543.90
Prob > chi2 = 0.0000
Log likelihood = -1529.4313 Pseudo R2 = 0.4540

|        | Coef.  | Std. Err. | z     | P>|z| | [95% Conf. Interval] |
|--------|--------|-----------|-------|------|----------------------|
| trips  |        |           |       |      |                      |
| so     | 0.4717 | 0.0171    | 27.60 | 0.00 | 0.4382291 .5052227 |
| ski    | 0.4182 | 0.0171    | 27.60 | 0.00 | 0.3743491 .4620959 |
| i      | -1.11132 | 0.195885 | -5.68 | 0.00 | -1.497189 -.729304 |
| fc3    | 0.8981652 | 0.0789854 | 11.37 | 0.00 | 0.7433857 1.052974 |
| c1     | -0.0034297 | 0.0031805 | -1.00 | 0.00 | -0.0095405 .0027705 |
| c3     | -0.0452364 | 0.0016803 | -25.47 | 0.00 | -0.0468102 -.0432625 |
| c4     | 0.0361336 | 0.0016803 | 13.34 | 0.00 | 0.0308229 .0414444 |
| _cons  | 0.2649934 | 0.0937224 | 2.83  | 0.00 | 0.0813009 .4486859 |

.chi2gof, cells(6) table
Chi-square Goodness-of-Fit Test for Poisson Model:
Chi-square chi2(5) = 252.57
Prob>chi2 = 0.00

<table>
<thead>
<tr>
<th></th>
<th>Fitted</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>417</td>
<td>.6328</td>
<td>.4196</td>
<td>.2131</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>68</td>
<td>.1032</td>
<td>.2208</td>
<td>.1177</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>38</td>
<td>.0577</td>
<td>.1031</td>
<td>.0454</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>34</td>
<td>.0516</td>
<td>.0617</td>
<td>.0101</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>.0258</td>
<td>.0449</td>
<td>.0191</td>
<td></td>
</tr>
<tr>
<td>5 or more</td>
<td>72</td>
<td>.129</td>
<td>.1499</td>
<td>.0209</td>
<td></td>
</tr>
</tbody>
</table>

.nbreg trips so ski i fc3 c1 c3 c4, nolog
Negative binomial regression
Number of obs = 659
LR chi2(7) = 478.33
Dispersion = mean
Log likelihood = -825.55758 Pseudo R2 = 0.2246

|        | Coef.  | Std. Err. | z     | P>|z| | [95% Conf. Interval] |
|--------|--------|-----------|-------|------|----------------------|
| trips  |        |           |       |      |                      |
| so     | 0.7219999 | 0.0453323 | 15.93 | 0.00 | 0.6331493 .8108497 |
| ski    | 0.6121388 | 0.1504163 | 4.07  | 0.00 | 0.3173282 .9069493 |
| i      | -0.0260589 | 0.0452342 | -0.58 | 0.56 | -0.1147163 .0625986 |
| fc3    | 0.6691677 | 0.3614399 | 1.85  | 0.06 | 0.0392415 1.377577 |
| c1     | 0.0480086 | 0.0159516 | 3.01  | 0.00 | 0.016744 0.0792732 |
| c3     | -0.092691 | 0.0082685 | -11.21 | 0.00 | -0.1083616 -.0764851 |
| c4     | 0.0388357 | 0.0117139 | 3.32  | 0.00 | 0.0185769 .0617945 |
| _cons  | -1.121936 | 0.2208848 | -5.08 | 0.00 | -1.554752 -.6891205 |

/lnalpha .3157293 .1060209 .1079221 1.113972 1.68797

Likelihood-ratio test of alpha=0: chi2bar(0) = 1407.75 Prob>chi2bar = 0.000
The chi-squared goodness-of-fit test for count-data models

. chi2gof, cells(6) table
Chi-square Goodness-of-Fit Test for NegBin Model:
  Chi-square chi2(5) = 23.54
  Prob>chi2 = 0.00

<table>
<thead>
<tr>
<th>Fitted</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>417</td>
<td>.6328</td>
<td>.6419</td>
<td>.0091</td>
</tr>
<tr>
<td>1</td>
<td>68</td>
<td>.1032</td>
<td>.1224</td>
<td>.0192</td>
</tr>
<tr>
<td>2</td>
<td>38</td>
<td>.0577</td>
<td>.0603</td>
<td>.0024</td>
</tr>
<tr>
<td>3</td>
<td>34</td>
<td>.0516</td>
<td>.0303</td>
<td>.0213</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>.0268</td>
<td>.0215</td>
<td>.0043</td>
</tr>
<tr>
<td>5 or more</td>
<td>72</td>
<td>.129</td>
<td>.1336</td>
<td>.0046</td>
</tr>
</tbody>
</table>

. zip trips so ski i fc3 c1 c3 c4, inflate(so ski i fc3 c1 c3 c4) robust nolog
Zero-inflated Poisson regression
  Number of obs = 659
  Nonzero obs  = 242
  Zero obs     = 417

Inflation model = logit
  Wald chi2(7)  = 75.75
  Log pseudolikelihood = -1163.419
  Prob > chi2   = 0.0000

|                      | Robust Coef. | Std. Err. | z     | P>|z| | [95% Conf. Interval] |
|----------------------|--------------|-----------|-------|---|---------------------|
| trips                |              |           |       |   |                     |
| so                   | .0396788     | .0834161  | 0.48  | 0.634 | -.1238138          | .2031715        |
| ski                  | .4691185     | .1763189  | 2.66  | 0.008 | .1235398           | .8146972        |
| i                   | -.0943536    | .0477011  | 1.98  | 0.048 | -.1878461          | -.0008612       |
| fc3                  | .6050712     | .2358399  | 2.57  | 0.010 | .1428335           | 1.067309        |
| c1                   | .0023539     | .0144217  | 0.16  | 0.870 | -.0259121          | .0306199        |
| c3                   | -.0364429    | .0108506  | -3.36 | 0.001 | -.0577096          | -.0151762       |
| c4                   | .0235891     | .0081858  | 2.89  | 0.004 | .0075987           | .0395795        |
| _cons                | 2.113707     | .5032877  | 4.20  | 0.000 | 1.127281           | 3.100133        |
| inflate              |              |           |       |   |                     |
| so                  | -.1.651993   | .2076671  | -7.96 | 0.000 | -.2.059013         | -1.244973       |
| ski                  | .0588168     | .4614636  | 0.13  | 0.899 | -.8456352          | .9632688        |
| i                  | -.0.719113   | .1110972  | -0.65 | 0.517 | -.2896579          | .1458352        |
| fc3                  | -.20.59898   | .6327039  | -32.56 | 0.000 | -21.83905         | -19.3589        |
| c1                  | -.0.0058103  | .0244693  | -0.24 | 0.812 | -.0537693          | .0421486        |
| c3                  | .0723226     | .0208863  | 3.46  | 0.001 | .0313686           | .113269         |
| c4                  | -.0.753998   | .0251974  | -2.99 | 0.003 | -.1247855          | -.0260137       |
| _cons                | 3.558284     | .532032   | 6.69  | 0.000 | 2.815522           | 4.601047        |
### Chi-square Goodness-of-Fit Test for ZIP Model

**Chi-square chi2(5) = 112.39**

**Prob>chi2 = 0.00**

<table>
<thead>
<tr>
<th>Fitted</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>417</td>
<td>.6328</td>
<td>.6354</td>
<td>.0026</td>
</tr>
<tr>
<td>1</td>
<td>68</td>
<td>.1032</td>
<td>.033</td>
<td>.0701</td>
</tr>
<tr>
<td>2</td>
<td>38</td>
<td>.0577</td>
<td>.042</td>
<td>.0156</td>
</tr>
<tr>
<td>3</td>
<td>34</td>
<td>.0516</td>
<td>.0448</td>
<td>.0068</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>.0258</td>
<td>.0431</td>
<td>.0173</td>
</tr>
<tr>
<td>5 or more</td>
<td>72</td>
<td>.129</td>
<td>.2016</td>
<td>.0726</td>
</tr>
</tbody>
</table>

### Zero-inflated negative binomial regression

**Number of obs = 659**

**Nonzero obs = 242**

**Zero obs = 417**

**Wald chi2(7) = 140.80**

**Log pseudolikelihood = -719.3693**

<table>
<thead>
<tr>
<th>Robust</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>trips</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>so</td>
<td>.170791</td>
<td>.0566653</td>
<td>3.01</td>
<td>0.003</td>
</tr>
<tr>
<td>ski</td>
<td>.492453</td>
<td>.1459854</td>
<td>3.37</td>
<td>0.001</td>
</tr>
<tr>
<td>i</td>
<td>-.0688226</td>
<td>.0414572</td>
<td>-1.66</td>
<td>0.097</td>
</tr>
<tr>
<td>fc3</td>
<td>.547295</td>
<td>.2205721</td>
<td>2.48</td>
<td>0.013</td>
</tr>
<tr>
<td>c1</td>
<td>.0399582</td>
<td>.0184614</td>
<td>2.16</td>
<td>0.030</td>
</tr>
<tr>
<td>c3</td>
<td>-.0658707</td>
<td>.0104456</td>
<td>-6.31</td>
<td>0.000</td>
</tr>
<tr>
<td>c4</td>
<td>.0207245</td>
<td>.0113653</td>
<td>1.82</td>
<td>0.068</td>
</tr>
<tr>
<td>_cons</td>
<td>1.091936</td>
<td>.2919291</td>
<td>3.74</td>
<td>0.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>inflate</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>so</td>
<td>-38.63617</td>
<td>2.002604</td>
<td>-19.29</td>
<td>0.000</td>
</tr>
<tr>
<td>ski</td>
<td>-16.06663</td>
<td>1.084321</td>
<td>-14.82</td>
<td>0.000</td>
</tr>
<tr>
<td>i</td>
<td>-.0209069</td>
<td>.3395567</td>
<td>-0.60</td>
<td>0.550</td>
</tr>
<tr>
<td>fc3</td>
<td>-11.42997</td>
<td>2.420961</td>
<td>-4.72</td>
<td>0.000</td>
</tr>
<tr>
<td>c1</td>
<td>-.023586</td>
<td>.0152638</td>
<td>-1.55</td>
<td>0.122</td>
</tr>
<tr>
<td>c3</td>
<td>.0775286</td>
<td>.0214598</td>
<td>3.61</td>
<td>0.000</td>
</tr>
<tr>
<td>c4</td>
<td>-.0628993</td>
<td>.0214707</td>
<td>-2.93</td>
<td>0.003</td>
</tr>
<tr>
<td>_cons</td>
<td>20.97537</td>
<td>2.85067</td>
<td>7.36</td>
<td>0.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/lnalpha</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-.1832683</td>
<td>.1150975</td>
<td>-1.59</td>
<td>0.111</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>alpha</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.8325447</td>
<td>.0968238</td>
<td>.6644104</td>
<td>1.043227</td>
</tr>
</tbody>
</table>
The chi-squared goodness-of-fit test for count-data models

Cameron and Trivedi (2013) initially analyze results from the Poisson and NB2 models. In the Poisson model, they notice that “the chi-squared goodness-of-fit test based on cells for 0, . . . , 4 and 5 or more trips [. . . ] leads to a value of 252.6, much larger than the \( \chi^2(5) \) critical value, [. . . ] indicating a poor fit of the Poisson to the data” (Cameron and Trivedi 2013, 248). In the NB2 model, “[t]he statistic [. . . ] is 23.5. Although this is a substantial improvement on the Poisson, the model is still rejected because the 5% critical value for \( \chi^2(5) \) is 11.07” (Cameron and Trivedi 2013, 248–249). Thus none of these models fit the data well, and other specifications should be considered.

They also state, “Plausible alternatives to the models considered above are hurdle models, zero-inflated models, and finite-mixture models” (Cameron and Trivedi 2013, 250). However, because the `chi2gof` command does not cover either hurdle or finite-mixture models, here we concentrate on zero-inflated models (ZIP and ZINB). In the ZIP model, the chi-squared goodness-of-fit test shows a value much larger than that found in the NB2 model. In the ZINB model, the test indicates a better fit than that of the NB2, but it still rejects the null hypothesis of correct specification of the model.

We also report a table with the cells, absolute frequencies, relative frequencies, predicted frequencies, and absolute differences between actual and predicted frequencies. This partially replicates results reported in table 6.14 in Cameron and Trivedi (2013). We can see that the Poisson model performs poorly, underpredicting zeros and overpredicting positive outcomes. Its inflated version, the ZIP model, does a better job in predicting the zeros, and this substantially improves the fit (the statistic is 112.39). However, it still performs worse than the NB2. Finally, the ZINB yields the lower goodness-of-fit test (the statistic is 18.34) despite not predicting much better than the NB2.

4.3 Example 3

Using data from the U.S. Medical Expenditure Panel Survey for 2003, Cameron and Trivedi (2010) analyze the determinants of the annual number of doctor visits (`docvis`) for a sample of the Medicare population aged 65 and higher. Covariates include having...
private insurance that supplements Medicare (\textit{private}), having public Medicaid insurance for low-income individuals that supplements Medicare (\textit{medicaid}), age (\textit{age}), squared age (\textit{age2}), the years of education (\textit{educyr}), the presence of an activity limitation (\textit{actlim}), and the number of chronic conditions (\textit{totchr}). They estimate the relationship between \textit{docvis} and the covariates by using alternative estimators and specifications. However, again we restrict the analysis to the results from Poisson and \textit{NB2} models.

They first fit a Poisson regression model using the \texttt{ML} estimator, as follows:

\begin{verbatim}
s. use http://www.stata-press.com/data/mus/mus17data, clear
.s. poisson docvis private medicaid age age2 educyr actlim totchr, nolog
Poisson regression
Number of obs = 3677
LR chi2(7) = 4477.98
Prob > chi2 = 0.0000
Log likelihood = -15019.64 Pseudo R2 = 0.1297

Coef. Std. Err. z  P>|z|  [95% Conf. Interval]

private .1422324  .0143311  9.92 0.000  .114144  .1703208
medicaid .0970005  .0189307   5.12 0.000  .0598969  .134104
age .2936722  .0259563  11.31 0.000  .2427988  .3445457
age2 -.0019311  .0001724 -11.20 0.000  -.0022691  -.0015931
educyr .0295562  .001882  15.70 0.000  .0258676  .0332449
actlim .1864213  .014566  12.80 0.000  .1578726  .2149701
totchr .2483898  .0046447  53.48 0.000  .2392864  .2574933
_cons -10.18221  .9720115 -10.48 0.000  -12.08732  -8.277101
\end{verbatim}

\texttt{. chi2gof, cells(5)}

\textbf{Chi-square Goodness-of-Fit Test for Poisson Model:}

\begin{verbatim}
Chi-square chi2(4) = 1011.40
Prob>chi2 = 0.00
\end{verbatim}

Results show that all the explanatory variables are statistically significant and have the expected sign. In particular, \textit{“docvis is increasing in age, education, number of chronic conditions, being limited in activity, and having either type of supplementary health insurance”} (Cameron and Trivedi 2010, 574). However, the likelihood-ratio test reported after \texttt{nbreg} clearly shows that the parameter $\alpha$ is statistically significant. Thus the null hypothesis of equidispersion that the Poisson model implies is rejected by the data.\footnote{Actually, Cameron and Trivedi (2010) use an auxiliary regression between \((y - \hat{\mu})^2 - y \hat{\mu} \) and $\hat{\mu}$ to test for equidispersion.}

\texttt{5. Actually, Cameron and Trivedi (2010) use an auxiliary regression between \((y - \hat{\mu})^2 - y \hat{\mu} \) and $\hat{\mu}$ to test for equidispersion.}
The chi-squared goodness-of-fit test for count-data models

. use http://www.stata-press.com/data/r17/docvis
. nbreg docvis private medicaid age age2 educyr actlim totchr, nolog

Negative binomial regression
Number of obs = 3677
LR chi2(7) = 773.44
Dispersion = mean
Log likelihood = -10589.339 Pseudo R2 = 0.0352

    docvis |      Coef.    Std. Err.      z    P>|z|     [95% Conf. Interval]
---------+---------------------------------------------------------------
    private |   .164093 .033219 4.94 0.000   .098986 .2292001
   medicaid |   .10033 .045421 2.21 0.027    .01131 .18936
     age |   .29413 .060159 4.89 0.000   .17622 .412038
    age2 |  -.00192 .00040 .48 .000  -.00271 -.00114
   educyr |   .02869 .00422 6.79 0.000    .02042 .03697
    actlim |   .18954 .03476 5.45 0.000   .12141 .25767
    totchr |   .27764 .01215 22.86 0.000   .25384 .30145
       _cons |  -10.297 2.24743 -4.58 0.000  -14.702 -5.893
     /lnalpha |   -.4453 .03068  -5 0.000  -.5054 .3851
      alpha |   .64067 .01965 31.71 0.000   .60327 .68036

Likelihood-ratio test of alpha=0:  chibar2(01) = 8860.60 Prob>chibar2 = 0.000

. chi2gof, cells(5)

Chi-square Goodness-of-Fit Test for NegBin Model:
    Chi-square   = 39.72
Prob>chi2     = 0.00

Cameron and Trivedi (2010) then consider alternative models for handling the observed overdispersion, including the NB model, the Poisson and NB hurdle models, and the Poisson and NB finite-mixture models. They also compare their goodness of fit using the Akaike and Bayes criteria. These analyses lead them to conclude “that the NB2 hurdle model provides the best fitting and the most parsimonious specification” (Cameron and Trivedi 2010, 598). Still, the differences in fit between the NB2 hurdle model and the NB or the NB2 finite-mixture model are very small. On this basis, the NB2 model can be chosen to make inferences. The chi-squared goodness-of-fit test suggests, however, that this model is misspecified.

4.4 Example 4

Using the same dataset as in the previous example, Cameron and Trivedi (2010, 600–605) analyze the determinants of the number of emergency room visits by the survey respondent (er). They state, “The full set of explanatory variables in the model was initially the same as that used in the docvis example. However, after some preliminary analysis, this list was reduced to just three health-status variables—age, actlim, and totchr—that appeared to have some predictive power for er” (Cameron and Trivedi 2010, 600).
They first fit a NB model, as follows:

```stata
use http://www.stata-press.com/data/mus/mus17data_z
.nbreg er age actlim totchr, nolog
```

Negative binomial regression                  Number of obs  =      3677
LR chi2(3)         =   225.15
Dispersion        =   mean
Prob > chi2        =   0.0000
Log likelihood    =  -2314.4927  Pseudo R2       =   0.0464

|                | Coef. | Std. Err. |      z  |     P>|z| |  [95% Conf. Interval] |
|----------------|-------|-----------|--------|--------|-----------------------|
| age            |  0.0088528 |  0.0061341 |    1.44 |  0.149 | -0.0031697 - 0.0208754 |
| actlim         |  0.6859572 |  0.0848127 |    8.09 |  0.000 |  0.5197274 - 0.8521869 |
| totchr         |  0.2514885 |  0.0292559 |    8.60 |  0.000 |  0.1941481 - 0.308829  |
| _cons          | -2.799848  |  0.4593974 |   -6.09 |  0.000 | -3.700251 - 1.899446  |

/lnalpha       |  0.4464685 |  0.1091535 |   0.2325315 | 0.6604055 |
| alpha         |  1.562783  |  0.1705834 |   1.26179 | 1.935577 |

Likelihood-ratio test of alpha=0:  chibar2(01) =  237.98 Prob>chibar2 = 0.000
.chi2gof, cells(5)

Chi-square Goodness-of-Fit Test for NegBin Model:
  Chi-square chi2(4)  =  1.84
  Prob>chi2          =  0.76

Only age is not statistically significant in this model. However, because the proportion of zeros is relatively high—“[t]he first four values [...] account for over 99% of the probability mass of er” (Cameron and Trivedi 2010, 600)—they also consider the inflated version of the NB2 model.
The chi-squared goodness-of-fit test for count-data models

. use http://www.stata-press.com/data/mus/mus17data_z
. zinb er age actlim totchr, inflate(age actlim totchr) vuong nolog

Zero-inflated negative binomial regression

Number of obs = 3677
Nonzero obs = 710
Zero obs = 2967

Inflation model = logit
LR chi2(3) = 34.29
Log likelihood = -2304.868 Prob > chi2 = 0.0000

| Coef. Std. Err.  z    P>|z|   [95% Conf. Interval] |
|------------------|------------------|------------------|------------------|
| er               |                  |                  |                  |
| age             | .0035485         | .0076344         | 0.46             | 0.642            | -.0114146 | .0185116 |
| actlim          | .2743106         | .1768941         | 1.55             | 0.121            | -.0723954 | .6210165 |
| totchr          | .1963408         | .0558635         | 3.51             | 0.000            | .0868504 | .3058313 |
| _cons           | -1.822978        | .6515914         | -2.80            | 0.005            | -3.100074 | -.5458825 |
| inflate         |                  |                  |                  |
| age             | -.0236763        | .0284226         | -0.83            | 0.405            | -.0793835 | .0320309 |
| actlim          | -4.22705         | 18.91192         | -0.22            | 0.823            | -.41.29372 | 32.83962 |
| totchr          | -.3471091        | .2052892         | -1.69            | 0.091            | -.7494886 | .0525055 |
| _cons           | 1.846526         | 2.071003         | 0.89             | 0.373            | -2.212565 | 5.905618 |

| /lnalpha        | .1602371         | .235185          | 0.68             | 0.496            | -.3007171 | .6219193 |
| alpha           | 1.173789         | .2760576         | 4.14             | 0.000            | 1.6416962 | 2.685873 |

Vuong test of zinb vs. standard negative binomial: z = 1.99 Pr>z = 0.0233

. chi2gof, cells(5)

Chi-square Goodness-of-Fit Test for ZINegBin Model:

| Chi-square chisq(4) = 6.70
| Prob>chisq = 0.15

To compare both models, Cameron and Trivedi (2010) use penalized log-likelihood-base statistics (the Akaike information criterion and Bayesian information criterion). Interestingly, “[t]his example indicates that having many zeros in the dataset does not automatically mean that a zero-inflated model is necessary. For these data, the ZINB model is only a slight improvement on the NB2 model and is actually no improvement at all if Bayesian information criterion is used as the model-selection criterion” (Cameron and Trivedi 2010, 605). Results of the chi-squared goodness-of-fit test confirm these conclusions because, although none of the models show signs of misspecification, the NB model yields a smaller statistic.

5 Concluding remarks

In this article, we discuss the implementation of the chi-squared goodness-of-fit test of Andrews (1988a,b) as a postestimation command. The new command `chi2gof` reports the test statistic, its degrees of freedom, and its p-value. It also stores these scalars as returned results in `r(chi2gof)`, `r(dof_chi2gof)`, and `r(p_chi2gof)`, respectively. As an option, the command produces a table with the actual, predicted, and absolute differences between actual and predicted frequencies. `chi2gof` can be used after the `poisson`, `nbreg`, `zip`, and `zinb` commands.
This specification test compares the sample relative frequencies of the dependent variable with the predicted frequencies of the model using a quadratic form and an estimate of the asymptotic variance of the corresponding population moment condition. Unlike Pearson’s test (or the Hosmer–Lemeshow test), the chi-squared goodness-of-fit test can be constructed from any regular asymptotically normal estimator of the conditional expectation of the range of the dependent variable. In particular, \texttt{chi2gof} computes the test statistic using the outer product of the gradient form of the test (see Cameron and Trivedi [2005, 2013]).

We illustrate the use of the test in four examples from Cameron and Trivedi (2010, 2013). Under the null hypothesis of correct specification of the model, this statistic asymptotically follows a chi-squared distribution with $J - 1$ degrees of freedom, $J$ being the number of cells in which the dependent variable is partitioned. Thus, in applications, the model should be suspected of being misspecified (that is, the model moment conditions are not satisfied) if the resulting test is statistically significant. Otherwise, there is no evidence of misspecification in the model.

6 References


———. 2010. \textit{Microeconometrics Using Stata}. Rev. ed. College Station, TX: Stata Press.


The chi-squared goodness-of-fit test for count-data models

About the authors

Miguel Manjón and Oscar Martínez are associate professors at the Department of Economics, Rovira i Virgili University (Spain).
txttool: Utilities for text analysis in Stata

Unislawa Williams
Spelman College
Atlanta, GA
uwilliams@spelman.edu

Sean P. Williams
SunTrust Bank
Atlanta, GA
sean.williams.1000@gmail.com

Abstract. This article describes txttool, a command that provides a set of tools for managing free-form text. The command integrates several built-in Stata functions with new text capabilities. These latter functions include a utility to create a bag-of-words representation of text and an implementation of Porter’s (1980, Program: Electronic library and information systems 14: 130–137) word-stemming algorithm. Collectively, these utilities provide a text-processing suite for text mining and other text-based applications in Stata.

Keywords: dm0077, txttool, text mining, Porter stemmer, bag of words, cleaning, stop words, subwords

1 Introduction

Stata users recently introduced new commands for increasingly sophisticated management of text data, in particular, screening and kountry (Belotti and Depalo 2010; Raciborski 2008). As these authors note, while text data can be challenging to analyze, they occur in many applications, including free-form electronic patient records, country identifiers in international relations and economics datasets, and open-ended survey responses. In addition, text mining, or the quantitative analysis of unstructured text data, has received increasing attention in the social sciences (Benoit, Laver, and Mikhaylov 2009; Lowe and Benoit 2013). These applications vary from categorizing and classifying legislative speeches to evaluating Russian military discourse (Laver, Benoit, and Garry 2003; Grimmer and Stewart 2013).

Generally, Stata lacks the text-management utilities to prepare text data for these kinds of applications. Although some steps in the data-preparation phase can be accomplished with built-in commands, using them can be tedious and can lead to errors. For example, removing punctuation, extra white spaces, and special characters from text could involve dozens of lines of code, and the code would need to be modified for different situations. On the other hand, some steps in the data-preparation phase, such as stemming words or representing instances of words as counts in numeric variables, are not available in Stata at all.

The txttool command fills this gap by providing utilities for several text-preparation tasks:

1. Cleaning: removing punctuation, special characters, and extra white spaces from text and converting it to lowercase.
2. Stop-word removal: removing words that occur too frequently to discriminate outcomes or classes (that is, “the”, “of”, “and”, etc.) or user-specified words that lead to better results in a particular context when removed.

3. Substitution: replacing large numbers of individual words to correct misspellings or variations or to insert user-defined categories for words when analytically useful.

4. Stemming: an implementation of Porter’s (1980) stemming program that reduces a word to a stem or a root. For example, the words “programs”, “programmed”, and “programming” all reduce to the stem “program”. The stem is not always a valid word, and it does not need to be. Rather, its purpose is to reduce the overall word count by grouping closely related words into the same stem.

5. Bag of words: converting a string into both a list of the unique words found in the string and a count of each unique word. This allows a variety of quantitative analyses, including discriminate analysis, clustering, and the creation of dictionaries, to represent different outcomes in other variables.

All of these features are described in more detail below. Afterward, extensions with other programs and with Mata are discussed.

2 The txttool command

2.1 Syntax

```
txttool varname [if] [in], {generate(newvar)|replace} [stem stopwords(filename) subwords(filename) bagwords [prefix(string)] noclean nooutput]
```

`varname` is the string variable containing the text to be processed.

2.2 Options

`generate(newvar)` creates a new variable, `newvar`, containing the processed text of `varname`. The `newvar` will be a copy of `varname` that has been stemmed, has had the stop words removed, has had words substituted, or has been cleaned, depending on the other options specified. Either `generate()` or `replace` is required.

`replace` replaces the text in `varname` with text that has been stemmed, has had the stop words removed, has had words substituted, or has been cleaned, depending on the other options specified. Either `generate()` or `replace` is required.

`stem` calls the Porter stemmer implementation to stem all the words in `varname`.

`stopwords(filename)` indicates that the program should remove all instances of words contained in `filename`. The `filename` is a list of words in a text file. Although a list of
frequently used English words is supplied with `txttool`, users can use different lists of stop words in different applications by specifying different filenames. Stop-word lists without punctuation are recommended.

`subwords(filename)` indicates that the program should substitute instances of words in `filename` with another word in `filename`. The `filename` is a tab-delimited text file, where the first column is the word to be replaced and the second column is the substitute text. Users can use different lists of words to substitute in different applications by specifying different filenames. Subword lists without punctuation are recommended.

`bagwords` tells `txttool` to create a bag-of-words representation of the text in `varname`. The bag-of-words representation consists of new variables, one for each unique word in `varname`, with the count of the occurrences of each word. The new variables are named with the convention `prefix_word`, where `prefix` is optionally supplied by the user, and `word` is the unique word in the text. The options `generate()` and `bagwords` can be used together to represent the processed text as one column with word counts.

`prefix(string)` supplies a prefix for the variables created in `bagwords`. The default is `prefix(w_)`. Supplying a prefix will automatically invoke the `bagwords` option. Note that `txttool` does not know what variables will be created before processing the text, so it cannot confirm the absence of variables already named with the specified prefix. Errors will therefore result if the chosen prefix matches an existing variable.

`noclean` specifies that the program should not remove punctuation, extra white spaces, and special characters from `varname`. By default, `txttool` will clean and lowercase `varname`. The `noclean` option is not allowed with `bagwords`. In addition, because the Porter stemmer does not stem punctuation and because the stop-words and subwords lists should not include punctuation, `noclean` should be used with caution.

`nooutput` suppresses the default output. By default, `txttool` reports the total number of words and the count of unique words before and after processing, as well as the time elapsed during processing. The `nooutput` option suppresses this output, which can save some time with large processing tasks.

### 2.3 Remarks

The options are processed in the following order: `noclean`, `subwords()`, `stopwords()`, `stem`, `generate()` or `replace`, and, finally, `bagwords`. Thus the `noclean` option is examined first, and if it is not specified, punctuation and special characters are removed. Then subwords are substituted and stop words removed; the remaining text is stemmed or bagged. The Porter stemmer algorithm does not recognize punctuation or non-English characters, and Stata does not allow variable names with punctuation and non-English characters, so cleaning must precede stemming and bagging.
Because cleaning comes first, the user-defined lists for `subwords()` and `stopwords()` are most effective when they are themselves “cleaned”. Otherwise, they may reintroduce punctuation and other characters after cleaning. The `noclean` option allows users to process `subwords()` and `stopwords()` without first cleaning the original text, if this is required in a particular instance. However, a more effective approach is cleaning the stop-word and subword lists by reading the lists into Stata and processing the lists with `txttool stopwordlist, gen(stopwordlist2)` and `txttool subwordlist, gen(subwordlist2)` to obtain word lists that have had any punctuation and special characters removed. These lists can then be exported as text files to use in later applications with `txttool`.

The program’s default behavior is to remove all characters except white space (American standard code for information interchange [ASCII] code 32), numerals (ASCII codes 48–57), and letters (ASCII codes 97–122 after lowercasing). Therefore, the default behavior is to remove punctuation, non-English characters, and nonprinting characters. Removing these characters is assumed by the Porter stemmer, which was created for English, and necessary for creating new variables of the unique words with the `bagwords` option because Stata does not allow special characters in variable names. While `bagwords` is not allowed with `noclean`, the `txttool` command allows the `stem` option when `noclean` is specified, although the stemmer may not function as expected, so users should examine the results carefully. Analysis of non-English text can still use the `stopwords()`, `subwords()`, `generate()`, and `replace` options even if `noclean` must be specified to accommodate the characters in a particular language.

3 Examples

3.1 Examples of options usage

Imagine we have text data in the form of open-ended answers to a survey on voter attitudes.

```stata
use example_text
describe
txtexample
```

<table>
<thead>
<tr>
<th>texexample</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Unemployment is the major issue, but no one’s talking about it</td>
</tr>
<tr>
<td>2. I’ve been looking and looking but I still can’t find a job</td>
</tr>
<tr>
<td>3. I hear a lot of talk about the economy improving</td>
</tr>
</tbody>
</table>

It is best to show how to prepare the text for analysis step by step. First, cleaning removes punctuation and any special characters and lowercases the text.

```stata
.txttool txtexample, gen(cleaned)
```

```
Input: 28 unique words, 33 total words
Output: 27 unique words, 33 total words
Total time: .453 seconds
```
The researcher may wish to use a preexisting coding scheme and group terms such as “unemployment” and “jobs” into one term such as “employment.” In addition, the researcher may correct common misspellings in a particular context, for example, replacing “ecomony” with “economy” in the third observation. Note also that the subword option can substitute phrases for words so that, for example, contractions in the text can be expanded. The researcher may define the subword list as the following tab-delimited text file:

```
unemployment    employment
job             employment
economy         economy
ive             i have
```

Using this list of substitutions produces the following text:

```
. txttool txtexample, gen(subbed) subwords("subwordexample.txt")
(output omitted)
. list subbed

<table>
<thead>
<tr>
<th>subbed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. employment is the major issue but no ones talking about it</td>
</tr>
<tr>
<td>2. i have been looking and looking but i still cant find a employment</td>
</tr>
<tr>
<td>3. i hear a lot of talk about the economy improving</td>
</tr>
</tbody>
</table>
```

The stopwords() option will remove user-defined words. The txttool program is packaged with a list of common English words and contractions to be removed. Common words such as “I”, “of”, etc., generally have little discriminating power but increase memory requirements. Using the packaged list of stop words produces

```
. txttool txtexample, gen(stopped) subwords("subwordexample.txt")
> stopwords("stopwordexample.txt")
(output omitted)
. list stopped

<table>
<thead>
<tr>
<th>stopped</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. employment major issue talking</td>
</tr>
<tr>
<td>2. looking looking still find employment</td>
</tr>
<tr>
<td>3. hear lot talk economy improving</td>
</tr>
</tbody>
</table>
```
Stemming the text further reduces the words by removing all but the word stems.

```stata
.txttool txtexample, gen(stemmed) subwords("subwordexample.txt")
> stopwords("stopwordexample.txt") stem
(output omitted)
.list stemmed

<table>
<thead>
<tr>
<th>stemmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>employ</td>
</tr>
<tr>
<td>major</td>
</tr>
<tr>
<td>issu</td>
</tr>
<tr>
<td>talk</td>
</tr>
<tr>
<td>look</td>
</tr>
<tr>
<td>still</td>
</tr>
<tr>
<td>find</td>
</tr>
<tr>
<td>employ</td>
</tr>
</tbody>
</table>
```

Note that “talking” and “employment” are now reduced to “talk” and “employ”, respectively. Finally, the text is “bagged”, and a variable with the prefix “w_” is produced for each remaining word, along with the count of the word in each line of text.

```stata
.txttool txtexample, gen(bagged) subwords("subwordexample.txt")
> stopwords("stopwordexample.txt") stem bagwords prefix(w_)
.list w_*

<table>
<thead>
<tr>
<th>w_employ</th>
<th>w_major</th>
<th>w_issu</th>
<th>w_talk</th>
<th>w_look</th>
<th>w_still</th>
<th>w_find</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>w_hear</th>
<th>w_lot</th>
<th>w_econ</th>
<th>w_improv</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```

### 3.2 Creating a dictionary

One practical text-mining task is creating a dictionary for specific values of a variable. A dictionary is a list of words, phrases, parts of speech, or other tokens that distinguish one value of a variable from another. In a customer-relationship management application, for example, a researcher may have data on products that customers returned as well as customer descriptions of the products. Using the customer descriptions, the researcher can find the words or phrases that distinguish returned and unreturned products. The list of words can then be studied to see what aspects of product design or manufacture lead to returns; it can also be used to “tag” future customer descriptions to track trends or used in a predictive model to classify large numbers of descriptions of competitors’ products. Similar applications can be made to any text data with an associated outcome of interest.
The bagwords option makes it easy to create dictionaries of this kind. To illustrate, we will use the International Monetary Fund’s monitoring of fund arrangements data on loan terms. The data include the International Monetary Fund’s description of the conditions that a borrowing government must meet to maintain funding as well as a status variable that describes whether the country met the condition. We are specifically interested in the kinds of loan conditions that borrowers have the most difficulty meeting, which is a task for dictionary creation.\footnote{The data are available at \url{http://www.imf.org/external/np/pdr/mona/index.aspx}.} We set up the data as follows:

```
. import delimited using "Mona.csv", clear
   (20 vars, 6549 obs)
. keep if status=="M" | status=="NM"
   (2521 observations deleted)
. set seed 1234
. sample 500, count
   (3528 observations deleted)
```

To begin, we assess how many unique words and total words are contained in the description of the loan conditions, named `descpt`.

```
. txttool descpt, gen(test1)
   Input: 2798 unique words, 9706 total words
   Output: 2095 unique words, 9707 total words
   Total time: .896 seconds
```

After cleaning, 2,095 unique words remain out of 9,707 total words in only 500 descriptions of loan conditions. This is obviously a large number of words to parse by reading. We use the example list of packaged stop words with txttool to reduce the word count.

```
. txttool descpt, gen(test2) stopword("stopwordexample.txt")
   Input: 2798 unique words, 9706 total words
   Output: 2021 unique words, 6188 total words
   Total time: .996 seconds
```

Adding the stem option further reduces the word count.

```
. txttool descpt, gen(test3) stopword("stopwordexample.txt") stem
   Input: 2798 unique words, 9706 total words
   Output: 1552 unique words, 6188 total words
   Total time: 1.35 seconds
```

Thus, of the original 2,798 unique words, 703, or 25%, were removed (or made nonunique) through cleaning; another 74, or 3%, were removed with stop-word removal; and another 469, or 17%, were removed through stemming. Though greatly reduced, the new text description is still too large for manual parsing. By bagging the words, we create 1,552 new variables, one for each unique word, and we can turn to numeric methods.

```
. txttool descpt, gen(descpt2) stopword("stopwordexample.txt") stem bagwords
   > prefix(w_)
   (output omitted)
```
The next step is determining which of these words best distinguish conditions that were met \((\text{status}=="M")\) from those that were not met \((\text{status}=="NM")\). Although there are many ways to determine this, a simple approach is to use correlations between each word and the status variable. A loop through the word counts can list the words with particularly high or statistically significant correlations. Given the tabular nature of the data, we use tau-b correlations. Also we want to select only words that occur frequently enough that they can potentially describe more than one instance of meeting or failing to meet a condition, so we choose words that occur in at least 5% of the total words. The loop then outputs the words and the correlations.

\begin{verbatim}
    . generate status_numeric = (status=="M")
    . quietly foreach x of varlist w_* {
        2. summarize `x´, meanonly
        3. if r(mean) > .05 {
            4. tab `x´ status_numeric, all
            5. if abs(r(taub)) >.05 {
                6. noisily display "x" r(taub)
            }
        }
    }
\end{verbatim}

Note that positive correlations are associated more often with met conditions, while negative correlations distinguish the unmet conditions.

\begin{verbatim}
    v_budget .08594184
    v_plan .06356041
    v_law .05894236
    v_fund-.07997933
    v_new-.06189476
    v_account-.05247162
    v_ministri .05705297
    v_bank-.06719572
    v_includ-.06220605
    v_implement-.07452835
    v_adopt-.05802378
\end{verbatim}

Some of the terms uncovered by the procedure, such as “budget”, “plan”, and “law”, are especially associated with met rather than unmet conditions. We can understand why by inspecting a few of the loan conditions with a particular word:

\begin{verbatim}
    . list descpt if w_law>0
\end{verbatim}

\begin{verbatim}
<table>
<thead>
<tr>
<th>descpt</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Revocation of amendments to Article 5(3) of the Anti-Money Laundering..</td>
</tr>
<tr>
<td>26. Fiscal impact assessments evaluating the budgetary impact of all new ..</td>
</tr>
<tr>
<td>28. Passage by Parliament of a new Law on Labor Relations</td>
</tr>
</tbody>
</table>
\end{verbatim}

(output omitted)
On the other hand, several terms stand out for distinguishing conditions that are not met as often, especially those involving funding of programs (“fund”), accounting (“account”), central banking (“bank”), and following through on new programs (“implement”, “adopt”). In fact, creating anything “new” seems to be a hard condition to meet, given the word’s relatively high correlation with NM status. The means show that the word “new” appears almost twice as often in conditions that were not met versus those that were met.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Obs</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>w_new</td>
<td>419</td>
<td>.0692124</td>
<td>.2633645</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Obs</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>w_new</td>
<td>81</td>
<td>.1111111</td>
<td>.3162278</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

While we investigate the findings provided through the dictionary, we need to determine how well the dictionary distinguishes met and unmet conditions. To do so, we can use a simple linear discriminant analysis.

```
. discrim lda w_budget w_plan w_law w_fund w_new w_account w_ministri w_bank > w_includ w_implement w_adopt, group(status_numeric)
```

Linear discriminant analysis
Resubstitution classification summary

<table>
<thead>
<tr>
<th>Key</th>
<th>Number</th>
<th>Percent</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>True status_numeric</th>
<th>Classified</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>38</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>46.91</td>
<td>53.09</td>
</tr>
<tr>
<td>1</td>
<td>105</td>
<td>314</td>
</tr>
<tr>
<td></td>
<td>25.06</td>
<td>74.94</td>
</tr>
<tr>
<td>Total</td>
<td>143</td>
<td>357</td>
</tr>
<tr>
<td></td>
<td>28.60</td>
<td>71.40</td>
</tr>
</tbody>
</table>

Priors 0.5000

The results show that the dictionary does a respectable job at discriminating the outcomes, including finding 38 of the 81 cases of unmet conditions. In further research, loan conditions can be tagged with the dictionary words to score their difficulty, and they can be used in more comprehensive models of a borrower’s conformance that could include economic conditions, political relations, and other factors.
Different methods of identifying potentially distinguishing words, such as different limits on the minimum mean, different thresholds for significance, different correlations, and so on, will likely produce different results. Producing a workable dictionary requires several trials using different methods and validations before a researcher can find the best balance of discriminatory power, number of words, and insight. However, these trials are made easier with tools that appropriately reduce and represent the text.

4 Extensions

4.1 Extension with screening

Two commands, kountry and screening (Belotti and Depalo 2010; Raciborski 2008), offer powerful text data-management capabilities by performing the tagging operation previously described; that is, they apply a coding scheme against text variables. The articles describe examples of using this procedure to standardize country codes (in the case of kountry) or to apply standardized medical codes to electronic patient records (in the case of screening).

These commands are natural extensions of the dictionary-creation capabilities created by txttool. The bagwords option can be used to apply a dictionary and create it by simply counting the words in the dictionary found in the bagwords counts. However, with large datasets, bagging the words can bring substantial computational overhead to an otherwise straightforward problem of counting the occurrences of particular words.

The screening program provides a very convenient way to address the same problem without the additional computational overhead. Returning to the example in section 3.2, we identified a list of five words that were positively correlated with status (those that identified loan conditions that were easier for countries to meet) and six words that were negatively correlated with status (those conditions that were more difficult to meet). First, we prepare the data:

```
. import delimited using "Mona.csv", clear
(20 vars, 6549 obs)
. keep if status=="M" | status=="NM"
(2521 observations deleted)
. generate status_numeric = (status=="M")
. txttool descpt, gen(descpt2) stopword("stopwordexample.txt") stem
```

We can then quickly apply the dictionary from the previous example with the screening command:

```
. screening, sources(descpt2) keys(fund new account bank implement adopt)
> cases(negcases)
. screening, sources(descpt2) keys(budget plan law ministri includ)
> cases(poscases)
```
We then score the loan descriptions in the data and tabulate the results:

```
. egen totneg=rowtotal(negcases*)
. egen totpos=rowtotal(poscases*)
. summarize status_numeric if totpos>totneg

  Variable | Obs   | Mean  | Std. Dev. | Min  | Max  
------|-------|-------|-----------|------|------
status_num-c | 926   | .8650108 | .3418967 | 0    | 1    

. summarize status_numeric if totpos<totneg

  Variable | Obs   | Mean  | Std. Dev. | Min  | Max  
------|-------|-------|-----------|------|------
status_num-c | 1147  | .81517 | .3883289 | 0    | 1    
```

The results indicate that our dictionary distinguishes met and unmet loan conditions on the larger dataset. A logit model indicates that the counts of both positive and negative words are statistically significant.

```
. logit status_numeric totneg totpos, nolog

Logistic regression
Number of obs = 4028
LR chi2(2)    =  9.14
Prob > chi2   =  0.0104
Log likelihood = -1785.4032 Pseudo R2 =  0.0026

  status_num-c | Coef.  | Std. Err. | z     | P> |z| [95% Conf. Interval] 
-------------|-------|-----------|------|----|------------------------
totneg       | -.1403654 | .0575889 | -2.44| 0.015 | -.2532375 | -.0274932 
totpos       | .1348363 | .0675538 | 2.00 | 0.046 | .0024333 | .2672393 
_cons        | 1.655704  | .0602519 | 27.48| 0.000 | 1.537613 | 1.773796
```

This example shows that `screening` can complement `txttool` by quickly applying dictionaries and scoring text data. But `txttool` can also complement the `screening` command, which can remove some special characters and matching keywords with varying numbers of letters (that is, not matching on the entirety of the keyword) to allow for varying spellings and word endings. The cleaning routine in `txttool`, on the other hand, is more comprehensive because it removes all special characters.

Furthermore, `stemming` can make `screening`’s matching keys more efficient by not requiring fewer letters, thus matching the same words without introducing false positives with fewer letters. For example, to match the keyword “flag” with four letters, `screening` will match instances of “flags” and “flagged” but will also introduce potential false positives by matching instances of “flagpole”, “flagellate”, and “flagellum”. The words “flag”, “flags”, and “flagged” all stem to the single stem “flag”, while “flagpole”, “flagellate”, and “flagellum” do not. Therefore, using `txttool` to clean, stem, and remove stop words from the text first can increase the accuracy and ease of matching.
4.2 Extension to Mata

Before version 13, Stata’s limit on the size of text variables placed some restrictions on the types of text data that could be analyzed. However, Mata has no such limits on the size of text and is therefore much more suitable for analyzing larger text data, including longer open-ended survey responses, comment data, or even entire documents.

The options of txttool are written in Mata and can be used interactively in Mata to analyze larger text fields. The options are available in the following Mata routines:

<table>
<thead>
<tr>
<th>Option</th>
<th>Mata command</th>
</tr>
</thead>
<tbody>
<tr>
<td>clean</td>
<td>cleantxt(txtfield)</td>
</tr>
<tr>
<td>stem</td>
<td>stemcolumn(txtfield)</td>
</tr>
<tr>
<td>stopwords()</td>
<td>stopwords(txtfield, filename)</td>
</tr>
<tr>
<td>subwords()</td>
<td>subwords(txtfield, filename)</td>
</tr>
<tr>
<td>bagwords</td>
<td>wordbag(txtfield, prefix, touse)</td>
</tr>
</tbody>
</table>

Where txtfield is a column vector of text data, filename is a string scalar indicating the filename of a stop-word or subword list; prefix is a string scalar indicating the prefix attached to the word count variables created by bagwords; and touse is a selection vector designating which observations to write to. In addition, the routine porterstem(string) can be used to stem words interactively or in user-written routines. The porterstem() function returns only the stem of the string used as an argument; for example, porterstem(articles) returns articl.

5 Conclusion

This article introduced the txttool command, a text data-management suite that integrates native Stata functionality for removing characters and substituting words into a simple command to clean text and remove or substitute unwanted words. txttool also adds an implementation of Porter’s stemming algorithm to reduce words to more useful stems and as an option for creating variables to represent the counts of individual unique words. The command is a useful foundation for text-mining tasks such as creating dictionaries and predictive models based on word frequencies. In addition, the command extends, and is extended by, the functionality of other text-management commands such as screening.
6 References


About the authors

Unislawa Williams is an assistant professor of political science at Spelman College, Atlanta, GA. Her research interests include international relations and forecasting.

Sean P. Williams is senior vice president at SunTrust Bank.
Abstract. This article describes the user-written command \texttt{iop} to estimate ex-ante inequality of opportunity for different types of variables. Inequality of opportunity is the part of inequality that is due to circumstances beyond the control of the individual. Therefore, it is the ethically offensive part of inequality. Several estimation procedures have been proposed over the past years, and \texttt{iop} is a comprehensive and easy-to-use command that implements many of them. It handles continuous, dichotomous, and ordered variables. In addition to the point estimates, \texttt{iop} also provides bootstrap standard errors and two decomposition methods.

Keywords: st0361, iop, inequality of opportunity, dissimilarity index, mean log deviation, decomposition

1 Introduction

The concept of inequality of opportunity has received much attention in development economics over the last decade. In his seminal contribution, Roemer (1998) proposed to divide total inequality into inequality due to different effort levels, to luck, and to different opportunities. The idea is that not all types of inequalities are equally bad. Checchi and Peragine (2010) call the part of inequality that is due to different levels of effort the ethically nonoffensive inequality. Different effort should lead to different outcomes; thus inequality due to different levels of effort might be desirable. In contrast, the ethically offensive part of inequality is the part that is due to circumstances beyond the control of individuals. These circumstances are factors that people cannot change through effort and that affect their outcome. Typical examples for circumstances include gender, race, and family background. Hence, in a situation of perfect equality of opportunities, circumstances should not affect the outcome of individuals. Let us use a school exam as an example. If students get different grades because they studied for different amounts of time, we consider the inequality in the grade as something desirable. If the differences in grades were due to only family background and not to different levels of effort, the same inequality would be considered as ethically offensive. Therefore, the goal is to split total inequality into ethically offensive and nonoffensive parts.

We distinguish ex-ante and ex-post inequality of opportunity (Fleurbaey and Peragine 2013). Ex-ante equality of opportunity is achieved when circumstances do not matter for the outcome. The ex-post approach focuses more on effort and states that
equality of opportunity is achieved when all people making the same degree of effort achieve the same outcome independently of their circumstances. While the two approaches seem to differ only marginally at first, there are important differences that sometimes make them incompatible.\footnote{Fleurbaey and Peragine (2013) discuss the differences in detail and provide conditions in which the two approaches are incompatible.}

Conceptually, the two approaches are equally valid, and it is hard to favor one over the other. However, empirically, the ex-ante approach is easier to implement than the ex-post approach. For both approaches, the main challenge is that both effort and luck are not observable; therefore, it is difficult to distinguish them empirically. While the ex-post approach requires at least an estimate of effort, the ex-ante approach does not and can be estimated without it. This is likely to be the main reason why empirical applications focus mostly on ex-ante inequality of opportunity, given that estimating effort requires very strong assumptions. We follow the empirical applications and focus on ex-ante inequality of opportunity.

Researchers have proposed several methods to assess ex-ante inequality of opportunity over the years. The regression approach became very popular and was widely used for studies in different countries and for different outcomes. The main idea of this method is to relate outcome to circumstances by parametric or nonparametric regression methods. The intuition is that in a world of equal opportunities, the circumstances should not matter, so the regression should have a low fit. If we find that circumstances affect the outcome, we consequently have inequality of opportunity. A weakness of this approach is that it provides only lower-bound estimates of inequality of opportunity. This is primarily because the part of inequality due to unobserved circumstances might be wrongly attributed to effort and luck instead of to inequality of opportunity. Ramos and Van de gaer (2012) discuss this approach in detail and provide additional reasons why it yields lower-bound estimates of inequality of opportunity.

In this article, we describe the user-written command \texttt{iop}, which implements several recently proposed methods to estimate ex-ante inequality of opportunity. \texttt{iop} can estimate inequality of opportunity for both continuous and dichotomous variables. Moreover, by dichotomizing ordered variables at every possible level and applying the methods for dichotomous variables, \texttt{iop} can also handle ordered variables. We focus on two methods proposed by Ferreira and Gignoux (2011) and Ferreira and Gignoux (2014) for continuous variables. For dichotomous outcome variables, we implement the method proposed by Paes de Barros, de Carvalho, and Franco (2007) and a translation invariant version of it suggested by Soloaga and Wendelspiess Chávez Juárez (2013b). Focusing on these methods is justified by their practical use in recent empirical applications and their ability to be used to estimate other methods. For instance, by including dummy variables for each type\footnote{A “type” is defined by a combination of circumstances. Thus all circumstances are identical within a type.} and applying the method proposed by Ferreira and Gignoux (2011), we get the results proposed by Checchi and Peragine (2010).
In addition to the point estimates of inequality of opportunity, we propose and implement two decomposition methods. First, total inequality of opportunity can be decomposed according to different circumstances by using the Shapley decomposition. The purpose of this decomposition is to understand which circumstances drive inequality of opportunity. This decomposition allows the user to understand how much all circumstances affect inequality and how much each circumstance contributes to total inequality of opportunity. Second, \texttt{iop} proposes an Oaxaca-type decomposition of the difference between two groups in a composition and a coefficient effect. This decomposition is used to analyze differences in the level of inequality of opportunity between two geographical units or between the same unit at different times. The Oaxaca decomposition identifies what part of the observed differences is due to differences in the distributions of circumstances and what part is due to differences in the impact of circumstances on the outcome variable.

In this article, we first introduce the regression approach to the measurement of inequality of opportunity in section 2. We then present the command \texttt{iop} in section 3 and include examples using the Programme for International Student Assessment (PISA) data in section 4. In the conclusion, we address some limitations and issues of the command and provide an outlook on future developments.

2 Methods

2.1 The regression approach

There are different approaches to assess inequality of opportunity. The regression approach comprises many approaches, which all assess ex-ante inequality of opportunity. To discuss this family of methods, we first introduce some notation. Let $y$ be the outcome variable of interest and $C$ be a matrix of circumstances beyond the control of the individual. The core element of these methods is to relate the outcome to the vector of circumstances. In general, we can describe this by the expected conditional outcome

$$\hat{y} = E(y | C)$$

which can be estimated in different ways according to the research question and the dependent variable. For instance, Paes de Barros, de Carvalho, and Franco (2007) have a binary outcome variable (for example, access to schooling) and use a logit or probit model to estimate (1). Ferreira and Gignoux (2011) use income as a dependent variable and estimate the same equation with an ordinary-least squares (OLS) regression and with nonparametric methods by averaging over types. Checchi and Peragine (2010) also estimate inequality of opportunity for income and perform a similar analysis but use only nonparametric estimation techniques to assess (1). Finally, Ferreira and Gignoux (2014) use linear regression for test scores.
Independently of the way (1) is estimated, inequality of opportunity is then computed using a common inequality measure $I(\cdot)$ applied to $\hat{y}$:

$$\theta_a = I(\hat{y})$$

The idea behind this is simple. All variation in the vector $\hat{y}$ is exclusively due to circumstances; hence, it refers to inequality of opportunity. The best choice of the appropriate inequality measure depends on the scope of the analysis and on the dependent variable. Paes de Barros, de Carvalho, and Franco (2007) use the dissimilarity index, Ferreira and Gignoux (2011) use the mean logarithmic deviation, and Ferreira and Gignoux (2014) use the variance. Dividing the absolute inequality measure by the same metric $I(\cdot)$ applied to the actual outcome $y$ gives a relative measure of inequality of opportunity:

$$\theta_r = \frac{I(\hat{y})}{I(y)}$$

This last step is possible only when the inequality measure $I(\cdot)$ is equally defined for $\hat{y}$ and $y$. For example, this is not the case when the actual outcome is binary and $\hat{y}$ is the estimated probability.

The choice of the appropriate inequality measure $I(\cdot)$ is crucial and depends mainly on the outcome variable. Table 1 provides an overview of different measures proposed in the literature and implemented in the command \texttt{iop}. 


<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>Income</td>
<td>PISA score</td>
<td>Access to schooling</td>
<td>Access to schooling</td>
</tr>
<tr>
<td>Method to estimate $E(y</td>
<td>C)$</td>
<td>OLS</td>
<td>OLS</td>
<td>Probit or logit</td>
</tr>
<tr>
<td>Inequality measure $I(y)$</td>
<td>Mean log deviation: $\frac{1}{N} \sum_{i=1}^{N} \ln(\frac{y_i}{\bar{y}})$</td>
<td>Variance: $\frac{1}{N} \sum_{i=1}^{N} (y_i - \bar{y})^2$</td>
<td>Dissimilarity index: $\frac{1}{N} \sum_{i=1}^{N}</td>
<td>y_i - \bar{y}</td>
</tr>
<tr>
<td>Absolute measure $\delta_a$</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Relative measure $\delta_r$</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Translation invariant</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Scale invariant</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Abbreviation used in $\text{iop}$</td>
<td>fg1a or fg1r</td>
<td>fg2r</td>
<td>pdb</td>
<td>vs</td>
</tr>
</tbody>
</table>
The methods presented in table 1 are for continuous and dichotomous variables. However, by dichotomizing ordered variables at each possible level, one can also apply the latter two methods to ordered variables. The methods are mainly different in terms of properties. For the continuous case, Ferreira and Gignoux (2011) use a method that is particularly well suited for variables such as income, which has an inherent scale. For example, income is naturally defined from zero to infinity. However, sometimes, the continuous variable has no such natural points. For instance, student test scores can be translated and rescaled without losing the sense of the variable. Here the method proposed by Ferreira and Gignoux (2014) should be preferred because their measure is both translation and scale invariant, while the former method is only scale invariant.

With respect to dichotomous variables, two methods are proposed: one method ensures scale invariance, and the other method ensures translation invariance. Paes de Barros, de Carvalho, and Franco (2007) use a logit or probit model to estimate the conditional probability and to apply the dissimilarity index. This method ensures scale invariance of the inequality of opportunity measure, but it is sensitive to translation. It is used, for instance, to compute the Human Opportunity Index introduced by the World Bank and explained in Paes de Barros et al. (2009). Soloaga and Wendelspiess Chávez Juárez (2013) apply a variation of this method that focuses on translation invariance of the measure instead of on scale invariance.

We focus on these four references for two reasons. First, these methods have been used the most in recent empirical work. Second, these methods are members of a larger family of methods, and they allow the user to also estimate some related approaches. For instance, by creating dummies for each type and using them as circumstances, we get the nonparametric estimator proposed by Checchi and Peragine (2010). Moreover, iop can handle other non- or semiparametric methods, such as splines.

2.2 Decompositions of the inequality of opportunity measure

The regression approach provides us with a point estimate of absolute or relative inequality of opportunity. However, to fully understand the phenomenon of inequality of opportunity and its evolution, one may want to further decompose the measure. There are two interesting decompositions. First, we can decompose inequality of opportunity in a given country into its sources by estimating the relative importance of each circumstance. This decomposition is based on the Shapley value. Second, we can decompose the difference in inequality of opportunity between two populations. For example, the different populations can refer to different countries or to the same country in two points of time, by gender, etc. This Oaxaca decomposition allows us to distinguish which part of the difference is due to different distributions of circumstances and which part is due to differences in how the circumstances affect the outcome. We will now discuss the two decomposition methods in more detail.

4. To estimate the Human Opportunity Index, Stata users can download the command hoi (Azevedo et al. 2010).
The Shapley decomposition

The measure of total inequality of opportunity can be divided into its components, attributing a part of total inequality to each circumstance. We will use the Shapley decomposition. To compute the Shapley decomposition, we first estimate the inequality measure for all possible permutations of the circumstance variables. We then compute the average marginal effect of each circumstance variable on the measure of inequality of opportunity. This procedure is computationally intensive because $2^K$ ($K =$ number of circumstances) must be computed. However, the Shapley decomposition has substantial advantages over other decomposition methods. First, the decomposition is order independent, and second, the different components equal the total value.

As a note of caution, Ferreira and Gignoux (2014) argue that such decomposition should not be seen as causal and can give only an idea of the relative importance. This is because most circumstances are highly correlated, so the coefficients might suffer from multicollinearity. This multicollinearity is a problem for the decomposition but not for the point estimates of inequality of opportunity.

Group decomposition in the spirit of Oaxaca

A second decomposition of inequality of opportunity that might be of interest is the decomposition in subgroups, for instance, women and men. To have inequality of opportunity, two conditions must be satisfied: people have to differ in circumstances (the composition effect), and these circumstances must affect the outcome. Therefore, differences in inequality of opportunity can be based on differences in the circumstances (composition effect) or on differences in the impact of circumstances on the outcome (association effect). We propose this decomposition by computing the inequality of opportunity for each group individually and then by computing counterfactual inequality of opportunity measures. For instance, this is done by computing the level of inequality of opportunity of women by using the returns to circumstances (the estimated regression coefficients) of men. All differences between the true value for women and this counterfactual measure are attributable to differences in the circumstances (composition effects). Table 2 shows the four possible inequality measures that can be computed for the two groups.

<table>
<thead>
<tr>
<th>Table 2. Group decomposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution</td>
</tr>
<tr>
<td>Men</td>
</tr>
<tr>
<td>Women</td>
</tr>
</tbody>
</table>
On the diagonal (upper left to lower right), we have the actual inequality of opportunity estimates for both genders. The upper-right value is the counterfactual estimate using the coefficients of women and the composition of circumstances of men. The lower-left value is based on men’s coefficients and women’s circumstances. This decomposition approach has been used, for instance, by Contreras et al. (2012) in a study on inequality of opportunity in Chile. Besides comparing two groups, this method also allows the researcher to analyze the differences in one group for two different points in time. Understanding where the differences between two groups or two periods are coming from is crucial for policy design.

3 The iop command

3.1 Syntax

\[ iop \ depvar [ indepvars ] [ if ] [ in ] [ weight ] [ , detail shapley(stat) sgroup(str) oaxaca(groupvar stat) type(d|o|c) logit bootstrap(int) ] \]

where \( depvar \) is the outcome variable (for example, income or access to education), and \( indepvars \) are the circumstance variables as defined in section 2. \( stat \) refers to the measure of inequality of opportunity that should be decomposed, and \( groupvar \) is a categorical variable containing the definition of subgroups of the sample (for example, gender dummy).

\( fweights \) and \( iweights \) are allowed; see \( [U] 11.1.6 \) \( weight \) for details.

Note that the first version of \( iop \) had a different syntax and estimated only the method proposed by Paes de Barros, de Carvalho, and Franco (2007).\(^5\) The old syntax is still working to ensure backward compatibility.\(^6\) Nevertheless, we encourage all users to switch to the new syntax because it offers more convenient analyses.

3.2 Description

The command \( iop \) implements the four methods presented in table 1 and performs the two decomposition methods presented in section 2.2. First, we can compute a decomposition in the relative contribution of each circumstance by using the idea of the Shapley decomposition (Shorrocks 1982). Second, we can compute a decomposition for subpopulations defined in variable \( groupvar \) by using the Oaxaca–Blinder decomposition (Oaxaca 1973; Blinder 1973).

\(^{5}\) See Soloaga and Wendelspiess Chávez Juárez (2013a) for details.

\(^{6}\) \( iop \) automatically recognizes which syntax the user requests and adapts the analysis. When one uses the old syntax, a warning is displayed.
The point estimates of inequality of opportunity

The algorithm used by \texttt{iop} is very simple and is based on existing Stata commands. For binary variables, \texttt{iop} first estimates a probit model of the outcome variable on the set of circumstances. For continuous variables, it performs an \texttt{OLS} estimation. For ordered variables, \texttt{iop} estimates the probit model on each possible definition of the dichotomous variable, meaning that it creates a new dummy variable for each level of the ordered variable.

Once the algorithm estimates the regression, either probit or \texttt{OLS}, it computes the predicted values and applies the corresponding inequality measure. This provides a point estimate of inequality of opportunity. To get the relative measure, it further divides the values by the same inequality measure (for example, by the mean log deviation) of the original outcome variable.\footnote{The relative measure is available for only continuous outcome variables, because the inequality measure is equally defined for the actual and the conditional outcome. For binary variables, the actual outcome is dichotomous, while the conditional outcome (probability) is continuous. This makes it impossible to compute the relative inequality of opportunity measure in a sound way.}

\section{3.3 Options}

\texttt{iop} has seven options to adapt the analysis to the researcher’s needs. Three options are used to activate and adapt the decomposition methods, one option is used to activate the bootstrap standard errors, and the remaining options allow the user to change from the probit to the logit model, to display more details, and to correct the type of dependent variable if it was guessed incorrectly.

\texttt{detail} makes the underlying regressions (\texttt{OLS}, probit, or \texttt{logit}) visible. By default, these regressions are not displayed.

\texttt{shapley(stat)} estimates the relative importance of each circumstance variable. The argument tells \texttt{iop} which statistics to decompose. The possible values depend on the type of the variable:

<table>
<thead>
<tr>
<th>Type</th>
<th>Possible arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous</td>
<td>\texttt{fg1a}, \texttt{fg1r}, and \texttt{fg2r}</td>
</tr>
<tr>
<td>Dummy/Ordered</td>
<td>\texttt{pdb} or \texttt{ws}</td>
</tr>
</tbody>
</table>

The Shapley decomposition becomes very computationally intensive when the number of circumstances increases. Therefore, it is advisable to use this option with only a few circumstance variables.

\texttt{sgroup(str)} allows the user to group some circumstance variables when computing the Shapley value and to reduce the number of computations required. Grouping variables makes sense particularly when the variables are directly related (for example, father’s and mother’s education) or when they are inseparable (age and age
squared). To define the groups, the user has to indicate the variable names and separate the groups by a comma. For instance, assume we have the 4 variables $x_1$, $x_2$, $z_1$, and $z_2$ and would like to group the $x$ and the $z$ variables. To do this, we indicate \texttt{sgroups(x1 x2, z1 z2)}. In this case, the computation of the Shapley value requires $2^2 = 4$ instead of $2^4 = 16$ estimations. Note that the grouping of variables affects the computation of the Shapley value but does not affect the estimation of inequality of opportunity.

\texttt{oaxaca} \texttt{(groupvar stat)} activates the Oaxaca-type decomposition. The option takes two string arguments. The argument \texttt{groupvar} indicates the variable that contains the groups, and the second argument indicates which statistics must be decomposed. The group variable must be numeric and can contain value labels that are used in the display to make the output more readable. The decomposition works for only the absolute measures of inequality of opportunity (\texttt{fg1a, pdb, ws}). For the relative measures, such decomposition does not make sense, because the difference might also be due to the total amount of inequality. By correcting for that, we would be back to the absolute measure. For ordered variables, the option \texttt{oaxaca()} is not implemented, because it would yield an unmanageable amount of decompositions. In this case, a certain threshold should be chosen to dichotomize the ordered variable, and the decomposition should be used for only this threshold.

\texttt{type(d|o|c)} specifies the variable type. This option is optional because \texttt{iop} tries to figure out the type of the dependent variable on its own. If \texttt{iop} fails to identify the type, you can specify it with this option. The possible values are \texttt{d} (dummy variables), \texttt{o} (ordered variables), and \texttt{c} (continuous variables).

\texttt{logit} changes the model from the default probit to a logit model. This option is relevant only for dichotomous and ordered variables.

\texttt{bootstrap(int)} allows the user to add bootstrap standard errors to the point estimates. The argument \texttt{int} corresponds to the number of replications the user wants to estimate. Obtaining the bootstrap standard errors can be computationally intensive, so we suggest the users start with a relatively small number of replications.
3.4 Stored results

\texttt{iop} stores the following in \texttt{r()}:

**Scalars**
- \(r(\text{pdb})\): \texttt{pdb} measure
- \(r(\text{ws})\): \texttt{ws} measure
- \(r(\text{fg1a})\): \texttt{fg1a} measure
- \(r(\text{fg1r})\): \texttt{fg1r} measure
- \(r(\text{fg2r})\): \texttt{fg2r} measure
- \(r(\text{pdbSD})\): bootstrap std. error of \texttt{pdb}
- \(r(\text{wsSD})\): bootstrap std. error of \texttt{ws}
- \(r(\text{fg1aSD})\): bootstrap std. error of \texttt{fg1a}
- \(r(\text{fg1rSD})\): bootstrap std. error of \texttt{fg1r}
- \(r(\text{fg2rSD})\): bootstrap std. error of \texttt{fg2r}
- \(r(\text{bootN})\): number of bootstrap replications

**Matrices**
- \(r(\text{iop})\): matrix with all inequality measures
- \(r(\text{oaxaca})\): matrix of Oaxaca-type decomposition

The exact number of elements \texttt{iop} stores depends on the analysis performed. With respect to the scalars, it returns only computed values, thus it does not provide any empty scalars. For example, it provides the scalars with the bootstrap standard errors only when it uses the bootstrap method. For the matrices, the \(r(\text{iop})\) is always given, while the matrix \(r(\text{oaxaca})\) is provided only if such an analysis is performed.

4 Examples

In this section, we present some examples using the 2006 PISA data. In a first example, we estimate the level of inequality of opportunity for a specific country, and we perform the Shapley decomposition to identify the main drivers. In a second example, we compare different countries and use the Oaxaca-type decomposition to figure out the origin of the differences.

**Example 1: Analyzing inequality of opportunity in PISA scores**

First, we estimate the level of inequality of opportunity for Germany by using the test scores in mathematics as dependent variables and a set of family characteristics as circumstances. Among these explanatory variables, we have the occupation status of the father, parental education, the number of books at home, and a dummy for immigrants. To estimate inequality of opportunity, we indicate the dependent variable, and then we indicate the set of circumstances and specify the \texttt{if} qualifier to limit the analysis to Germany (\texttt{if cnt=="DEU"}). To complete the simple estimation, we ask \texttt{iop} to decompose the statistic \texttt{fg2r} by circumstances by using the Shapley decomposition (\texttt{shapley(fg2r)}).

---

For the Shapley decomposition, we define four variable groups: mother’s and father’s education (grouped together); a dummy for immigrants; the number of books at home; and the three indicators for the occupation of the father are grouped together.

The output of `iop` is given as follows:

```stata
. local circumstances="misced fisced immig books fcat1 fcat2 fcat3"
. iop pv1math `circumstances' if cnt=="DEU", bootstrap(100) shapley(fg2r)
. sgrou[ps](misced fisced,immig,books,fcat1 fcat2 fcat3)
I assume the variable to be: continuous
If this is not correct, use option type
Bootstrap...done!
```

Inequality of opportunity in `pv1math`

<table>
<thead>
<tr>
<th>Method</th>
<th>Absolute</th>
<th>Relative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ferreira-Gignoux (with scale)</td>
<td>0.004109</td>
<td>0.225217</td>
</tr>
<tr>
<td>Bootstrap std. err.</td>
<td>( 0.000258)</td>
<td>( 0.000258)</td>
</tr>
<tr>
<td>Ferreira-Gignoux (without scale)</td>
<td>not defined</td>
<td>0.241480</td>
</tr>
<tr>
<td>Bootstrap std. err.</td>
<td>( 0.012628)</td>
<td></td>
</tr>
</tbody>
</table>

Observations: 3967
Bootstrap replications: 100

Decomposition (Shapley method)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>In percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>0.014590</td>
<td>6.04%</td>
</tr>
<tr>
<td>Group 2</td>
<td>0.034399</td>
<td>14.25%</td>
</tr>
<tr>
<td>Group 3</td>
<td>0.136462</td>
<td>56.51%</td>
</tr>
<tr>
<td>Group 4</td>
<td>0.056029</td>
<td>23.20%</td>
</tr>
<tr>
<td>TOTAL</td>
<td>0.241480</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

The groups are defined as follows:
Group 1: misced fisced
Group 2: immig
Group 3: books
Group 4: fcat1 fcat2 fcat3

At the beginning of the output, `iop` indicates the type of variable that was assumed. In this case, `iop` correctly identified a continuous variable. If the variable type is detected incorrectly, it can be overwritten using the option `type()`. The main estimation of inequality of opportunity is presented in the first panel. In this case, the panel presents all three possible estimates for continuous variables. Because the PISA score has no inherent scale, we recommend using the second line (without a scale), which is the method proposed by Ferreira and Gignoux (2014).

The value 0.241 tells us that about one quarter of all heterogeneity in the PISA scores is due to observed circumstances. This means that about one quarter of total inequality can be considered to be ethically offensive and is not due to students’ different levels of effort or to luck. The bootstrap standard errors below the point estimates are based on 100 replications and are about 1.2%, which is relatively small. Recall that this is
a lower-bound estimate for the reasons outlined in the introduction and discussed in depth by Ramos and Van de gaer (2012).

The second panel provides results for the Shapley decomposition of the estimated inequality of opportunity. The results are presented by level and as percentages of total inequality of opportunity. In our example, the number of books at home accounts for more than half of total inequality of opportunity, while parental education categories (\text{fisced} and \text{misced}) do not account for much. A father’s job categories (\text{fcat1–fcat3}) account together for about 23% of total inequality of opportunity, and immigration status accounts for about 14%. Note, however, that Ferreira and Gignoux (2014) argue that this decomposition must be used with caution. Highly correlated circumstances might lead to biased coefficients, which is not directly a problem for the estimation of $\theta_{\text{IOP}}$. However, it might be problematic for the decomposition in relative contributions of circumstances.

Example 2: Dichotomous outcome and Oaxaca-type decomposition

In the second example, we use the same data but include Canada and the United States in the analysis. Instead of using the PISA score as we did above, we now use a binary indicator, taking the value of 1 for students that have achieved 500 points or more on the PISA test and 0 otherwise.\footnote{Five hundred is the average PISA test score of all countries. We perform this dichotomization exclusively for illustrative purposes to show how \text{iop} handles binary variables.} Moreover, we use the option \text{oaxaca(country pdb)} to decompose the measure proposed by Paes de Barros, de Carvalho, and Franco (2007) in the spirit of an Oaxaca–Blinder decomposition. The variable \text{country} is a categorical variable (numerical) with value labels.

Example 2: Dichotomous outcome and Oaxaca-type decomposition

<table>
<thead>
<tr>
<th>Method</th>
<th>Absolute</th>
<th>Relative</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{PdB} (Dissimilarity index) vs (adapted DI)</td>
<td>.115558</td>
<td>not defined</td>
</tr>
<tr>
<td></td>
<td>.266169</td>
<td>not defined</td>
</tr>
</tbody>
</table>

Oaxaca-like decomposition

<table>
<thead>
<tr>
<th>Distribution</th>
<th>CAN</th>
<th>GER</th>
<th>USA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN</td>
<td>0.09212</td>
<td>0.13256</td>
<td>0.17369</td>
</tr>
<tr>
<td>GER</td>
<td>0.10727</td>
<td>0.15270</td>
<td>0.20991</td>
</tr>
<tr>
<td>USA</td>
<td>0.10433</td>
<td>0.15184</td>
<td>0.19557</td>
</tr>
</tbody>
</table>
The output produced by \texttt{iop} starts again by guessing the type of dependent variable and then provides the general analysis of all the countries together. The point estimates are 0.116 and 0.266 for the two methods, respectively. The adapted dissimilarity index (\texttt{ws}) is defined on the interval of 0 to 1, so the value of 0.266 suggests that a rather large amount of inequality is due to circumstances.

This general analysis is followed by the Oaxaca-type decomposition presented in matrix form. On the diagonal, we have the estimate for each country, where Canada displays the lowest level of inequality of opportunity, followed by Germany and the United States. The remaining values are counter-factual estimates, where the column refers to the estimated coefficients, and the rows refer to the composition. For instance, the value in the first column, \texttt{CAN}, and the last row, \texttt{USA}, would be the level of inequality of opportunity with the distribution of circumstances of the United States and the estimated coefficients of Canada. The value lies much closer to the original value of Canada, which suggests that most of the difference is due to differences in the link between circumstances and outcome, while very little of the difference is due to a different structure of circumstances.

\textbf{Example 3: Ordered variables}

A final example presents the output for ordered variables. For simplicity and for illustrative purposes, we use the same variables as before. Instead of dichotomizing the scores as we did for example 2, we create an ordered variable with 4 categories. The categories are

\[
\begin{align*}
\text{mathORD} & = \\
& \begin{cases} 
1 & \text{if score} \leq 400 \\
2 & \text{if } 400 < \text{score} \leq 500 \\
3 & \text{if } 500 < \text{score} \leq 600 \\
4 & \text{if } 600 < \text{score}
\end{cases}
\end{align*}
\]

Additionally, to change the variable, we indicate to \texttt{iop} that we want to use logit instead of probit. The output is as follows:

\begin{verbatim}
. iop mathORD misced fisced immig books fcat1 fcat2 fcat3, logit
Note: logit was used instead of probit
I assume the variable to be: ordered
if this is not correct, use option type

<table>
<thead>
<tr>
<th>Inequality of opportunity in mathORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>mathORD &lt; 2</td>
</tr>
<tr>
<td>mathORD &lt; 3</td>
</tr>
<tr>
<td>mathORD &lt; 4</td>
</tr>
</tbody>
</table>

Only absolute estimates are reported
Observations: 27832
\end{verbatim}
Before showing the estimates, \texttt{iop} informs the user that it used logit instead of probit and that it detected an ordered variable. The actual output of results is very much like the output of results for dummy variables, the difference being that there are two estimates for every possible threshold of the ordered variable. In this respect, the first line provides the estimate for inequality of opportunity in the probability of having at least 400 points in the score. The second and the third lines are for at least 500 and at least 600 points, respectively. Note that the second line is comparable—but, because of the change from probit to logit, not identical—with example 2. The example shows that using the scale invariant measure (PdB), we find the highest values for the highest threshold, while the translation invariant measure indicates the highest level of inequality for the threshold in the middle.\footnote{A discussion on the conceptual differences between the two measures can be found in Soloaga and Wendelspiess Chávez Juárez 2013b.}

\section{Concluding remarks and limitations}

In this article, we described the user-written command \texttt{iop}, which estimates several methods to assess ex-ante inequality of opportunity. In addition to the point estimates, \texttt{iop} proposes two decompositions. The first decomposition allows the researcher to identify the relative importance of the included circumstances using the Shapley value. The second decomposition allows the researcher to better understand differences in inequality of opportunity between groups (for example, between countries or regions within a country) using an Oaxaca-type decomposition. The main goal of the command \texttt{iop} is to offer interested researchers an easy-to-use command that allows them to estimate inequality of opportunity with different methods. The choice of the implemented methods was driven by the recent use of these methods in empirical applications. We are confident that \texttt{iop} supports most of the commonly used methods.

In this respect, we would also like to highlight some limitations of the current version of the command. First, \texttt{iop} estimates only ex-ante inequality of opportunity. It would be difficult to combine the alternative ex-post methods in one command, because they require different data and have substantially different approaches. Second, \texttt{iop} supports parametric estimates by using OLS for continuous variables and by using probit and logit models for dichotomous variables. However, a nonparametric approach based on type averages can be estimated by using type dummy variables in the parametric regression. Finally, \texttt{iop} omits analytical standard errors of the estimators and limits itself to bootstrap standard errors for the point estimates. There are no bootstrap standard errors included for the decompositions, because their statistical properties are unclear.

We plan to further develop \texttt{iop} in accordance with the propositions of estimators for inequality of opportunity. We are always happy to receive comments and suggestions for future developments.
6 References


Paes de Barros, R., M. de Carvalho, and S. Franco. 2007. Preliminary notes on the measurement of socially-determined inequality of opportunity when the outcome is discrete. Working paper.


About the authors

Florian Wendelspiess Chávez Juárez is a postdoctoral scholar at the University of Geneva, Switzerland.

Isidro Soloaga is professor of economics at the Universidad Iberoamericana in Mexico City.
femlogit—Implementation of the multinomial logit model with fixed effects

Klaus Pforr
GESIS—Leibniz–Institute for the Social Sciences
Mannheim, Germany
klaus.pforr@gesis.org

Abstract. Fixed-effects models have become increasingly popular in social-science research. The possibility to control for unobserved heterogeneity makes these models a prime tool for causal analysis. Fixed-effects models have been derived and implemented for many statistical software packages for continuous, dichotomous, and count-data dependent variables. Chamberlain (1980, Review of Economic Studies 47: 225–238) derived the multinomial logistic regression with fixed effects. However, this model has not yet been implemented in any statistical software package. Possible applications would be analyses of effects on employment status, with special consideration of part-time or irregular employment, and analyses of effects on voting behavior that implicitly control for long-time party identification rather than measuring it directly. This article introduces an implementation of this model with the new command femlogit. I show its application with British election panel data.

Keywords: st0362, femlogit, multinomial logit, fixed effects, panel data, multilevel data, unobserved heterogeneity, discrete choice, random effects, conditional logit

1 Introduction

Fixed-effects models have become increasingly popular in sociology. The possibility to control for unobserved heterogeneity makes these models a prime tool for causal analysis (Gangl 2010; Brüderl and Ludwig 2015). Fixed-effects models for continuous, dichotomous, and count dependent variables are widely used and available in Stata and many other software packages. However, a fixed-effects estimator for polytomous discrete dependent variables is not yet available for any statistical software package (Allison 2009, 44). The available alternatives for such dependent variables are the pooled multinomial logistic or probit regression (Wooldridge 2010, 609; Rabe-Hesketh and Skrondal 2012, 653–658) and the multinomial logistic or probit regression with random effects (Wooldridge 2010, 619ff.; Rabe-Hesketh and Skrondal 2012, 659ff.). For both models, we must assume that any unobserved heterogeneity is independent of the observed covariates.

In this article, I present an implementation of the multinomial logistic regression with fixed effects (femlogit) in Stata. The femlogit command implements an estimator by Chamberlain (1980). The implementation draws on the native Stata multinomial logit and conditional logit model implementations. The actual ml evaluator uses Mata functions to implement the conditional likelihood function.
Possible applications of the fixed-effects estimator include analyses of effects on employment status, with special consideration of part-time or irregular employment, and analyses of the effects on voting behavior that implicitly control for stable individual differences in party preference rather than measuring it directly.

After explaining the mathematical background and the implementation of the model, I will discuss the syntax of femlogit. Then I will show the application of the ado-file and the interpretation of its results with a model of voting behavior with British election panel data.

2 Statistical model

The statistical model was first proposed by Chamberlain (1980, 231). More extensive expositions are found in Lee (2002, 143ff.) and Pforr (2013). Here I assume a sample of individuals \(i = 1, \ldots, N\) with observations across time \(t = 1, \ldots, T_i\). The outcome variable, \(o_j\) with \(j = 1, \ldots, J\), is a polytomous categorical variable with \(J\) identical levels for all individuals and observation times. The values of the outcome levels are unrestricted: \(\forall j: \alpha_j \in \mathbb{R}\). For each individual \(i\) and each observation time \(t\), the chosen outcome, \(y_{it}\), is measured as the dependent variable and a vector of \(M\) independent variables \(x_{it} = (x_{it1}, \ldots, x_{itm})\). Next to the realized choices, I define \(y^*_{itj}\) as the latent propensity for each individual \(i\) at time \(t\) to choose outcome \(j\). With this notation, I assume the following relation between the propensities, \(y^*_{itj}\), and the independent variables, \(x_{itj}\):

\[
\forall j \in (1, \ldots, J): y^*_{itj} = \alpha_{ij} + x_{it}\beta_j + \epsilon_{itj} 
\]

In this equation, \(\beta_j\) is the coefficient vector, which must be estimated. \(\alpha_{ij}\) is a random variable. The error term, \(\epsilon_{itj}\), is a type I (Gumbel-type) extreme-value random variable, independent and identically distributed across all outcomes \(j\). The link to the chosen outcome is defined by

\[
\forall j \in (1, \ldots, J): \Pr(y_{it} = o_j | \alpha_i, \beta, x_{it}) = \Pr\left(\max_{k \in (1, \ldots, J)} y^*_{itk} = y^*_{itj} | \alpha_i, \beta, x_{it}\right) 
\]

With these assumptions, I can derive the probabilities of each outcome. To guaranteed identifiability, I define an arbitrarily chosen outcome \(B \in (1, \ldots, J)\) as the base outcome, and I restrict the respective coefficients to 0: \(\alpha_{iB} = 0, \beta_B = 0\). From this, I obtain

\[
\Pr(y_{it} = o_j | \alpha_i, \beta, x_{it}) = \begin{cases} 
\frac{\exp(\alpha_{ij} + x_{it}\beta_j)}{1 + \sum_{k \neq B} \exp(\alpha_{ik} + x_{it}\beta_k)} & j \neq B \\
\frac{\exp(\alpha_{ij} + x_{it}\beta_j)}{1 + \sum_{k \neq B} \exp(\alpha_{ik} + x_{it}\beta_k)} & j = B
\end{cases}
\]

1. The subscript \(i\) at \(T_i\) means that the model allows for analyzing unbalanced panel data. However, attrition must be at least at random; that is, attrition is completely at random when conditioning for the independent variables (Wooldridge 2010, 828).
So far, I have set up the assumptions for the pooled multinomial logistic regression, which can be consistently estimated if there is no unobserved heterogeneity: \( \forall j : \alpha_{ij} = \alpha_j \).

The advantage of the multinomial logit model with fixed effects is that it allows for individual unobserved heterogeneity with respect to the intercepts. The heterogeneity terms, \( \alpha_{ij} \), are random variables with no restrictions on the joint distribution with the independent variables, \( x_{it} \). Directly estimating the individual \( \alpha_{ij} \) creates an incidental parameter problem, which leads to inconsistent estimators with asymptotics solely based on \( N \to \infty \). However, with additional assumptions, it is possible to consistently estimate the coefficient vector \( \beta \). First, we assume that the observed covariates are strictly exogenous conditional on the unobserved heterogeneity.

Second, we assume that the error terms are independent across time. That is, autocorrelation is ruled out.

Chamberlain (1980) states that under these additional assumptions, the term \( \theta_{ij} = \sum_{t=1}^{T_i} \delta_{y_{it}o_j} \), where \( \delta \) denotes the Kronecker delta function with respect to \( y_{it} \) and \( o_j \), is a sufficient statistic for the unobserved heterogeneity, \( \alpha_{ij} \). This relation means that the sum of occurrences of an outcome \( j \) for an individual \( i \) across time is a sufficient statistic for inclination toward that outcome.

Because there is a sufficient statistic for the unobserved heterogeneity, one can reformulate the likelihood function so that the estimands, \( \alpha_{ij} \), disappear. The probability mass function for the sequence of chosen outcomes across time for individual \( i \) conditional on the sufficient statistic is

\[
\begin{align*}
\prod_{t=1}^{T_i} \prod_{j=1}^{J} \Pr (y_{it} = o_j | \alpha_i, \beta, x_i, \theta_i) \delta_{y_{it}o_j} = \\
\sum_{\upsilon_i \in \Upsilon_i} \prod_{t=1}^{T_i} \prod_{j=1}^{J} \Pr (v_{it} = o_j | \alpha_i, \beta, x_i, \theta_i) \delta_{v_{it}o_j}
\end{align*}
\]

The summation in the denominator is taken over all “potential” sequences of chosen outcomes \( \upsilon_i = (v_{i1}, \ldots, v_{iT_i}) \) that fulfill the condition of the sufficient statistic \( \theta_i \). The set \( \Upsilon_i \) contains all sequences \( \upsilon_i \) for which the sum of occurrences of each outcome \( j \) is the same as it is for the realized sequence \( y_i \). Formally, this means

\[
\Upsilon_i = \left\{ (v_{i1}, \ldots, v_{iT_i}) \bigg| \forall j \in (1, \ldots, J): \sum_{t=1}^{T_i} \delta_{v_{it}o_j} = \sum_{t=1}^{T_i} \delta_{y_{it}o_j} = \theta_{ij} \right\}
\]
Technically, \( \check{\sum} \) is the set of all permutations of the realized sequence of chosen outcomes \( y_i \). Taking into account the assumptions and definitions above, we can write (4) as

\[
f_{y_i|\alpha_i, \beta, x_i, \theta_i} = \frac{\exp\left( \sum_{t=1}^{T_i} \sum_{j=1, j \neq B}^{J} \delta_{y_{it}, o_j} x_{it} \beta_j \right)}{\sum_{v_i \in \check{\sum}} \exp\left( \sum_{t=1}^{T_i} \sum_{j=1, j \neq B}^{J} \delta_{v_{it}, o_j} x_{it} \beta_j \right)}
\]

Having derived the probability mass function, we see that the simplified expression of the log-likelihood function of the multinomial logit model with fixed effects follows its definition. The contribution to the log likelihood of individual \( i \) is

\[
\ln \ell_i (\beta|y_i, x_i) = \ln f_{y_i|\alpha_i, \beta, x_i, \theta_i} = \sum_{t=1}^{T_i} \sum_{j=1, j \neq B}^{J} \delta_{y_{it}, o_j} x_{it} \beta_j - \ln \sum_{v_i \in \check{\sum}} \exp\left( \sum_{t=1}^{T_i} \sum_{j=1, j \neq B}^{J} \delta_{v_{it}, o_j} x_{it} \beta_j \right)
\]

Therefore, the overall log-likelihood function for the sample—given a simple random sample of panel groups—is

\[
\ln L (\beta|y, x) = \sum_{i=1}^{N} \ln \ell_i (\beta|y_i, x_i)
\]

The maximum likelihood (ML) estimator of (7) is a consistent asymptotically normal estimator of the coefficient vector \( \beta \) (Wooldridge 2010, 473–481).

### 2.1 Special case: Binary logit with fixed effects

The binary logit with fixed effects is a special case of the multinomial logit model with fixed effects with \( J = 2 \). Usually, the outcome variable \( o_j \) is coded as \( o_1 = 0 \) and \( o_2 = 1 \). Also the base outcome is commonly defined as \( B = 1 \). This simplifies (2) to

\[
\Pr (y_{it} = 1|\alpha_i, \beta, x_{it}) = \frac{\exp (\alpha_i + x_{it} \beta)}{1 + \exp (\alpha_i + x_{it} \beta)},
\]

\[
\Pr (y_{it} = 0|\alpha_i, \beta, x_{it}) = \frac{1}{1 + \exp (\alpha_i + x_{it} \beta)}
\]

Note that the heterogeneity term, \( \alpha_i \), is now a scalar because it reflects only the contrast between outcome \( o_2 = 1 \) and \( o_1 = 0 \). Similarly, the remaining coefficient vector \( \beta \) also reflects only this contrast. Furthermore, (6), which is the basis of the log-likelihood function, is simplified to

\[
f_{y_{it}|\alpha_i, \beta, x_i, \theta_i} = \frac{\exp\left( \sum_{t=1}^{T_i} y_{it} x_{it} \beta \right)}{\sum_{v_i \in \check{\sum}} \exp\left( \sum_{t=1}^{T_i} v_{it} x_{it} \beta \right)}
\]

Note that the simplification of \( \delta_{v_{it}, o_j} \) to \( y_{it} \) rests on the specific dummy coding of \( y_{it} \). For more details on this model and its implementation in Stata, see [R] clogit.
2.2 Interpretation

Usually, the estimates of binary and multinomial response models are interpreted as odds-ratio or logit effects or as effects on the predicted probabilities and related constructs (for example, average marginal effects).

Regarding the first class, odds-ratio and logit effects are criticized as unintuitive. Moreover, with this interpretation approach based on arbitrary restriction assumption of the variance of the error term \( \epsilon \) in (1), effects across nested models or across different groups cannot be easily compared (Allison 1999; Kohler, Karlson, and Holm 2011; Best and Wolf 2015; Breen, Karlson, and Holm 2013).

Therefore, for nonlinear cross-sectional models, the second class of interpretation constructs is recommended (Long and Freese 2006, 157ff.). This option is not given for the fixed-effects model. The probability expression in (2) cannot be evaluated, because the unobserved heterogeneity vector \( \alpha \) is not estimated. Even if plausible values for \( \alpha \) are inserted in the equation, to conduct significance tests, one has to find plausible values for their variances and covariances with the other independent variables. Cameron and Trivedi (2005, 797) suggest using the binary logistic regression with fixed effects to interpret predicted probabilities of the estimation (6), which can be generalized to the multinomial case. However, although this circumvents the problem of finding a plausible conditional distribution for the unobserved heterogeneity \( f_{\alpha|x} \), the object of interpretation here is more unintuitive than with the odds-ratio and logit effects. With this approach, one interprets the effects of a unit or marginal change in the independent variables at a specific time \( x_t \) on the probability that a specific time series of outcomes \( y_1, \ldots, y_T \) is realized conditional on the probability of all permutations of the time series. For realistic applications, any choice of the time series of outcomes is arbitrary. Furthermore, the interpretation of the conditional probability remains imprecise, because the permutation can be understood only as an analogue for the tendency to choose each outcome. The odds-ratio effects interpretation as shown above is the only viable option for the binary and multinomial fixed-effects logistic regression.

2.3 Robust standard errors

For other models, specifically those with panel data, it is common to report Huber–White or sandwich-estimator standard errors. These standard errors are robust to specific violations of model assumptions. For linear panel-data models, sandwich-estimator standard errors are robust to heteroskedasticity and serial correlation (see Cameron and Trivedi [2005, 705ff.]). For multilevel models with continuous dependent variables, sandwich-estimator standard errors can be robust to heteroskedasticity and correlation within higher-level units across lower-level units.
femlogit

However, for a nonlinear model with fixed effects as described here, the robustness of the sandwich estimator is restricted to violation of homoskedasticity on the level of the panel groups (Wooldridge 2010, 608–625). The assumption of error independence across time, (3), must be maintained; that is, the sandwich estimator is not robust to violation of this assumption. However, the sandwich estimator is robust to violation of independence across panel groups \( i \). This is equivalent to heteroskedasticity robustness. Note that this implies robustness to varying error variances within and between clusters of panel groups.

If the assumption of independence across time is violated, the ML estimator of (7) is inconsistent and can be interpreted only as a quasi-ML estimator, where the sandwich-estimator standard errors can be used to “test hypotheses about the best approximation to the true density” (Wooldridge 2010, 503). Note that xtlogit, fe also precludes robust standard errors.

3 Implementation

To implement femlogit, I use the moptimize() Mata suite because it allows me to implement the evaluator as a gf2 type. This increases precision and computational speed (Gould, Pitblado, and Poi 2010, 20–24). Moreover, the gf2-type evaluator enables a straightforward consideration of the panel-data structure and an easier integration into the svy command suite.\(^{2}\) The evaluator is implemented as a Mata function. Besides being the natural choice with moptimize(), this allows a straightforward integration of the Mata function cvpermute(), which is used to loop over the set \( \Upsilon_i \) in (5). The gf2-type evaluator expects arguments to be the dependent variable column vector \( y: \sum_{i=1}^{N} T_i \times 1 \), the independent variable matrix \( x: \sum_{i=1}^{N} T_i \times M \), and an initial coefficient row vector \( \beta: 1 \times (J - 1) M \). The evaluator returns the column vector \( \{ \ln \ell_i(\beta) \}: N \times 1 \) of the individual contributions for all panel groups, the gradient matrix \( g: N \times (J - 1) M \), and the Hessian matrix \( H: (J - 1) M \times (J - 1) M \). For the latter two function outcomes, the first and second partial derivatives, with respect to all coefficients \( \beta_{jm} \), are derived analytically and inserted in the Mata evaluator.

The moptimize() call is embedded in an ado-wrapper, following the structure of the implementations of mlogit and clogit. Worth mentioning here are the definition of the estimation sample and the initial values of the coefficient vector \( \beta \). For the estimation sample, first, observations with missing values on the dependent, independent, or panel-group indicator variables are deleted. Second, collinear independent variables are excluded. Finally, panel groups without variance across time in the dependent variables, as well as independent variables without variance across time in all panel groups, are dropped. The initial values for the coefficient vector are the estimated coefficients of a pooled multinomial logit model. This follows the implementation of clogit, where the initial values are taken from the pooled binary logit model. The implemented command identifies panel groups by using the panel-group indicators set by xtset.

\(^{2}\) Note that there is no support for weights and the svy command suite in the current version of the implementation.
3.1 Data structure

The implementation expects the data to be organized in long format—that is, from the panel-data perspective, each observation represents a time point of one person. The following is a modified version of the example data used in \([\text{R}]\) \text{clogit}:\(^3\)

\begin{verbatim}
. use femlogitid
. list in 1/11

<table>
<thead>
<tr>
<th>id</th>
<th>y</th>
<th>x1</th>
<th>x2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1014</td>
<td>3</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1014</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1014</td>
<td>2</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>1014</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>1017</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1017</td>
<td>2</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>1017</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1019</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1019</td>
<td>2</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>1019</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>1019</td>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>
\end{verbatim}

The first four observations belong to the person with the id = 1014. The independent variables are \(x_1\) and \(x_2\), and \(y\) is the categorical dependent variable with four levels \((0, 1, 2, 3)\). Note that the different levels of the categorical dependent variable are stored in one variable and one case, similarly to \(mlogit\). In contrast, the implementation of \text{clogit} \ expects the outcomes of the dependent variable for each time point to be stored in long format.

3.2 Computational problems

The current implementation enumerates the sum over all permutations of the individual sequences \(y_i\) in the denominator of (6). This means that computation time increases with the number of permutations in the dependent variable. In practice, this will rise with \(T_i\). The computation time can be very high, even if \(T_i\) is large for only a small subset of individuals \(i = 1, \ldots, N\). If computation becomes unwieldy, a random subset of available measurements of all observation units should be analyzed. This selection should not depend on the number of available measurements for each observation. Increasing \(N\) should not increase the computation time severely.

---

\(^3\) The data \text{femlogitid.dta} and syntax \text{femlogit_example1.do} can be found in the online appendix.
4 The femlogit command

4.1 Syntax

The command femlogit is called with the following syntax:

\[
\text{femlogit depvar [indepvars] [if] [in] [group(varlist) baseoutcome(#)] constraints(clist) difficult or robust]}
\]

\textit{depvar} and \textit{indepvars} may not contain factor variables or time-series operators. No prefix commands are allowed. Weights and \texttt{vce()} are not allowed at this point.

4.2 Options

\texttt{group(varlist)} specifies one or more identifier variables (numeric or string) for the matched groups. It overrides the default group indicator that is specified with \texttt{xtset}.

\texttt{baseoutcome(#)} specifies the value of \textit{depvar} to use as the base outcome. The default is to choose the mode outcome.

\texttt{constraints(clist)} specifies the linear constraints to be applied during estimation. The default is to perform unconstrained estimation. \textit{clist} has the form \texttt{# -# [ , # [-#] ... ]}.

\texttt{difficult} specifies that the “hybrid” method be used in nonconcave regions of the likelihood function instead of the default “modified Marquardt” method (Gould, Pitblado, and Poi 2010, 15–17).

\texttt{or} reports the estimated coefficients transformed to odds ratios, that is, \texttt{exp(b)} rather than \textit{b}. Confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated.

\texttt{robust} uses the robust or sandwich estimator of variance. This is valid only for quasi-ML interpretation (Wooldridge 2010, 502ff.). It can be interpreted only as heteroskedasticity robustness, not as panel robustness.
4.3 Stored results

`femlogit` stores the following in `e()`:

Scalars

- `e(N)`: number of observations
- `e(N_drop)`: number of observations dropped because of invariant dependent variable
- `e(N_group_drop)`: number of groups dropped because of invariant dependent variable
- `e(k)`: number of parameters
- `e(k_eq)`: number of equations in `e(b)`
- `e(k_eq_model)`: number of equations in overall model test
- `e(df_m)`: model degrees of freedom
- `e(N)`: number of observations
- `e(ll)`: log likelihood dropped because of invariant dependent variable
- `e(ll_0)`: log likelihood, constant-only invariant dependent model variable
- `e(chi2)`: \( \chi^2 \) test
- `e(p)`: significance
- `e(rank)`: rank of `e(V)`
- `e(ic)`: number of iterations
- `e(rc)`: return code
- `e(ilog)`: iteration log (up to 20 iterations)
- `e(V)`: variance–covariance matrix of the estimator
- `e(V_modelbased)`: model-based variance
- `e(V)`: variance–covariance matrix
- `e(Cns)`: constraints matrix
- `e(gradient)`: gradient vector

Macros

- `e(cmd)`: `femlogit`
- `e(cmdline)`: command as typed
- `e(depvar)`: name of dependent variable
- `e(chi2type)`: Wald or LR, type of model \( \chi^2 \) test
- `e(properties)`: `b V` test
- `e(predict)`: predict
- `e(vce)`: oim or robust
- `e(vctype)`: Robust
- `e(opt)`: moptimize
- `e(which)`: max
- `e(ml_method)`: `gf2`
- `e(user)`: `femlogit_eval_gf2()`
- `e(technique)`: nr
- `e(crittype)`: log likelihood or log pseudolikelihood
- `e(marginsok)`: xb
- `e(marginsnotok)`: stdp stddp
- `e(eqnames)`: names of equations
- `e(group)`: name of `group()` variable

Matrices

- `e(b)`: coefficient vector
- `e(Cns)`: constraints matrix
- `e(gradient)`: gradient vector
- `e(gradient)`: gradient vector
- `e(gradient)`: gradient vector
- `e(gradient)`: gradient vector
- `e(gradient)`: gradient vector
- `e(gradient)`: gradient vector
- `e(gradient)`: gradient vector
- `e(gradient)`: gradient vector

Functions

- `e(sample)`: marks estimation sample

5 Application: Effect of ideological distance on voting behavior with British election panel data

In this section, I demonstrate the `femlogit` command and explain how to interpret the results. My example follows the one that Skrondal and Rabe-Hesketh (2003) use to illustrate the application of multilevel random-effects models for polytomous and ordinal dependent variables. They analyze data from the 1987–1992 panel of British Election Study (Heath et al. 1992) to fit a model of the recalled vote choice for the Conservative, Labour, or Liberal party and a model of the rank order of the parties. Here I concentrate on the recalled vote choice and use the `femlogit` command to estimate the effect of the distance on the left–right policy dimension between the voter and the party on the vote choice. I control for the time-varying rating of perceived inflation and implicitly for all
time-variant factors at the voter level. The analysis syntax for this example is found in `femlogit_example2.do`, which is provided in the online appendix.

The raw data are taken from Rabe-Hesketh and Skrondal (2012, 680ff.). Cleaning and preparation leads to the following analysis data:

```
. describe
Contains data
  obs: 2,458
  vars: 9
  size: 46,702

  variable name    storage  display     value label
  name             type     format    label
  serialno          int       %8.0g  Respondent number
  rldist2           float     %9.0g  Dist(Labour)-Dist(Conservative)
  rldist3           float     %9.0g  Dist(Liberal)-Dist(Conservative)
  male              byte      %8.0g  Male
  manual            byte      %8.0g  Manual worker
  inflation         byte      %8.0g  Perceived inflation
  age               float     %9.0g  Age in 10 yr units
  yr92              byte      %8.0g  1992 election indicator
  choice            byte      %12.0g     Recalled vote for party

Sorted by: serialno
Note: dataset has changed since last saved
```

The dependent variable `choice` is a discrete variable with three alternatives—“Conservative”, “Labour”, and “Liberal”. In the multinomial logit model with fixed effects, the following four independent variables are used: the difference of the distance between the voter and the Labour party and the distance between the voter and the Conservative party (`rldist2`); the difference of the distance between the voter and the Liberal party and the distance between the voter and the Conservative party (`rldist3`); a rating of the perceived inflation (`inflation`); and a wave dummy (`yr92`).

The data are in long format. As the summary command for panel data, `xtdescribe`, shows, the dataset contains information on 1,344 persons across both elections. For 1,114 persons, the time series across both waves is complete. For the remaining 230 persons, information is missing for at least one wave.
K. Pforr

. xtset serialno yr92
  panel variable: serialno (unbalanced)
time variable: yr92, 0 to 1
delta: 1 unit
  . xtdescribe
  serialno: 2, 11, ..., 5997 n = 1344
  yr92: 0, 1, ..., 1 T = 2
  Delta(yr92) = 1 unit
  Span(yr92) = 2 periods
  (serialno*yr92 uniquely identifies each observation)

  Distribution of T_{i}: min 5% 25% 50% 75% 95% max
  Freq. Percent Cum. Pattern
  1114 82.89 82.89 11
  121 9.00 91.89 1.1
  109 8.11 100.00 .1
  1344 100.00 XX

The differences in the policy distances vary not only across voters and waves but also across alternatives. This allows us to specify the model as a mixed-logit model (Cameron and Trivedi 2005, 495). That is, I estimate one coefficient for the alternative-varying policy distances and alternative-specific coefficients for the alternative-invariant voters' rating of inflation and the wave dummy. To do this, I define the following constraints for the effects of the policy distances:

  . constraint 1 [Labour]rldist3=0
  . constraint 2 [Liberal]rldist2=0
  . constraint 3 [Labour]rldist2=[Liberal]rldist3

With these constraints, the effect of the relative policy distance between the voter and the Liberal party plays no role in the propensity to vote for labor in comparison with the Conservative party and vice versa. The relative policy distance between the voter and the Labour party is irrelevant in the propensity to vote for the Liberal party instead of the Conservative party. The third constraint guarantees that the relative policy distances have the same effect on both propensities.

4. This specification should not be confused with logistic regression with random slopes or random covariate effects, which is implemented as mixlogit by Hole (2007).
The estimation output of `femlogit` for this model is as follows:

```
. femlogit choice rldist2 rldist3 inflation yr92, group(serialno) const(1/3)
> b(1)
note: 1097 groups (1964 obs) dropped because of all positive or all negative outcomes.
Iteration 0: log likelihood = -156.16844
Iteration 1: log likelihood = -139.49392
Iteration 2: log likelihood = -138.19403
Iteration 3: log likelihood = -138.19006
Iteration 4: log likelihood = -138.19006

Fixed-effects multinomial logistic regression
Number of obs = 494
Wald chi2(5) = 45.69
Log likelihood = -138.19006
Prob > chi2 = 0.0000
( 1) [Labour]rldist3 = 0
( 2) [Liberal]rldist2 = 0
( 3) [Labour]rldist2 - [Liberal]rldist3 = 0

| choice       | Coef.  | Std. Err. | z     | P>|z|  | [95% Conf. Interval] |
|--------------|--------|-----------|-------|------|----------------------|
| Conservative | (base outcome) |
| Labour       |        |           |       |      |                      |
| rldist3      | -.0590691 | .0145332 | -4.06 | 0.000| -.0875536 -.0305846 |
| inflation    | .8354586 | .3692285  | 2.26  | 0.024| .111784 1.559133    |
| yr92         | .6791261 | .2734095  | 2.48  | 0.013| .1432534 1.214999   |
| Liberal      |        |           |       |      |                      |
| rldist3      | -.0590691 | .0145332 | -4.06 | 0.000| -.0875536 -.0305846 |
| inflation    | .5786913 | .305657   | 1.89  | 0.068| -.0203854 1.177768  |
| yr92         | -.2315669 | .2188483 | -1.06 | 0.290| -.6605018 .1973679  |
```

The output header shows that 1,097 voters and 1,964 observations are dropped, because there is no variance in the dependent variable across waves for these voters. That is, the model is fit with 247 voters and 494 observations. The iteration log shows that the ML algorithm converged after four steps. The log likelihood for the first step is derived from the initial coefficient vector, which is the result of pooled multinomial logit with the same variable structure. The header also shows the Wald test statistic of 45.69. The five degrees of freedom reflect the reduced number of a free number of parameters. Note that the command returns a Wald test instead of a likelihood-ratio test because constraints were specified.

The coefficient table shows the logarithm of the relative-risk ratios for a one-unit change in the corresponding variables. That is, with an increase in the relative distance between a voter and the Labour party by one unit ceteris paribus, the logarithm of the probability to vote for labor divided by the probability to vote for the Conservative party decreases by 0.059. Equivalently, ceteris paribus, this relative distance increases by one unit, and the odds to vote for labor versus voting conservative increase by a factor of $\exp(0.059) = 0.943$; that is, they decrease by 6.7%. Similarly, with each unit increase in the inflation rating ceteris paribus, the odds to vote for labor versus voting...
conservative increase by 130.6%, and the odds to vote liberal versus voting conservative increase by 78.4%. One can interpret the odds effects for other contrasts by looking at the respective coefficient or variable differences. For example, if the inflation rating increases by one unit ceteris paribus, the odds to vote labor versus voting liberal increase by a factor of \( \exp(0.835 - 0.579) = 1.293 \) or 29.3%.

As mentioned previously, the multinomial logit model with fixed effects allows for possibly confounding unobserved heterogeneity at the level of the voter with respect to the preferences for a specific party. Alternative models have to rule this out or have to measure the heterogeneity. In table 1, I show the respective effects for the pooled multinomial logistic regression and the multinomial logistic regression with random effects. For the first model, panel–robust standard errors are used to account for possible correlation across waves. The latter model is fit with \texttt{gsem}, as described in [SEM] \texttt{example 41g}. In the alternative models, heterogeneity is captured in the time-invariant variables \texttt{male}, \texttt{age}, and \texttt{manual}. 
Table 1. Pooled, random-effects, and fixed-effects models for voting example

<table>
<thead>
<tr>
<th></th>
<th>POMLOGIT</th>
<th>REMLOGIT</th>
<th>FEMLOGIT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>exp((\beta))/se</td>
<td>exp((\beta))/se</td>
<td>exp((\beta))/se</td>
</tr>
<tr>
<td>Labour</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relat. policy dist.</td>
<td>0.896***</td>
<td>0.818***</td>
<td>0.943***</td>
</tr>
<tr>
<td></td>
<td>(0.005)</td>
<td>(0.011)</td>
<td>(0.014)</td>
</tr>
<tr>
<td>Inflation</td>
<td>2.134***</td>
<td>3.812***</td>
<td>2.306*</td>
</tr>
<tr>
<td></td>
<td>(0.236)</td>
<td>(0.815)</td>
<td>(0.851)</td>
</tr>
<tr>
<td>1992 election</td>
<td>1.153</td>
<td>1.564*</td>
<td>1.972*</td>
</tr>
<tr>
<td></td>
<td>(0.112)</td>
<td>(0.346)</td>
<td>(0.539)</td>
</tr>
<tr>
<td>Male</td>
<td>0.452***</td>
<td>0.261***</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.068)</td>
<td>(0.082)</td>
<td></td>
</tr>
<tr>
<td>Age</td>
<td>0.702***</td>
<td>0.499***</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.037)</td>
<td>(0.056)</td>
<td></td>
</tr>
<tr>
<td>Manual worker</td>
<td>1.952***</td>
<td>5.188***</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.302)</td>
<td>(1.767)</td>
<td></td>
</tr>
<tr>
<td>Constant</td>
<td>0.059***</td>
<td>0.007***</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.029)</td>
<td>(0.007)</td>
<td></td>
</tr>
<tr>
<td>Liberal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relat. policy dist.</td>
<td>0.896***</td>
<td>0.818***</td>
<td>0.943***</td>
</tr>
<tr>
<td></td>
<td>(0.005)</td>
<td>(0.011)</td>
<td>(0.014)</td>
</tr>
<tr>
<td>Inflation</td>
<td>1.735***</td>
<td>2.938***</td>
<td>1.784</td>
</tr>
<tr>
<td></td>
<td>(0.185)</td>
<td>(0.584)</td>
<td>(0.545)</td>
</tr>
<tr>
<td>1992 election</td>
<td>0.808*</td>
<td>0.771</td>
<td>0.793</td>
</tr>
<tr>
<td></td>
<td>(0.080)</td>
<td>(0.159)</td>
<td>(0.174)</td>
</tr>
<tr>
<td>Male</td>
<td>0.493***</td>
<td>0.304***</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.073)</td>
<td>(0.092)</td>
<td></td>
</tr>
<tr>
<td>Age</td>
<td>0.810***</td>
<td>0.632***</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.039)</td>
<td>(0.066)</td>
<td></td>
</tr>
<tr>
<td>Manual worker</td>
<td>0.900</td>
<td>1.235</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.132)</td>
<td>(0.393)</td>
<td></td>
</tr>
<tr>
<td>Constant</td>
<td>0.102***</td>
<td>0.013***</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.048)</td>
<td>(0.012)</td>
<td></td>
</tr>
<tr>
<td>(\text{Var}(\alpha_{\text{Lab.}}))</td>
<td>14.672***</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(2.988)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\text{Var}(\alpha_{\text{Lib.}}))</td>
<td>13.915***</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(2.325)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\text{Cov}(\alpha_{\text{Lab.}}, \alpha_{\text{Lib.}}))</td>
<td>11.441***</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(2.377)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>log likelihood</td>
<td>-1946.269</td>
<td>-1764.331</td>
<td>-138.190</td>
</tr>
<tr>
<td>(N) obs.</td>
<td>2458</td>
<td>2458</td>
<td>494</td>
</tr>
<tr>
<td>(N) groups</td>
<td>1344</td>
<td>1344</td>
<td>247</td>
</tr>
</tbody>
</table>

Notes: * p < .05, ** p < .01, *** p < .001; base outcome: Conservative party; reference categories: 1987 election, female, not manual worker; FEMLOGIT: multinomial logit model with fixed effects; POMLOGIT: pooled multinomial logistic regression; REMLOGIT: multinomial logistic regression with random effects.
6 Conclusion

In this article, I introduce an implementation of multinomial logistic regression with fixed effects as derived by Chamberlain (1980). With this model, it is possible to consistently estimate effects of time-varying regressors on the log-odds of multinomial outcomes when time-invariant unobserved heterogeneity is present. In particular, time-invariant unobserved heterogeneity may be correlated with predictor variables. The implemented ado `femlogit` is applied to real data. In an example with British election panel data, I estimate the effect of perceived distance in the left–right political dimension between a candidate and a voter on voting behavior. The specific advantage of the multinomial logit model with fixed effects in this example is that the effect of policy distance on vote intention is estimated net of all time-invariant voter characteristics that may affect vote intention, perceived policy distance, or both.

7 Acknowledgments

I thank Josef Brüderl, Jeffrey Pitblado, Henning Best, Stephanie Eckman, Michael Herrmann, Andreas Horr, Reinhard Schunck, the participants of the 2011 German Stata Users Group meeting in Bamberg, and an anonymous reviewer for useful comments and suggestions. I thank Sophia Rabe-Hesketh and Anders Skrondal for providing the data used in the examples.

8 References


**About the author**

Klaus Pforr is a project consultant and researcher at GESIS—Leibniz-Institute for the Social Sciences in Mannheim, Germany.
Analysis of partially observed clustered data using generalized estimating equations and multiple imputation

Kathryn M. Aloisio
Smith College
Northampton, MA
kaloisio@gmail.com

Sonja A. Swanson
Harvard School of Public Health
Boston, MA
sswanson@hsph.harvard.edu

Nadia Micali
University College London
London, UK
n.micali@ucl.ac.uk

Alison Field
Harvard School of Public Health
Boston, MA
alison.field@childrens.harvard.edu

Sonja A. Swanson
Harvard School of Public Health
Boston, MA
sswanson@hsph.harvard.edu

Abstract. Clustered data arise in many settings, particularly within the social and biomedical sciences. For example, multiple-source reports are commonly collected in child and adolescent psychiatric epidemiologic studies where researchers use various informants (for instance, parents and adolescents) to provide a holistic view of a subject’s symptoms. Fitzmaurice et al. (1995, American Journal of Epidemiology 142: 1194–1203) have described estimation of multiple-source models using a standard generalized estimating equation (GEE) framework. However, these studies often have missing data because additional stages of consent and assent are required. The usual GEE is unbiased when data are missing completely at random in the context of Little and Rubin (2002, Statistical Analysis with Missing Data [Wiley]). This is a strong assumption that may not be tenable. Other options, such as the weighted GEE, are computationally challenging when missingness is nonmonotone. Multiple imputation is an attractive method to fit incomplete data models while requiring only the less restrictive missing-at-random assumption. Previously, estimation of partially observed clustered data was computationally challenging. However, recent developments in Stata have facilitated using them in practice. We demonstrate how to use multiple imputation in conjunction with a GEE to investigate the prevalence of eating disorder symptoms in adolescents as reported by parents and adolescents and to determine the factors associated with concordance and prevalence. The methods are motivated by the Avon Longitudinal Study of Parents and their Children, a cohort study that enrolled more than 14,000 pregnant mothers in 1991–92 and has followed the health and development of their children at regular intervals. While point estimates for the missing-at-random model were fairly similar to those for the GEE under missing completely at random, the missing-at-random model had smaller standard errors and required less stringent assumptions regarding missingness.
Fitting GEE models using multiply imputed data

Keywords: st0363, ALSPAC study, eating disorders, multiple informants, weighted estimating equations, generalized estimating equations, multiple imputation, missing data, missing at random, missing completely at random

1 Introduction

Clustered data arise in many settings, particularly within the social and medical sciences, and they require sophisticated analytical methods. Standard-error estimates that do not account for association within clusters will be inaccurate and inferences will be invalid (Cannon et al. 2001).

For example, multiple-source reports are commonly collected in child and adolescent psychiatric epidemiologic studies where researchers use various informants (for example, parents and adolescents) to provide a holistic view of a subject’s symptoms. These clustered reports also arise in other settings, such as geriatric studies, school settings, and health services research (Caria et al. 2011).

Several articles have reviewed methods to integrate reports from multiple sources (Fitzmaurice et al. 1995; Horton and Fitzmaurice 2004; Caria et al. 2011). Fitzmaurice et al. (1995) proposed methodology for simultaneously analyzing information from multiple-source outcomes by applying a generalized estimating equation (GEE) approach (Liang and Zeger 1986). GEEs account for the correlation between reports to model the average response for observations sharing covariates.

A practical difficulty in analyzing multiple-source reports is that there is often a substantial amount of missingness. In multiple-source studies, data may be missing from a single source or multiple sources, because additional stages of consent and assent are required. Analyzing data without appropriately accounting for missingness can induce bias and loss of efficiency.

The usual GEE is unbiased whenever missingness is missing completely at random (MCAR), which means that missingness does not depend on observed or unobserved measurements (Little and Rubin 2002). The GEE permits a report to contribute to one equation and not to the other, but using the available-case method may be biased if the missing mechanism is not MCAR (Liang and Zeger 1986).

Xie and Paik (1997) proposed a weighted GEE that handles missingness when the probability of missingness depends on the outcomes or observed covariates. This method assumes the less restrictive missingness mechanism, which was named missing at random (MAR) by Little and Rubin (2002). To fit the weighted GEE, one must estimate the probability of subjects’ being observed, drop all the partially observed subjects, and fit the reweighted model using only the complete cases. Horton et al. (2001) implemented this with multiple-source reports but had to use an ad hoc procedure to account for complex nonmonotone patterns of missingness. The monotone structure is rarely seen in observational studies with many covariates and is absent in the motivating example. Furthermore, accounting for a complex nonmonotone pattern is computationally difficult (Li et al. 2011). Therefore, other approaches are needed.
An alternative approach to this problem implements multiple imputation (MI), a flexible and principled method for fitting incomplete data regression models (Rubin 1987). After specifying an appropriate imputation model, the algorithm “fills in” the missing data with plausible values that account for the uncertainty that comes with using predicted values. The MI method does not require the missingness pattern to be MCAR or monotone.

Simulation studies with longitudinal binary data and missing data have been implemented to assess different analytical approaches, including the usual GEE, the weighted GEE, and MI in conjunction with estimating equations (MI–GEE). Beunckens, Sotto, and Molenberghs (2008) found that using the MI–GEE approach was more successful than using the usual GEE and the weighted GEE approach. DeSouza, Legedza, and Sankoh (2009), Yoo (2010), and Birhanu et al. (2011) expanded the simulation study, and each concluded that MI–GEE outperformed the weighted GEE and is a valid analysis tool for nonnormal and repeated binary responses. Frank Liu and Zhan (2011) undertook a similar simulation study and found contrary evidence for MI–GEE but concluded that the null finding may be due to the misspecification of the imputation model. The flexibility of the MI–GEE allows the imputation model to be adjusted. Lloyd et al. (2013) describe how to undertake estimation for longitudinal regression by using the ice and uvis user-written commands in Stata 11.

Previously, estimation of partially observed clustered data was computationally challenging. However, recent developments in Stata have facilitated their use in practice. This article demonstrates estimation of a GEE model with multiply imputed data by using the mi system in Stata 13.

We first describe the motivating study, the Avon Longitudinal Study of Parents and Children (ALSPAC), a long-running cohort study using parent and adolescent questionnaires to research the health and development of the adolescents. Then we describe how GEE models can be used to fit generalized linear models using available case data. Next we introduce MI and simultaneous estimation of GEE models using multiply imputed data within Stata. Then we fit the GEE models to our motivating data by using both available cases and the imputed data. We conclude by discussing the method, possible extensions, and areas for future research.

2 Example: Multiple source reports of adolescent eating-disorder behaviors

2.1 Study sample

These methods are motivated by data from the ALSPAC, a longitudinal, prospective study of women and pregnancy (Golding et al. 2001; Boyd et al. 2013). All pregnant women living in the area of Avon, UK, who were expected to deliver their babies between 1 April 1991 and 31 December 1992 were invited to take part in the study. Adolescents from 14,541 pregnancies were enrolled. Of these, 12,388 singleton adolescents were alive.
Fitting GEE models using multiply imputed data

at age one and provided the study with complete information on each adolescent’s sex and maternal age. Adolescents and their parents were followed to investigate a range of psychological, physical, and social outcomes.

Parents and adolescents who were still enrolled in the study were sent questionnaires when the adolescent was age 14 and again when age 16. The analytic sample consists of 7,986 adolescents that had at least one adolescent or parent report at age 14 and 16, and it includes fully observed family demographics.

Adolescents completed questions on eating disorder symptoms adapted from the purging behavior assessments in the McKnight Risk Factor Survey and the Youth Risk Behavior Surveillance System Questionnaires (Kann et al. 1996). Adolescents were asked whether they had engaged in eating disorder behaviors in the past year, including binge eating (overeating with loss of control; two questions), vomiting, laxative use, and fasting. Parents completed a questionnaire version of the Eating Disorder Developmental and Well-Being Assessment with no skip rules (Goodman et al. 2000; Ford, Goodman, and Meltzer 2003). Parents were asked whether their teenager had engaged in eating disorder behaviors in the past three months, including binge eating (overeating with loss of control; one question), vomiting, laxative use, and fasting.

To demonstrate, this article will focus on predicting reports of vomiting behavior at age 16. Analyses for other eating disorder symptoms at ages 14 and 16 are reported in Swanson et al. (2014).

Ethical approval for the study was obtained from the ALSPAC Laws and Ethics Committee, the Local Research Ethics Committees, and the Smith College and Amherst College Institutional Review Boards.

2.2 Variables

The questionnaire sent to participants when the adolescents were age 16 asked the parents, “Over the last 3 months, has your study teenager made herself/himself sick to avoid putting on weight?” It also asked the adolescents, “During the past year, how often did you make yourself throw up (vomit) to lose weight or avoid gaining weight?” Because of inconsistency of possible answer options across informants, these two questions were recoded into two dichotomous variables (vomit_p16, vomit_c16) as either any or no endorsement.

For the GEE approach, we needed to reshape our dataset from wide form (one row per subject) to long form (two rows per subject). We created a binary-source variable (adolescent report versus parent report, child) and combined the outcome variables into vomit_16.

The models also included three dichotomous covariates that measured maternal education (A levels or above [college entrance] versus less than A levels, edu_a), adolescent’s sex at birth (female versus male, female), and maternal parity at birth of the adolescent under study (multiparae [any siblings] versus primiparae, multiparae).
Table 1. Prevalence for the covariates

<table>
<thead>
<tr>
<th></th>
<th>Overall</th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(n = 7968)</td>
<td>(n = 3834)</td>
<td>(n = 4134)</td>
</tr>
<tr>
<td>Maternal education</td>
<td>57.7%</td>
<td>57.5%</td>
<td>57.9%</td>
</tr>
<tr>
<td>(less than A levels)</td>
<td>4429/7679</td>
<td>2135/3715</td>
<td>2294/3964</td>
</tr>
<tr>
<td>Parity</td>
<td>46.6%</td>
<td>47.0%</td>
<td>46.2%</td>
</tr>
<tr>
<td>(primiparae)</td>
<td>3594/7714</td>
<td>1749/3724</td>
<td>1845/3990</td>
</tr>
</tbody>
</table>

Table 2. Prevalence for report of adolescent vomiting at age 16

<table>
<thead>
<tr>
<th></th>
<th>Overall</th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent report</td>
<td>0.30%</td>
<td>0.19%</td>
<td>0.41%</td>
</tr>
<tr>
<td>(16/5252)</td>
<td>(5/2578)</td>
<td>(11/2674)</td>
<td></td>
</tr>
<tr>
<td>Adolescent report</td>
<td>4.93%</td>
<td>0.82%</td>
<td>7.78%</td>
</tr>
<tr>
<td>(236/4788)</td>
<td>(16/1962)</td>
<td>(220/2826)</td>
<td></td>
</tr>
</tbody>
</table>

3 Methods

3.1 Notation

Following the notation in Horton and Fitzmaurice (2004), we assume there are $N$ independent subjects, each with an outcome obtained from $J$ sources. Let $Y_{ij}$ represent the dichotomous outcome obtained for the $i$th subject from the $j$th source (with $i = 1, \ldots, N$ and $j = 1, \ldots, J$). The study has two sources ($J = 2$), where $Y_{i1}$ is the first source report (adolescent, $\text{child}=1$), and $Y_{i2}$ is the second source report (parent, $\text{child}=0$). In addition, let $X_{ij}$ be a $p \times 1$ vector of covariates associated with the outcome obtained for the $i$th subject from the $j$th source ($X_{ij}$ contains both source information and subject-specific information). We let $Y_i = (Y_{i1}, \ldots, Y_{ij})'$ be the $J \times 1$ outcome vector for the $i$th subject and $X_i$ be the associated $J \times p$ matrix of covariates.

3.2 Analytic approaches

**GEE model for multiple sources**

If we had only one source, there would be one observation per subject (no clustering), and we could proceed to fit a logistic regression model for the dichotomous outcome or another model from the generalized linear model family. However, the clustered nature of multiple sources, where two reports from the same adolescent are likely to be positively associated, requires a more sophisticated model.
GEEs were first described by Liang and Zeger (1986) and are an attractive method to fit population-averaged regression models for clustered data. The GEE assumes a “working” correlation matrix and uses an empirical variance estimator (also known as a robust or Huber–White or “sandwich” variance) to obtain estimates for the logistic regression model, which accounts for the clustering within subjects. Liang and Zeger (1986) proved that the GEE yields consistent estimates of the regression parameters and of their variances under mild assumptions about dependence and correct specification of the mean model.

The general form for regression models for the mean of some function of \( Y_i \), conditional on both source and risk factors (this setting includes adolescent gender, maternal education, and parity), is given by

\[
g(E(Y_{ij}|X_{ij})) = X'_{ij} \beta
\]

where \( g(\cdot) \) is a known link function. For our setting with a binary outcome, we can set \( g(y) = \log(y/(1-y)) = \logit(y) \) (for example, the logit function). The full model applied to the motivating example would be the following:

\[
\text{logit}\{E(Y_{ij}|X_{ij})\} = \beta_0 + \beta_1 \text{multiparae} + \beta_2 \text{edu}a + \beta_3 \text{female} + \beta_4 \text{child} \\
+ \beta_5 (\text{child} \times \text{female}) + \beta_6 (\text{child} \times \text{multiparae}) \quad (1)
\]

\[
+ \beta_7 (\text{child} \times \text{edu}a)
\]

The coefficients are log odds-ratios, where \( \beta_5, \beta_6, \) and \( \beta_7 \) represent the interaction of the source effect with the three covariates.

Model (1) can be simplified if interactions were found to be nonsignificant. For the predicted prevalence of the vomiting behavior model, we dropped the extraneous interactions (those with \( p \)-values \( \geq 0.05 \)) and refit the model to obtain estimates for a parsimonious model, which retained the gender by source interaction:

\[
\text{logit}\{E(Y_{ij}|X_{ij})\} = \beta_0 + \beta_1 \text{multiparae} + \beta_2 \text{edu}a + \beta_3 \text{female} + \beta_4 \text{child} \\
+ \beta_5 (\text{child} \times \text{female}) \quad (2)
\]

Without the parity by source interaction and maternal education by source interaction in (2), \( \exp(\beta_1) \) and \( \exp(\beta_2) \) are interpreted as odds ratios for parity and maternal education, respectively, within levels of source and gender. We can interpret the interaction term by extracting the equation for parent reports [presented in (3)]; similarly, we can obtain the equation for adolescent reports (4).

\[
\text{logit}\{E(Y_{ij}|X_{ij}, \text{child} == 0)\} = \beta_0 + \beta_1 \text{multiparae} + \beta_2 \text{edu}a + \beta_3 \text{female} \quad (3)
\]

\[
\text{logit}\{E(Y_{ij}|X_{ij}, \text{child} == 1)\} = (\beta_0 + \beta_4) + \beta_1 \text{multiparae} + \beta_2 \text{edu}a \\
+ (\beta_3 + \beta_5) \text{female} \quad (4)
\]

Note that for the adolescent report (4), \( \beta_4 \) is the log odds for additional prevalence for adolescent reports, and \( \beta_5 \) is the additional log odds for female adolescent reports.
3.3 Accounting for missing data

Missing data occur in almost all real-world investigations (Little and Rubin 2002). This was also the case for the ALSPAC study, which is demonstrated using the `miss` option for `tabulate`.

```
. by female: tabulate vomit_c16 vomit_p16, miss
```

<table>
<thead>
<tr>
<th>vomit_c16</th>
<th>vomit_p16</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1673</td>
<td>1946</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>.</td>
<td>888</td>
<td>1,872</td>
</tr>
<tr>
<td>Total</td>
<td>2,573</td>
<td>3,834</td>
</tr>
</tbody>
</table>

```
. by female = 1
```

<table>
<thead>
<tr>
<th>vomit_c16</th>
<th>vomit_p16</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1,986</td>
<td>2,606</td>
</tr>
<tr>
<td>1</td>
<td>135</td>
<td>220</td>
</tr>
<tr>
<td>.</td>
<td>542</td>
<td>1,308</td>
</tr>
<tr>
<td>Total</td>
<td>2,663</td>
<td>4,134</td>
</tr>
</tbody>
</table>

From this output, we observe that 1,688 male adolescents returned completed questionnaires and that 2,130 female adolescents returned completed questionnaires out of the total 7,968 sample subjects. By adding the `miss` option, we show that 4,150 (52%) of the possible sample are missing adolescent, parent, or both reports, regardless of gender. For instance, for male subjects, 7% \(\frac{271}{3834}\) have adolescent reports but are missing parent reports, 23% \(\frac{888 + 2}{3834}\) have missing adolescent reports but have parent reports, and 26% \(\frac{982}{3834}\) are missing both adolescent and parent reports for the questionnaire from age 16. Accounting for the partially observed responses is crucial for obtaining reliable results for future inferences.

There are three concerns that typically arise with missing data: 1) loss of efficiency; 2) complication in data handling and analysis; and 3) bias due to differences between the observed and unobserved data. Next we introduce a nomenclature for missing data.

**Missing-data nomenclature**

For each of the \(N\) subjects, the outcome vector, \(Y\), and the vector of predictors, \(X\), are either observed or missing. We denote \(Y^{\text{obs}}\) as the observed component of the outcome and \(X^{\text{obs}}\) as the observed components of the predictors. Similarly, we denote \(Y^{\text{mis}}\) and \(X^{\text{mis}}\) as the unobserved components of the outcome and predictors, respectively. In addition, \(Z^{\text{obs}} = (Y^{\text{obs}}, X^{\text{obs}})\) and \(Z^{\text{mis}} = (Y^{\text{mis}}, X^{\text{mis}})\) denote the vector of observed variables and missing variables, respectively. We also use \(\gamma\) to denote the regression
parameters. Lastly, we define a set $R$ of response indicators (that is, $R_i = 1$ if the $i$th element of $Z$ is observed, and it equals 0 otherwise).

Little and Rubin (2002) defined classifications for the probability distribution generating the missing data. MCAR is characterized as

$$P(R|Z, \gamma) = P(R|Z^{obs}, Z^{mis}, \gamma) = P(R|\gamma)$$

That is, the probability of being missing is the same for all cases. Heuristically, the reasons for missingness are unrelated to the observed or unobserved data. MCAR is simple but is unlikely to happen in practice.

The mechanism MAR assumes

$$P(R|Z, \gamma) = P(R|Z^{obs}, \gamma)$$

That is, the probability of being missing is the same after conditioning on the observed data. Heuristically, this states that missingness depends on only observed quantities, including outcomes, predictors, and auxiliary variables. Most analyses start with this assumption because it is more likely to happen than MCAR, particularly within datasets containing many variables (Collins, Schafer, and Kam 2001). It is possible to test the MCAR assumption against the alternative hypothesis that missingness is MAR (Diggle and Kenward 1994).

Missing not at random (MNAR) concerns researchers and analysts the most because MNAR means that the probability of data being missing varies for reasons that are unknown to the researcher (missingness is related to the unobserved quantities). Symbolically, $P(R|Z)$ cannot be simplified, and it must be modeled as part of the likelihood. Little and Rubin (2002) call this “nonignorable”. While MNAR missingness is important when undertaking sensitivity analyses, we will not consider it further.

The pattern of missingness, monotone versus nonmonotone, can also influence how we address missing data. A dataset is said to have a monotone-missing pattern when the variables in the dataset can be arranged in a stair-step pattern (that is, nonincreasing or nondecreasing) when missingness on one implies missingness on the other (Little and Rubin 2002). The monotone pattern is generally uncommon with observational studies, as with the motivating study, where we have some subjects missing a parent report and others missing an adolescent report.

We can use misschk (Long and Freese 2014) to display the missingness pattern for a subset of the variables used in our motivating example.
. misschk female edua multiparae vomit_c16 vomit_p16

Variables examined for missing values

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th># Missing</th>
<th>% Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>female</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>edua</td>
<td>289</td>
<td>3.6</td>
</tr>
<tr>
<td>3</td>
<td>multiparae</td>
<td>254</td>
<td>3.2</td>
</tr>
<tr>
<td>4</td>
<td>vomit_c16</td>
<td>3180</td>
<td>39.9</td>
</tr>
<tr>
<td>5</td>
<td>vomit_p16</td>
<td>2716</td>
<td>34.1</td>
</tr>
</tbody>
</table>

Missing for which variables?

<table>
<thead>
<tr>
<th></th>
<th>Freq.</th>
<th>Percent</th>
<th>Cum.</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>234</em></td>
<td>32</td>
<td>0.40</td>
<td>0.40</td>
</tr>
<tr>
<td><em>234</em></td>
<td>12</td>
<td>0.15</td>
<td>0.55</td>
</tr>
<tr>
<td><em>23</em></td>
<td>18</td>
<td>0.23</td>
<td>0.78</td>
</tr>
<tr>
<td><em>23</em></td>
<td>17</td>
<td>0.21</td>
<td>0.99</td>
</tr>
<tr>
<td><em>245</em></td>
<td>79</td>
<td>0.99</td>
<td>1.88</td>
</tr>
<tr>
<td><em>24</em></td>
<td>40</td>
<td>0.50</td>
<td>2.38</td>
</tr>
<tr>
<td><em>25</em></td>
<td>50</td>
<td>0.63</td>
<td>3.11</td>
</tr>
<tr>
<td><em>2</em></td>
<td>41</td>
<td>0.51</td>
<td>3.63</td>
</tr>
<tr>
<td>___345</td>
<td>57</td>
<td>0.72</td>
<td>4.34</td>
</tr>
<tr>
<td>__<em>34</em></td>
<td>27</td>
<td>0.34</td>
<td>4.68</td>
</tr>
<tr>
<td>__<em>3</em></td>
<td>23</td>
<td>0.29</td>
<td>4.97</td>
</tr>
<tr>
<td>__<em>3</em></td>
<td>68</td>
<td>0.85</td>
<td>5.82</td>
</tr>
<tr>
<td>__<em>45</em></td>
<td>1,578</td>
<td>19.80</td>
<td>25.63</td>
</tr>
<tr>
<td>__<em>4</em></td>
<td>1,355</td>
<td>17.01</td>
<td>42.63</td>
</tr>
<tr>
<td>____5</td>
<td>879</td>
<td>11.03</td>
<td>53.66</td>
</tr>
<tr>
<td>____5</td>
<td>3,692</td>
<td>46.34</td>
<td>100.00</td>
</tr>
</tbody>
</table>

Total 7,968 100.00

Missing for how many variables?

<table>
<thead>
<tr>
<th></th>
<th>Freq.</th>
<th>Percent</th>
<th>Cum.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3,692</td>
<td>46.34</td>
<td>46.34</td>
</tr>
<tr>
<td>1</td>
<td>2,343</td>
<td>29.41</td>
<td>75.74</td>
</tr>
<tr>
<td>2</td>
<td>1,735</td>
<td>21.77</td>
<td>97.52</td>
</tr>
<tr>
<td>3</td>
<td>166</td>
<td>2.08</td>
<td>99.60</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>0.40</td>
<td>100.00</td>
</tr>
</tbody>
</table>

Total 7,968 100.00

Note that the most common missing patterns include subjects missing both adolescent and parent reports (20%, n = 1578), subjects missing just the adolescent report (17%, n = 1355), and subjects missing just the parent report (11%, n = 879). However, we do not have a monotone-missingness pattern, because for each of the covariates (parity and maternal education), there are cases missing adolescent, parent, or both reports.

**Available-case method**

The available-case method includes all cases where the variable of interest is present. This method is more efficient than complete-case analyses, where any case with a missing value is removed. In the motivating study, there were n = 7968 available cases versus
Fitting GEE models using multiply imputed data

$n = 3692$ complete cases, as shown in the missing-patterns table above. The available-case method is also unbiased when missingness is MCAR. However, complications can arise because the analytic sample base changes from model to model and may lead to problems of comparability.

**Weighted estimating equations**

Weighted estimating equations (Xie and Paik 1997; Horton et al. 2001; Li et al. 2011) are an attractive approach if missingness is monotone. However, using them is not feasible in this setting, even for modeling parent and adolescent reports for one age, because some subjects have missing adolescent reports while others have missing parent reports. However, weighted estimating equations are supported in Stata 13. For more details, see `help weight`.

**MI**

MI is a principled method used to account for missing data (Rubin 1976). It involves a three-step approach for fitting incomplete data regression models. First, it creates plausible values for missing observations that reflect uncertainty about the nonresponse model. These values are used to “fill in” or impute the missing values (generally under a MAR assumption). This process is repeated, which results in the creation of several “completed” datasets. Second, each of these datasets is analyzed using complete-data methods. Finally, the results are combined, which allows the uncertainty regarding the imputation to be considered (Little and Rubin 2002). Because increasing the number of imputed datasets minimizes variability introduced into the results because of the imputation process (Horton and Lipsitz 2001; White, Royston, and Wood 2011; van Buuren 2012), we recommend a set of 25 imputations, though more are computationally possible.

**Specifying imputation model.** MI requires the analyst to provide an appropriate specification of the imputation model. If this model is misspecified, there is potential for bias (White, Royston, and Wood 2011). In general, the imputation model must be compatible with the model used for the analysis, with all potential covariates and important higher-order associations included (Little and Rubin 2002). For example, in model (2), we want to assess the source by gender interaction with reported instances of vomiting. Even though gender is fully observed, we need to include gender in the imputation model because we include gender effects in the analysis model. Also, to preserve the source by gender interaction, we have to account for the interaction term. We did this by stratifying the imputation model by gender (Royston 2005), though this could also have been accomplished by including the interaction when specifying custom prediction equations (see `help mi impute chained`).

In addition to all the variables that can be used in the analysis model, any auxiliary variables that may contain information about missing data should be included. For our model, we included a measure for self-reported body mass index, the mother’s age at delivery, and the adolescent’s age at the time of reporting. Furthermore, the
outcome variable should always be present in the imputation model to obtain valid results (Moons et al. 2006). By including all the variables necessary for the model and any auxiliary variables that may contain information about missing data, the MAR assumption becomes more plausible, and the quality of the imputed values improves (Collins, Schafer, and Kam 2001).

**Specifying imputation method.** The choice of imputation method depends on the pattern of missing values. As opposed to having a monotone-missing pattern, our data have an arbitrary missing pattern. When a pattern of missing values is arbitrary, iterative methods are used to fill in missing values. To accommodate our arbitrary missing-value patterns, we imputed the data using chained equations with a variable-by-variable approach. The imputation model is specified separately for each variable and involves the other variables as predictors. At each stage of the algorithm, an imputation is generated for all the missing values in a given variable, and this imputed variable is used to impute the next variable. This process repeatedly imputes missing values by using a Gibbs sampling procedure until the process reaches convergence. For this example, we used 25 iterations.

**Combining complete-case results.** The last step of the imputation method uses “Rubin’s rules” to combine the repeated-imputation results, where the total variance stems from the following three sources (Little and Rubin 2002):

1. The variance is a result of taking a sample rather than observing the entire population. This is the conventional statistical measure of variability.
2. The extra variance is caused by missing values in the sample.
3. The extra simulation variance is a result of the estimate being estimated for a finite number of imputations.

### 4 Application in Stata

MI can be used in combination with the estimation of a wide variety of models, including the GEE model, using the `mi` system in Stata 13.

To use the `mi` system, we begin by reading in the dataset and creating the analytic set. We include additional variables from the cohort study in the imputation model to make the MAR assumption more plausible (Collins, Schafer, and Kam 2001).

```stata
use alspac_informant, clear
. keep vomit_c14 vomit_p14 vomit_c16 vomit_p16
. keep lax_c14 lax_p14 lax_c16 lax_p16
. keep fast_c14 fast_p14 fast_c16 fast_p16
. keep binge_c14 binge_p14 binge_c16 binge_p16
. keep anyedsx_c14 anyedsx_p14 anyedsx_c16 anyedsx_p16
. keep thin_c14 thin_p14
. keep educ multiparae m_age_at_delivery female weightkg heightm c_age_at_report
. keep bmi cid_153a
```
4.1 Registering variables

Next we need to set how Stata should add additional imputations. We chose to use the marginal long (\texttt{mlong}) data structure because it uses slightly less memory than the wide (\texttt{wide}) data structure. However, the wide format is slightly faster.

\begin{verbatim}
    . mi set mlong
\end{verbatim}

Then we register each of the variables within the dataset as either variables to impute or variables to not impute.

The variables that must be imputed require registration:

\begin{verbatim}
    . mi register imputed vomit_c14 vomit_p14 vomit_c16 vomit_p16 lax_c14 lax_p14
    > lax_c16 lax_p16 fast_c14 fast_p14 fast_c16 fast_p16
    > thin_c14 thin_p14 binge_c14 binge_p14 binge_c16 binge_p16
    > anyedsx_c14 anyedsx_p14 anyedsx_c16 anyedsx_p16
    > edu multiparae weightkg heightm c_age_at_report bmi
    (6009 m=0 obs. now marked as incomplete)
\end{verbatim}

The variables that do not require imputation but will be used in the imputation model are registered as regular variables.

\begin{verbatim}
    . mi register regular m_age_at_delivery female
\end{verbatim}

With the added covariates, we redisplay the table from \texttt{misschk} listing missingness for different numbers of variables.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1,959</td>
<td>24.59</td>
<td>24.59</td>
</tr>
<tr>
<td>1</td>
<td>345</td>
<td>4.33</td>
<td>28.92</td>
</tr>
<tr>
<td>2</td>
<td>106</td>
<td>1.33</td>
<td>30.25</td>
</tr>
<tr>
<td>(output omitted)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>7</td>
<td>0.09</td>
<td>99.94</td>
</tr>
<tr>
<td>27</td>
<td>5</td>
<td>0.06</td>
<td>100.00</td>
</tr>
<tr>
<td>Total</td>
<td>7,968</td>
<td>100.00</td>
<td></td>
</tr>
</tbody>
</table>

Note that with these 31 variables, there are only 1,959 complete cases. Three variables are completely observed: the ID, the gender, and the age of mother at the time of delivery.

4.2 Imputation model specification

We then create the imputed datasets in Stata by using \texttt{mi impute}. This requires us to specify the imputation model. We must first select the imputation method. For univariate imputation, where the pattern of missingness is monotone, we can choose from a variety of imputation models based on the type of variable. For example, \texttt{mi}
**impute regress** will fit a linear regression model for a continuous variable or **mi impute poisson** for a count variable.

For multivariate imputation with different types of variables (that is, a mixture of continuous and discrete), the situation is more complicated. If the pattern of missingness is monotone, we can use **mi impute monotone** to assign an imputation method to each variable. If there is an arbitrary missing pattern (as in the present analysis), we can use **mi imputed mvn** for multivariate normal variables or **mi impute chained** for the chained-equation method. Table 3 lists these options and other options that can be selected as the imputation method.

Table 3. MI methods available within Stata 13

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Univariate</td>
<td></td>
</tr>
<tr>
<td><strong>regress</strong></td>
<td>Linear regression</td>
</tr>
<tr>
<td><strong>pmm</strong></td>
<td>Predictive mean matching</td>
</tr>
<tr>
<td><strong>truncreg</strong></td>
<td>Truncated regression</td>
</tr>
<tr>
<td><strong>intreg</strong></td>
<td>Interval regression</td>
</tr>
<tr>
<td><strong>logit</strong></td>
<td>Logistic regression</td>
</tr>
<tr>
<td><strong>ologit</strong></td>
<td>Ordered logistic regression</td>
</tr>
<tr>
<td><strong>mlogit</strong></td>
<td>Multinomial logistic regression</td>
</tr>
<tr>
<td><strong>poisson</strong></td>
<td>Poisson regression</td>
</tr>
<tr>
<td><strong>nbreg</strong></td>
<td>Negative binomial regression</td>
</tr>
<tr>
<td>Multivariate</td>
<td></td>
</tr>
<tr>
<td><strong>monotone</strong></td>
<td>Sequential imputation using a monotone missing pattern</td>
</tr>
<tr>
<td><strong>chained</strong></td>
<td>Sequential imputation using chained equations</td>
</tr>
<tr>
<td><strong>mvn</strong></td>
<td>Multivariate normal regression</td>
</tr>
</tbody>
</table>

Because our data setting did not feature monotone missingness and the study variables were not normally distributed, we adopted the chained-equation approach (Raghunathan et al. 2001; White, Royston, and Wood 2011; van Buuren 2012). Using 25 chains for 25 iterations, we fit a linear regression model **regress** for the incomplete continuous variables and used predicted mean matching (**pmm**) for the binary variables. Predicted mean matching is similar to the regression method except that for each missing value, it imputes a value randomly drawn from a set of observed values whose predicted values are closest to the predicted value for the missing value from the simulated regression model. Generally, predicted mean matching is used for continuous variables. However, predictive mean matching proves to be unbiased for dichotomous variables, it ensures that imputed values are plausible, and it may be more appropriate if the normality assumption is violated (Horton, Lipsitz, and Parzen 2003).

Both sets of models included all symptoms from both sources at each age and other covariates stratified by gender to account for the interaction (StataCorp 2013).
Fitting GEE models using multiply imputed data

```
mi impute chained (regress) weightkg heightm c_age_at_report bmi ///
(pmm) vomit_c14 vomit_p14 vomit_c16 vomit_p16 ///
lax_c14 lax_p14 lax_c16 lax_p16 ///
fast_c14 fast_p14 fast_c16 fast_p16 ///
thin_c14 thin_p14 ///
binge_c14 binge_p14 binge_c16 binge_p16 ///
anyedsx_c14 anyedsx_p14 anyedsx_c16 anyedsx_p16 ///
edua multiparae = m_age_at_delivery, ///
dots noisily add(25) by(female) augment
```

This model was implemented on an Intel® Core™2 Duo Processor and took approximately two hours.

4.3 GEE with imputed datasets

With the imputed datasets, complete-case methods can be used to estimate models using `mi estimate`. Stata supports estimation of many regression models with imputed data, including linear regression models, binary-response regression models, count-response regression models, ordinal-response regression models, categorical-response regression models, quantile regression models, survival regression models, panel-data models, and survey regression models. The present study uses `mi estimate xtgee` to fit the GEE models because of the clustering within subjects.

To preserve associations between the parent and adolescent reports, we imputed the data in wide form (one row per subject). However, to fit the model, we need to reshape our datasets from wide form to long form (two rows per subject). This is easy to do using the post `mi` data manipulation commands. To clarify the process, we will display the data for the first five subjects. We can select the original dataset with `mi xeq 0`.

```
. mi xeq 0: list id female edu multiparae vomit_c16 vomit_p16 if id < 6
m=0 data:
-> list id female edu multiparae vomit_c16 vomit_p16 if id < 6
```

<table>
<thead>
<tr>
<th>id</th>
<th>female</th>
<th>edu</th>
<th>multiparae</th>
<th>vomit_c16</th>
<th>vomit_p16</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Then we rename the outcome variables for the `reshape` command so that \( j = 1 \) indicates an adolescent report and \( j = 0 \) indicates a parent report.

```
. mi rename vomit_c16 vomit_161
. mi rename vomit_p16 vomit_160
```
K. M. Aloisio, N. Micali, S. A. Swanson, A. Field, and N. J. Horton

```
   . mi reshape long vomit_16, i(cid_153a) j(child)
  reshaping m=0 data ...
  (note: j = 0 1)
Data wide -> long

   Number of obs. 7968 -> 15936
   Number of variables 35 -> 35
   j variable (2 values) -> child
   xij variables: vomit_160 vomit_161 -> vomit_16

Now we have doubled the number of rows (15,936) in the long format. We then display
the same first five subjects in the long format.

   . mi xeq 0: sort id; list id female edua multiparae vomit_16 if id <6
   m=0 data:
   -> sort id
   -> list id female edua multiparae child vomit_16 if id <6

<table>
<thead>
<tr>
<th>id</th>
<th>female</th>
<th>edu</th>
<th>multiparae</th>
<th>child</th>
<th>vomit_16</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>.</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Recall that (2) includes the gender by source interaction. We recode a new variable
using imputed data by implementing mi passive. We named the interaction variable
femchild.

   . mi passive: generate femchild = female*child

Before fitting the model, we must declare the type of complex data (for example,
mi stset for survival data or mi svyset for survey data). For panel data, we use the
xtset command. Note that we are paneling on the adolescent ID variable.

   . mi xtset cid_153a
      panel variable: cid_153a (balanced)
Then we proceed to fit (2).

```
. mi estimate: xtgee vomit_16 multiparae edua female child femchild,
>    family(binomial) link(logit) corr(indep)
```

Multiple-imputation estimates
Imputations = 25
GEE population-averaged model
Number of obs = 15936
Group variable: cid_153a Number of groups = 7968
Link: logit Obs per group: min = 2
Family: binomial avg = 2.0
Correlation: independent max = 2
Scale parameter: 1

Average RVI = 0.6721
Largest FMI = 0.4821
DF adjustment: Large sample
DF: min = 107.46
avg = 143.06
max = 208.34
Model F test: Equal FMI F( 5, 642.9) = 46.50
Within VCE type: Conventional Prob > F = 0.0000

|          | Coef. | Std. Err. | t     | P>|t|  | [95% Conf. Interval] |
|----------|-------|-----------|-------|------|----------------------|
| vomit_16 |       |           |       |      |                      |
| multiparae         | .2721927 | .1302315  | 2.09  | 0.038 | .0153082 .5290773   |
| edua         | .1300673 | .1272539  | 1.02  | 0.308 | -.120803 .3809375  |
| female       | .2884962 | .5023234  | 0.57  | 0.567 | -.7072522 1.284245  |
| child        | .9982947 | .4376222  | 2.28  | 0.024 | .1322118 1.864378   |
| femchild     | 1.975446 | .5530279  | 3.57  | 0.001 | .8798404 3.071051   |
| _cons        | -5.883525| .3970033  | 14.82 | 0.000 | -6.67001 -5.09704   |

The model indicates that when we control for other factors, the odds for exhibiting vomiting behavior for an adolescent with siblings is 1.31 [95% confidence interval (CI) 1.02–1.70] times the odds for an only-child adolescent exhibiting vomiting behavior. Maternal education was found to not be significantly associated with vomiting behavior (odds ratio 1.14; [95% CI 0.89–1.46]) after we controlled for other factors.

To interpret the gender by source interaction, we calculated the four predicted probabilities using (3) and (4) with the other covariates set to 0 and the inverse logit function \( \text{invlogit}(\beta) = \exp(\beta) / (1 + \exp(\beta)) \). We calculated the predicted probability for the male adolescent report \( \text{invlogit}(\beta_0 + \beta_4) \), 0.8%, [95% CI 0.4%–1.1%]; the male’s parent report \( \text{invlogit}(\beta_0) \), 0.3%, [95% CI 0.08%–0.5%]; the female adolescent report \( \text{invlogit}(\beta_0 + \beta_3 + \beta_4 + \beta_5) \), 6.8%, [95% CI 5.3%–8.3%]; and the female’s parent report \( \text{invlogit}(\beta_0 + \beta_3) \), 0.4%, [95% CI 0.1%–0.6%].

From these, we can determine important distinct patterns. First, estimates for vomiting are higher when vomiting is reported by the adolescent instead of his or her parent for both male and female adolescents. In addition, for adolescent reporting, there is a significant difference between females and males reporting endorsement of vomiting behaviors. However, there is not a significant gender difference for parent reporting. This result has implications for our understanding of the diagnosis and the prevalence of reported symptoms, as discussed in more detail by Swanson et al. (2014).

To compare these results with an available case model, we can fit the model to only the original dataset by using `mi xeq 0`. 
When the two methods are compared, the estimates are similar, and the standard errors from the MI model assuming MAR are consistently smaller than the standard errors for the MCAR model.

### 5 Discussion

Clustered data with partially observed responses and predictors arise in many situations. In this article, we have detailed how to account for clustering when MI is used to account for missingness.

Multiple-source data often occur when one analyzes studies with complex survey designs. Along with clustering, stratification and sampling weights must be considered in the analysis. This can be done in Stata by using the survey design tools (Horton and Fitzmaurice 2004).

Many analytic approaches rely on the accuracy of the assumptions associated with the proposed method. Negligence or inaccurate analysis of the collected data can introduce bias. Assumptions that missingness is MAR are inherently unverifiable without auxiliary information. When one uses methods that incorporate other variables associated with missingness and responses, the possibility of bias is reduced, and the data are represented more accurately.

The GEE model is attractive because it can account for clustering or repeated measures induced by longitudinal data. However, the assumption of MCAR is very restric-
Fitting GEE models using multiply imputed data

tive because reasons for missingness are generally more complex than just being due to chance.

The weighted GEE loosens the often implausible MCAR missingness assumption. If a weighted model were feasible, it could be incorporated using survey weights, as described by Horton and Fitzmaurice (2004). However, the requirement that the patterns of missing be monotone is a major limitation. The use of MI is attractive because it can incorporate auxiliary variables to make MAR more tenable, and it does not require monotone missingness. One disadvantage of using MI is that it requires additional work to specify the imputation model. Further research could make it easier for users to specify imputation models.

While our estimates from the multiply imputed data were similar to those found using the GEE under MCAR, the MAR model had smaller standard errors and less restrictive assumptions regarding missingness. The ability to fit clustered data models within MI provides great flexibility for analysts. This principled analytic method was once limited by computational access, but, as we demonstrated, it is now readily available within general-purpose statistical software.

6 Acknowledgments

We are extremely grateful to all the families who took part in this study, to all the midwives who helped recruit participants, and to the whole ALSPAC team, which includes interviewers, computer and laboratory technicians, clerical workers, research scientists, volunteers, managers, receptionists, and nurses. The UK Medical Research Council (Grant ref: 74882), the Wellcome Trust (Grant ref: 076467), and the University of Bristol provide core support for ALSPAC. This research was specifically funded by grant R01-MH087786-04 from the National Institutes of Health and the Smith College Borie and Tomlinson Funds. The views expressed in this publication are those of the authors and not necessarily those of the National Health Service, the National Institute for Health Research, or the Department of Health. The funders had no involvement in any aspect of the study.

7 References


Fitting GEE models using multiply imputed data


StataCorp. 2013. *Stata 13 Multiple-Imputation Reference Manual*. College Station, TX: Stata Press.


**About the authors**

Kathryn Aloisio graduated from Smith College, Northampton, MA, with a BA in mathematics and statistics and is currently a master’s student in the Department of Mathematics and Statistics at the University of Massachusetts, Amherst, MA. Her research involves statistical computing for the analysis of clustered and incomplete data.

Sonja Swanson is a doctoral candidate in the Department of Epidemiology at the Harvard School of Public Health, Boston, MA. Her current research focuses on psychiatric epidemiology with a particular interest in improving methods for measurement and classification of psychiatric disorders.

Nadia Micali is a senior lecturer in child and adolescent psychiatry at the Institute of Child Health, University College London and a psychiatrist at Great Ormond Street Hospital, London, UK. She is a trained epidemiologist, and her research focuses on eating disorder epidemiology, risk factors for eating disorders, and developmental risk for psychiatric disorders.

Alison E. Field is an associate professor of pediatrics at Harvard Medical School and an associate professor in the Department of Epidemiology at the Harvard School of Public Health. Her research focuses on classification, risk factors, and the course of obesity and eating disorders.

Nicholas Horton is a professor of statistics in the Department of Mathematics and Statistics at Amherst College, Amherst, MA. His research interests involve the development and dissemination of methods for the analysis of clustered and incomplete data.
Lee (2009) treatment-effect bounds for nonrandom sample selection

Harald Tauchmann
University of Erlangen-Nuremberg,
Rheinisch-Westfälisches Institut für Wirtschaftsforschung (RWI),
and Centre of Health Economics Research (CINCH)
Nürnberg, Germany
harald.tauchmann@fau.de

Abstract. Nonrandom sample selection may render estimated treatment effects biased even if assignment of treatment is purely random. Lee (2009, Review of Economic Studies, 76: 1071–1102) proposes an estimator for treatment-effect bounds that limit the possible range of the treatment effect. In this approach, the lower and upper bound correspond to extreme assumptions about the missing information that are consistent with the observed data. In contrast to conventional parametric approaches to correcting for sample-selection bias, Lee’s bounds estimator rests on very few assumptions. I introduce the new command leebounds, which implements the estimator in Stata. The command allows for several options, such as tightening bounds by using covariates.

Keywords: st0364, leebounds, nonparametric, randomized trial, sample selection, attrition, bounds, treatment effect

1 Introduction

Random assignment of treatment provides an ideal setting for identifying treatment effects. Most prominent randomized trials are designed to generate a situation where randomness of treatment is guaranteed, ruling out any potential endogeneity bias. However, this ideal setting can easily be distorted by nonrandom sample attrition. For example, attrition may happen when participants dropout from a program, when a researcher is denied information on the outcome variable, or when death occurs during a clinical trial. While treatment is purely random in the original population, this does not hold for the actual estimation sample if attrition is linked to the treatment status, which potentially leads to attrition bias with perhaps unknown direction.

Parametrically correcting for attrition and selection bias has become a standard procedure in applied empirical research, rendering the seminal method by Heckman (1976, 1979) a workhorse of applied econometrics. This procedure is implemented in Stata by the heckman command. However, this parametric approach has been criticized for relying on restrictive assumptions, joint normality in particular, and for being vulnerable to misspecification (Puhani 2000; Grasdal 2001), which has led to the development of semi-parametric approaches (Ichimura and Lee 1991; Ahn and Powell 1993). Though these estimators rely on less restrictive distributional assumptions, valid exclusion restrictions
are even more essential. More recently, researchers have proposed bound estimators that require very few assumptions and do not rely on valid exclusion restrictions. Rather than correcting point estimates for potential bias, these estimators determine an interval for the true treatment effect. The interval is based on extreme assumptions about the impact of selection on the estimated effect that are consistent with the data. One such estimator is Horowitz and Manski (2000). This approach does not involve any assumption about the selection mechanism; however, it is applicable only to outcome variables that are bounded to a certain interval because missing information is imputed on the basis of minimal and maximal possible values. This impedes its application to numerous problems and regularly yields very wide bounds.

In this article, I introduce the new command `leebounds`, which facilitates the estimation of alternative bounds proposed by Lee (2009). These alternative bounds impose more structure on the assumed selection mechanism and allow for outcome variables with unbounded support while often yielding more narrow bounds. Thereby, `leebounds` complements the contributions of Beresteanu and Manski (2000) and Palmer et al. (2011), who have already made other bounds estimators available to Stata users. Beresteanu and Manski (2000) provide Stata code for the bounds estimators introduced by Manski (1990) and add further refinements (Manski 1994, 1995, 1997; Manski and Pepper 2000) to the original approach. Unlike Lee’s estimator, Manski’s bounds are meant to obtain treatment-effect bounds under (nonrandom) treatment selection. Palmer et al. (2011) introduce a Stata command for the bounds estimator developed by Balke and Pearl (1997), which is closely related to Manski’s estimators. Here the focus is on estimating treatment-effect bounds under imperfect compliance with a randomly assigned treatment.

In the following section, I summarize Lee’s bounds estimator. In section 3, I describe the syntax of `leebounds`. In section 4, I illustrate the application of `leebounds`. In section 5, I conclude the article.

2 The Lee (2009) bounds estimator

2.1 The intuition behind the estimator

Lee (2009) proposes a bounds estimator that estimates an interval for the true value of the treatment effect in the presence of nonrandom sample selection. The estimator rests on only two assumptions: random assignment of treatment and monotonicity. The latter implies that assignment to the treatment group can affect attrition in only one direction. That means that besides observations for which the outcome variable is observed irrespective of the assigned treatment status, the actual estimation sample includes either observations where the outcome is observed because of receiving the treatment or observations where the outcome is observed because of not receiving the treatment, but not both simultaneously.
Lee treatment-effect bounds

The bounds estimator trims either the treated or the nontreated observations so that the share of observations with observed outcome is equal for both groups. Trimming is either from above or from below. This corresponds to two extreme assumptions about missing information that are consistent with the observed data and a one-sided selection mechanism. That is, in the group that suffers less from attrition, either the largest or the smallest values of the outcome are regarded as “excess observations” and are excluded from the analysis. This implies that the treatment effect on those that never suffer from attrition is subject to estimation. In this article, I focus on the practical issue of how estimates for the bounds are calculated and, in particular, on how this procedure is implemented in Stata; for more theory, readers can refer to Lee (2009).

2.2 Estimation

Estimating treatment-effect bounds as suggested by Lee (2009) is computationally straightforward. Only a raw group mean and two trimmed group means of the outcome variable need to be calculated. Here \( Y_i \) denotes the outcome, \( T_i \) is a binary treatment indicator, and \( S_i \) is a binary selection indicator, with \( S_i = 0 \) indicating attritors for which \( Y_i \) is not observed. As usual, \( i \) indexes observations. The shares of observations with observed outcome in the treatment group, \( q_T \), and its counterpart for the control group, \( q_C \), can then be written as

\[
q_T = \frac{\sum_i 1(T_i = 1, S_i = 1)}{\sum_i 1(T_i = 1)} \\
q_C = \frac{\sum_i 1(T_i = 0, S_i = 1)}{\sum_i 1(T_i = 0)}
\]

Here \( 1(\cdot) \) denotes the indicator function. To simplify notation, we will consider the case \( q_T > q_C \); that is, the treatment group suffers less from attrition.\(^1\) Then

\[
q = \frac{q_T - q_C}{q_T}
\]

and \( 1 - q \) determines the quantiles at which the distribution of \( Y \) in the treatment group are trimmed to exclude extreme values of \( Y \) from the analysis. Hence,

\[
y_q^T = G_{Y|T=1,S=1}^{-1}(q) \\
y_{1-q}^T = G_{Y|T=1,S=1}^{-1}(1-q)
\]

determine the marginal values \( y_q^T \) and \( y_{1-q}^T \) of the outcome that enter the trimmed means, with \( G_Y^{-1} \) denoting the inverse empirical distribution function of \( Y \).

---

\(^1\) For the opposite case of \( q_T < q_C \), all arguments hold symmetrically, with \( q \) being defined as \( (q_C - q_T)/q_C \), the control group being trimmed at \( y_q^C \) and \( y_{1-q}^C \), respectively, and the treatment group remaining untrimmed. For \( q_T = q_C \), both the upper and the lower bound coincide with the difference in raw means.
Using this notation, we calculate estimates for the upper bound and the lower bound as

\[ \hat{\theta}_{\text{upper}} = \frac{\sum_i 1 (T_i = 1, S_i = 1, Y_i \geq y_q^T) Y_i}{\sum_i 1 (T_i = 1, S_i = 1, Y_i \geq y_q^T)} - \frac{\sum_i 1 (T_i = 0, S_i = 1) Y_i}{\sum_i 1 (T_i = 0, S_i = 1)} \]  

(2)

\[ \hat{\theta}_{\text{lower}} = \frac{\sum_i 1 (T_i = 1, S_i = 1, Y_i \leq y_{1-q}^T) Y_i}{\sum_i 1 (T_i = 1, S_i = 1, Y_i \leq y_{1-q}^T)} - \frac{\sum_i 1 (T_i = 0, S_i = 1) Y_i}{\sum_i 1 (T_i = 0, S_i = 1)} \]  

(3)

Lee (2009) considers a purely continuous outcome variable \( Y \). Yet, especially in survey data, variables that are inherently continuous are often imprecisely reported, resulting in “ties” in the observed outcome data. Monthly disposable income can serve as an example, for individuals tend to report a round number, such as $1,000 or $1,500. Such ties may violate the intuition behind (2) and (3) if the marginal values \( y_q^T \) and \( y_{1-q}^T \) are frequent. For this reason, leebounds excludes the \( \lfloor N_S^T \rfloor \) (rounded down to the nearest integer) smallest—respectively, largest—values of \( Y \) when calculating the trimmed means. Here \( N_S^T \) denotes the number of observations in the treatment group for which the outcome variable is observed. This means that a certain fraction of the observations that exhibit the marginal values \( y_q^T \) and \( y_{1-q}^T \) enter the trimmed means. With no ties in \( Y \), this procedure coincides with (2) and (3).

### 2.3 Tightening bounds

Estimating Lee (2009) bounds does not involve any covariates. This corresponds to the assumption of random assignment of treatment, under which the differences in conditional and unconditional expectations of \( Y \) coincide. Yet covariates that are determined before treatment can be used to tighten treatment-effect bounds. Covariates that have some explanatory power for attrition are used to split the sample into cells, and bounds are separately calculated for each cell. Finally, a weighted average of cells’ bounds is computed. The appropriate weights are the probabilities of cell membership for those that never suffer from attrition (Lee 2009, 1094). These probabilities are unknown. However, because of random assignment of treatment and monotonicity, they can consistently be estimated by \( \sum_i 1 (J_i = 1, S_i = 1, T_i = 0) / \sum_i 1 (S_i = 1, T_i = 0) \) for each cell \( J \), where \( J_i = 1 \) indicates membership in \( J \). Lee (2009) shows that such averaged bounds are tighter than those that do not use any covariates (Lee 2009, 1086).2 Tightening bounds is offered by leebounds as an option.

Technically, only a limited number of discrete variables can be used for tightening, because the number of observations and the joint distribution of treatment status and selection must allow for estimating the bounds for each cell. Thus estimation regularly fails if a large number of covariates are used. Tightening could also fail if the control group suffers relatively more from attrition for some cells, while attrition is more frequent.

---

2. The proof is for the population parameters, not for their sample analogs. Hence, especially for ill-suited covariates, estimated bounds may fail in getting tighter with option tight().

3. In practice, continuous variables (for example, age) must be transformed into categorical ones (age classes).
in the treatment group for other cells. Because of sampling error, this will frequently occur if the sample is split into too many cells. \(^4\) \texttt{leebounds} checks for this, issues a warning if it detects a selection pattern that is heterogeneous across cells, and saves a macro that indicates the type of the selection pattern.

### 2.4 Standard errors and inference

Estimates for the treatment-effect bounds are subject to sampling error. Lee (2009, 1088) provides analytic standard errors for them; we refer to the original paper for details about calculating standard errors. Analytical standard errors (or, alternatively, bootstrapped standard errors) are implemented in \texttt{leebounds}. Using these standard errors, one can determine “naive” confidence intervals that cover the interval \([g_{\text{lower}}, g_{\text{upper}}]\) with probability \(1 - \alpha\). Interestingly, on the basis of Imbens and Manski (2004), Lee (2009, 1089) also derives a confidence interval for the treatment effect itself, that is, the scalar parameter of ultimate interest. This interval is tighter than the combined confidence interval for \(g_{\text{lower}}\) and \(g_{\text{upper}}\). It captures both uncertainty about the selection bias and uncertainty about the sampling error. \texttt{leebounds} optionally estimates the confidence interval for the treatment effect.

### 3 The \texttt{leebounds} command

\texttt{leebounds} requires Stata 11 or higher. The prefix command \texttt{bootstrap} is allowed but is not recommended. \texttt{pweight} (default), \texttt{fweight}, and \texttt{iweight} are allowed; see \cite{11.1.6 weight}. Observations with a negative weight are skipped for any type of weight.

#### 3.1 Syntax

\begin{verbatim}
leebounds depvar treatvar [if] [in] [weight] [, select(varname)
   tight(varlist) cieffect vce(analytic|bootstrap) level(#) ]
\end{verbatim}

\texttt{depvar} is a numeric outcome variable, and \texttt{treatvar} is a binary treatment indicator that can be either numeric or a string variable. The (alphanumerically) larger value of \texttt{treatvar} is assumed to indicate treatment.

#### 3.2 Options

\texttt{select(varname)} specifies a binary selection indicator. \texttt{varname} can be either numeric or a string variable. The (alphanumerically) larger value of \texttt{varname} is assumed to indicate selection. If no selection indicator is specified, any observation with nonmissing information on \texttt{depvar} is assumed to be selected, and all observations with missing information on \texttt{depvar} are assumed to be not selected.

\(^4\) This may also indicate a violation of the monotonicity assumption.
t(\texttt{varlist}) specifies a list of covariates for computing tightened bounds. With \texttt{t()} specified, the sample is split into cells defined by the covariates in \texttt{varlist}. Continuous variables in \texttt{varlist} will cause the estimation procedure to fail.

c\texttt{ieffect} requests calculation of a confidence interval for the treatment effect. This interval captures both uncertainty about the selection bias and uncertainty about the sampling error.

\texttt{vce(\texttt{analytic} \mid \texttt{bootstrap})} specifies whether analytic or bootstrapped standard errors are calculated for estimated bounds. \texttt{analytic} is the default. \texttt{bootstrap} allows for the suboptions \texttt{reps(\#)} and \texttt{nodots}. For \texttt{vce(\texttt{analytic})}, the covariance for the estimated lower and upper bound is not computed. If this covariance is relevant, one should choose \texttt{vce(bootstrap)}. Instead of specifying \texttt{vce(bootstrap)}, one can use the prefix command \texttt{bootstrap}, which allows for numerous additional options. Yet \texttt{leebounds}'s internal bootstrapping routine is much faster than the prefix command, allows for sampling weights by performing a weighted bootstrap, and makes the option \texttt{cieffect} use bootstrapped standard errors.

\texttt{level(\#)} sets confidence level. One can change the reported confidence level by retyping \texttt{leebounds} without arguments and specifying only the option \texttt{level(\#)}. This affects the confidence interval for the bounds, but it does not affect the confidence interval requested with \texttt{cieffect}.

### 3.3 Stored results

\texttt{leebounds} stores the following in \texttt{e()}:

**Scalars**

- \texttt{e(N)}: number of observations
- \texttt{e(Nsel)}: number of selected observations
- \texttt{e(clower)}: lower bound of treatment-effect confidence interval (if \texttt{cieffect} was specified)
- \texttt{e(upper)}: upper bound of treatment-effect confidence interval (if \texttt{cieffect} was specified)
- \texttt{e(trim)}: (overall) trimming proportion
- \texttt{e(level)}: confidence level
- \texttt{e(cells)}: number of cells (if option \texttt{tight()} was specified)
- \texttt{e(N_reps)}: number of bootstrap repetitions (if option \texttt{vce(bootstrap)} was specified)

**Macros**

- \texttt{e(cmd)}: \texttt{leebounds}
- \texttt{e(cmdline)}: command as typed
- \texttt{e(title)}: title in estimation output
- \texttt{e(depvar)}: name of dependent variable
- \texttt{e(treatment)}: binary treatment indicator
- \texttt{e(wtype)}: weight type
- \texttt{e(wexp)}: weight expression
- \texttt{e(select)}: varname (if option \texttt{select()} was specified)
- \texttt{e(cellsel)}: cell-specific selection pattern, \texttt{homo} or \texttt{hetero} (if option \texttt{tight()} was specified)
- \texttt{e(covariates)}: \texttt{varlist} (if option \texttt{tight()} was specified)
- \texttt{e(trimmed)}: treatment or control
- \texttt{e(vce)}: \texttt{vcetype} specified in \texttt{vce()}
- \texttt{e(vcetype)}: title used to label Std. Err.
- \texttt{e(properties)}: \texttt{b V}
Lee treatment-effect bounds

Matrices
- \textbf{e(b)}: vector of estimated treatment-effect bounds
- \textbf{e(V)}: variance–covariance matrix of the estimates (covariance set to zero for \texttt{vce(analytic)})

Functions
- \textbf{e(sample)}: marks estimation sample

4 Examples

We use \texttt{cancer.dta}, which is shipped with Stata, for a simple illustrative application; for serious applications of Lee’s bounds estimator besides that in Lee (2009), see Augurzky et al. (2012) or Cawley and Price (2013). We analyze how being treated with an active ingredient (\texttt{drug == 2 | drug == 3}) versus being treated with a placebo (\texttt{drug == 1}) affects survival time (\texttt{studytime}). We treat the data as if information on survival time were available for only those who died during the study (\texttt{died == 1}). This is not entirely correct for those who did not die (\texttt{died == 0}), because we know that they survived at least for the rest of the study period. Yet, in our illustration, we regard them as attriters without any (valid) information on the outcome \texttt{studytime}.

\begin{verbatim}
. sysuse cancer.dta, clear
(Patient Survival in Drug Trial)
. generate activedrug = (drug == 2 | drug == 3)
. leebounds studytime activedrug, select(died)
Lee (2009) treatment effect bounds
Number of obs. = 48
Number of selected obs. = 31
Trimming porportion = 0.5489

|            | Coef. | Std. Err. | z    | P>|z| | [95% Conf. Interval] |
|------------|-------|-----------|------|------|----------------------|
| studytime  |       |           |      |      |                      |
| activedrug | lower | 2.866667  | 3.909154 | 0.73 | 0.463 | -4.795134 10.52847 |
|           | upper | 14.3      | 3.163771 | 4.52 | 0.000 | 8.099123 20.50088 |
\end{verbatim}

The output displays that 48 individuals participated in the trial. Out of these, 31 died during the study and 17 survived. The latter are regarded as not selected because we have no precise information about survival time. The trimming proportion corresponds to \( q \); see (1). The value 0.5489 indicates that the control group is trimmed by more than half, because the survival rate is much higher among individuals who were treated with an active drug. Correspondingly, the estimated treatment-effect bounds are pretty wide, ranging from a 2.87- to a 14.30-month gain in survival time. Taking standard errors into account, the lower bound does not significantly deviate from zero. To obtain a confidence interval for the treatment effect (see section 2.3), one can choose the \texttt{ci effect} option.
Lee (2009) treatment effect bounds

|                | Coef.   | Std. Err. | z  | P>|z|  | [95% Conf. Interval] |
|----------------|---------|-----------|----|------|---------------------|
| studytime      |         |           |    |      |                     |
| activedrug     |         |           |    |      |                     |
| lower          | 2.866667| 3.909154  | 0.73| 0.463| -4.795134 10.52847 |
| upper          | 14.3    | 3.163771  | 4.52| 0.000| 8.099123 20.50088  |

This interval is narrower than the combined confidence intervals for the bounds. One can allow for a less strict level of confidence by specifying `level(90)`. To illustrate the `vce()` option, we opt for bootstrapped rather than analytic standard errors.

```stata
. set seed 13052007
. leebounds studytime activedrug, sel(died) cie level(90) vce(boot, reps(250))
```

Lee (2009) treatment effect bounds

|                | Coef.   | Std. Err. | z  | P>|z|  | [90% Conf. Interval] |
|----------------|---------|-----------|----|------|---------------------|
| studytime      |         |           |    |      |                     |
| activedrug     |         |           |    |      |                     |
| lower          | 2.866667| 3.749864  | 0.76| 0.445| -3.301311 9.034644  |
| upper          | 14.3    | 3.00403   | 4.76| 0.000| 9.358811 19.24119  |

Bootstrapped standard errors are similar to their analytical counterparts. Even the 90% confidence interval for the treatment effect overlaps the value of zero. Finally, we try to tighten the bounds by using a covariate. The only one available is `age`, which we have to transform into a categorical variable. Here we choose three age categories, with each category having roughly the same number of observations.

```stata
. _pctile age, percentiles(33 66 99)
. generate agecat = recode(age, r(r1), r(r2), r(r3))
. leebounds studytime activedrug, select(died) cieffect tight(agecat)
```

Tightened Lee (2009) treatment effect bounds

<table>
<thead>
<tr>
<th></th>
<th>Observed</th>
<th>Bootstrap</th>
<th>Normal-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>studytime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>activedrug</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lower</td>
<td>2.866667</td>
<td>3.749864</td>
<td>-3.301311 9.034644</td>
</tr>
<tr>
<td>upper</td>
<td>14.3</td>
<td>3.00403</td>
<td>9.358811 19.24119</td>
</tr>
</tbody>
</table>
Lee treatment-effect bounds

| studytime | Coef. | Std. Err. | z    | P>|z| | [95% Conf. Interval] |
|-----------|-------|-----------|------|------|----------------------|
| activedrug | lower | 7 | 4.155293 | 1.68 | 0.092 | -1.144225 - 15.14423 |
|           | upper | 12.55556 | 4.29805 | 2.92 | 0.003 | 4.131531 - 20.97968 |

Tightening yields much narrower bounds for the treatment effect. Indeed, with the `tight()` option specified, the 95% treatment-effect confidence interval does not include the value zero.

```
. heckman studytime activedrug i.agecat, select(died = activedrug i.agecat)
```

Iteration 0:  log likelihood = -125.92466
Iteration 1:  log likelihood = -125.57366
Iteration 2:  log likelihood = -125.47902
Iteration 3:  log likelihood = -125.47786
Iteration 4:  log likelihood = -125.47786

Heckman selection model  Number of obs = 48
(regression model with sample selection)  Censored obs = 17
Uncensored obs = 31
Wald chi2(3) = 7.86
Log likelihood = -125.4779  Prob > chi2 = 0.0489

| studytime | Coef. | Std. Err. | z    | P>|z| | [95% Conf. Interval] |
|-----------|-------|-----------|------|------|----------------------|
| studytime | activedrug | 9.84113 | 4.180875 | 2.35 | 0.019 | 1.646767 - 18.03549 |
| agecat | 58 | -3.662228 | 3.57097 | -1.03 | 0.305 | -10.6612 - 3.336745 |
| | 67 | -6.869451 | 3.478465 | -1.97 | 0.048 | -13.68712 - .0517839 |
| _cons | | 12.1558 | 2.635776 | 4.61 | 0.000 | 6.989771 - 17.32182 |
| died | activedrug | -1.945951 | .542586 | -3.59 | 0.000 | -3.00943 - .8825016 |
| agecat | 58 | .9610446 | .5307829 | 1.81 | 0.070 | .0792707 - 2.00136 |
| | 67 | .8677531 | .5754751 | 1.51 | 0.132 | -.2601574 - 1.995664 |
| _cons | | 1.1392 | .5072488 | 2.25 | 0.025 | .1450108 - 2.1339 |
| /athrho | | .0328594 | .5928681 | 0.06 | 0.956 | -1.12914 - 1.19486 |
| /lnsigma | | 1.948617 | .1272405 | 15.31 | 0.000 | 1.69923 - 2.198004 |

LR test of indep. eqns. (rho = 0):  chi2(1) = 0.00  Prob > chi2 = 0.9555

Finally, to compare `leebounds` with fitting a conventional sample selection model, we run Stata's `heckman` command using the same data. Here the variable `agecat` enters both equations of the Heckman model as a control. `heckman` yields a point estimate centered between the lower and the upper bound estimated by `leebounds` with `agecat`
used for tightening. However, the result from `heckman` is imprecise. The estimated confidence interval for the treatment effect is almost as wide as its counterpart from `leebounds`. Hence, in this particular example, the restrictive assumptions inherent to the Heckman selection model do not pay off in terms of substantially reduced uncertainty about the size of the treatment effect. One reason for this may be that the data lack variables that explain selection into the estimation sample while not being directly linked to the outcome variable `studytime`.

5 Conclusion

In this article, I introduced the new command `leebounds`, which implements Lee’s (2009) treatment-effect bounds for data with random assignment of treatment that suffer from nonrandom sample selection. In addition to calculating point estimates for the bounds, the command accommodates the calculation of confidence intervals for the treatment effect and tightened bounds on the basis of covariates. `leebounds` complements the contributions of Beresteanu and Manski (2000) and Palmer et al. (2011), who have made other bounds estimators available to Stata users that, unlike Lee’s estimator, deal with selection into treatment and imperfect compliance with a randomly assigned treatment.

6 Acknowledgments

This work has been supported in part by the Collaborative Research Center “Statistical Modelling of Nonlinear Dynamic Processes” (SFB 823) of the German Research Foundation (DFG). I thank Arndt R. Reichert, Domenico Depalo, and participants of the 2012 German Stata Users Group meeting for many valuable comments.

7 References


Lee treatment-effect bounds


Heckman, J. J. 1976. The common structure of statistical models of truncation, sample selection and limited dependent variables and a simple estimator for such models. *Annals of Economic and Social Measurement* 5: 475–492.


About the author

Harald Tauchmann is a professor of health economics at the Friedrich-Alexander-Universität Erlangen-Nuremberg, a guest researcher at Rheinisch-Westfälisches Institut für Wirtschaftsforschung (RWI), Essen, Germany, and a research fellow at the Centre of Health Economics Research (CINCH), Essen, Germany. His research interests include health economics, applied econometrics, and statistical methods.
General-to-specific modeling in Stata

Damian Clarke
Department of Economics
University of Oxford
Oxford, UK
damian.clarke@economics.ox.ac.uk

Abstract. Empirical researchers are frequently confronted with issues regarding which explanatory variables to include in their models. This article describes the application of a well-known model-selection algorithm to Stata: general-to-specific (GETS) modeling. This process provides a prescriptive and defendable way of selecting a few relevant variables from a large list of potentially important variables when fitting a regression model. Several empirical issues in GETS modeling are then discussed, specifically, how such an algorithm can be applied to estimations based upon cross-sectional, time-series, and panel data. A command is presented, written in Stata and Mata, that implements this algorithm for various data types in a flexible way. This command is based on Stata’s regress or xtreg command, so it is suitable for researchers in the broad range of fields where regression analysis is used. Finally, the genspec command is illustrated using data from applied studies of GETS modeling with Monte Carlo simulation. It is shown to perform as empirically predicted and to have good size and power (or gauge and potency) properties under simulation.

Keywords: st0365, genspec, model selection, general to specific, statistical analysis, specification tests

1 Introduction

A common problem facing the applied statistical researcher is that of restricting her or his models to include the appropriate subset of variables from the real world. This is particularly the case in regression analysis, where the researcher has a determined dependent variable \( y \) but can (theoretically) include any number of explanatory variables \( X \) in the analysis of \( y \). Sometimes, the researcher can invoke a theory that provides guidance about what an appropriate set of \( X \) variables may be. However, at other times, an overarching theory may be absent or may fail to prescribe a parsimonious set of variables. In this situation, the researcher is confronted by issues of model selection: Of all the variables that could be important, which should be included in the final regression model?

Econometric theory expounds on this and can offer useful guidance to all classes of applied statistical researchers—both economists and noneconomists alike. One example of such guidance concerns the general-to-specific or general-to-simple (GETS) modeling procedure. GETS is a prescriptive way to select a parsimonious and instructive final model from a large set of real-world variables and enables the researcher to avoid unnecessary ambiguity or ad hoc decisions. This process involves the definition of a general
model that contains all potentially important variables and then, via a series of step-
wise statistical tests, the removal of empirically “unimportant” variables to arrive at
the proposed specific or final model.

There is a considerable amount of literature on the theoretical merits and draw-
backs of such a process of model selection. Hendry and coauthors (see, for example,
Krolzig and Hendry [2001]; Campos, Ericsson, and Hendry [2005]; Hendry and Krolzig
[2005]; and references therein) have various articles defining aspects of the GETS es-
timation procedure and its properties. Applications of GETS are common in analyses
of economic growth (Hendry and Krolzig 2004), consumption (Hoover and Perez 1999;
Campos and Ericsson 1999), and various phenomena in the noneconomic literature (Su-
carrat and Escribano 2012; Cairns et al. 2011).

GETS modeling is driven by a large group of variables¹ and a series of statistical tests
based on subsets of these models. The outcome of a GETS search process is a specific
model that is consistent with necessary properties for valid inference and that contains
all the statistically significant variables from the initial large set. In this sense, model
selection is based upon the observed data and the results of the tests on these data. Such
“data-driven” model selection is not without its critics. Both philosophical (Kennedy
2002a,b) and statistical (Harrell 2001) critiques have been levied against this approach,
with suggestions that it may result in the underestimation of confidence intervals and
p-values and should entail a penalty in terms of degrees of freedom lost.

Despite these critiques, significant arguments can be, and have been, made in favor
of a GETS modeling process.² Particularly, it appears to perform very well in recovering
the true data-generating process (DGP) in Monte Carlo experiments (Hoover and Perez
1999). For this reason, in this article, I introduce GETS modeling and the corresponding
gen spec statistical routine as an addition to the applied researcher’s toolkit in Stata.
This tool is similar to what already exists in other languages such as R and OxMetrix,
and it is a useful extension to Stata’s functionality. As will be shown, gen spec performs
as empirically expected and does a good job in recovering the true underlying model in
benchmark Monte Carlo simulations.

---

¹ Typically, this consists of all potentially important independent variables that the researcher can
include, along with nonlinearities and lagged dependent and independent variables.
² In the remainder of this article, I (purposely) avoid discussions of the merits and drawbacks of this
routine and instead focus on how researchers can implement such a process if they deem it desirable
and useful in their specific context. A long line of literature including counterarguments to the
above concerns exists (see, for example, Hansen [1996], who provides a balanced introduction), and
the interested reader is directed to these resources.
The `genspec` command, as well as the GETS statistical routine in general, is designed with regression analysis in mind. For this reason, `genspec` is based on Stata’s `regress` (or `xtreg`) command. When moving from a series of potential explanatory variables to one final specific model, `genspec` runs a number of stepwise regressions, with the subsequent testing and removal of insignificant variables. This routine is defined in a flexible way to make it functional in a range of modeling situations. It can be used with cross-sectional, panel, and time-series data, and it works with Stata functions that are appropriate in models of these types. It places no limitations on arbitrary misspecification of models, allowing such features as clustered standard errors, robust standard errors, and bootstrap- and jackknife-based estimation.

To define an algorithm that is appropriate for a range of very different underlying models, a researcher must make several decisions. GETS modeling requires that the preliminary general model be subjected to a range of prespecification tests to ensure that it complies with the modeling assumptions upon which estimation is based. These assumptions, and indeed the resulting tests, vary by the type of regression model in which a researcher is interested. In the following section, I define and discuss the appropriate tests to run in a range of situations, and I discuss how to select between competing final models in different circumstances.

To illustrate the performance of `genspec`, we take a preexisting benchmark in GETS modeling (Hoover and Perez 1999) and show that similar performance can be achieved in Stata. These results suggest that GETS modeling and the user-written `genspec` command may be useful to Stata users interested in defining appropriate, flexible, and data-driven economic models.

2 Algorithm description

As alluded to before, GETS modeling requires an initial group of variables, runs a series of regressions and automated tests, and provides the researcher with a final specific model. This initial group of variables provided by the researcher is referred to as the general unrestricted model (GUM) and should contain all potentially important independent variables. Before beginning analysis, the `genspec` algorithm tests the GUM for validity via a series of statistical tests (described later); if the GUM is valid, a regression is run, with the stepwise removal of the variable with the lowest t statistic. At each step of the process (or “search path”), a prospective final (or terminal) specification is produced with the true terminal specification found when no insignificant variables remain in the current regression model. A comprehensive description of the GETS search process is provided at the end of this section.

The search algorithm undertaken by `genspec` depends upon the model type and GUM specified by the user. Whether the underlying model is based upon cross-sectional, time-series, or panel data determines the set of initial tests (henceforth, “the battery”) and the set of subsequent tests run at each stage of the search path. In what follows, I discuss the general search algorithm followed for every model, delaying discussion of specific tests until the corresponding subsections for cross-sectional, time series, and panel models.
The genspec algorithm in Stata

In defining the search algorithm, we follow the one described in Hoover and Perez (1999) and in appendix A of Hoover and Perez (2004). Hoover and Perez (1999) is considered an important starting point in the description of a computational GETS modeling process (see, for example, Campos, Ericsson, and Hendry [2005]) and a valid description of the nature of GETS modeling. The algorithm implemented in Stata takes the following form:

1. The user specifies her or his proposed GUM and indicates the relevant data to Stata, using if and in qualifiers if necessary.

2. Of the full sample, 90% is retained, while the remaining 10% is set aside for out-of-sample testing. The battery of tests is run on this 90% sample at the nominal size.\(^3\) If one of these tests is failed, it is eliminated from the battery in the following steps of the search path. If more than one of these tests is failed by the GUM, the user is instructed that the GUM is likely a poor representation of the true model and an alternative general model is requested.\(^4\)

3. Each variable in the general model is ranked by the size of its \(t\) statistic, and the algorithm then follows \(m\) (by default, five) search paths. The first search path is initiated by eliminating the variable with the lowest (insignificant) \(t\) statistic from the GUM. The second follows the same process, but rather than eliminating the lowest, it eliminates the second lowest. This process is followed until reaching the \(m\)th search path that eliminates the \(m\)th-lowest variable. For each search path, the current specification then includes all remaining variables, and this specification is estimated by regression.

4. The current specification is then subjected to the full battery of tests, along with an \(F\) test, to determine whether the current specification is a valid restriction of the GUM. If any of these tests fails, the current search path is abandoned, and the algorithm jumps to the subsequent search path.

5. If the current specification passes the above tests, the variables in the current specification are once again ordered by the size of their \(t\) statistics, and the variable with the next-lowest \(t\) statistic is eliminated. This then becomes a potential current specification, which is subjected to the battery of tests. If any of these tests fails, the model reverts to the previous current specification, and the variable with the second-lowest (insignificant) \(t\) statistic is eliminated. Such a process is followed until a variable is successfully eliminated or until all insignificant variables have been attempted. If an insignificant variable is eliminated, stage 5 is restarted with the current specification. This process is followed iteratively until either all insignificant variables have been eliminated or no more variables can be successfully removed.

---

3. In sections 2.1–2.3, I discuss the specific nature of these tests and the determination of the in-sample and out-of-sample observations.
4. As in all terminal decisions, the user can override this decision and continue with her or his proposed GUM if so desired.
6. Once no further variables can be eliminated, a potential terminal specification is reached. This specification is estimated using the full sample of data. If all variables are significant, it is accepted as the terminal specification. If any insignificant variables remain, these are eliminated as a group, and the new terminal specification is subjected to the battery of tests. If it passes these tests, it is the terminal specification; if it does not, the previous terminal specification is accepted.

7. Each of the \( m \) terminal specifications is compared, and if these are different, the final specification is determined using encompassing or an information criterion (see the related discussion in sections 2.1–2.3).

2.1 Cross-sectional models

Cross-sectional models are subjected to an initial battery of five tests: a Doornik–Hansen test for normality of errors, the Breusch and Pagan (1979) test for homoskedasticity of errors,\(^5\) the Ramsey regression equation specification error test for the linearity of coefficients (Ramsey 1969), and an in-sample and out-of-sample stability \( F \) test. These two final tests consist of a comparison of regressions of each subsample with estimation results for the full sample: in the in-sample test, the two subsamples are composed of two halves of the full sample, while in the out-of-sample test, a comparison is made between the 90% and 10% samples. These tests are analogous to Chow (1960) tests.

Information criteria are used to determine the final model based on ordinary least squares with cross-sectional data. For each of the \( m \) potential terminal specifications, a regression is run, and the Bayesian information criterion (\( \text{BIC} \)) is calculated. The terminal specification that has the lowest \( \text{BIC} \) is determined to be the final specification.

2.2 Time-series models

In time-series models, an additional test is included in the battery discussed above: a test is run for autocorrelated conditional heteroskedasticity up to the second order (Engle 1982). To partition the sample into in sample and out of sample, a researcher discards the final 10% of observations to be used in out-of-sample tests. These are (as in all cases) returned to the sample in the calculation of the final model, and a \( \text{BIC} \) is once again used to choose between terminal specifications.

2.3 Panel-data models

Given the nature of panel data, the initial battery of tests here potentially includes two tests omitted in cross-sectional or time-series models. The first of these is a test for serial correlation of the idiosyncratic portion of the error term (discussed by Wooldridge [2010] and implemented for Stata by Drukker [2003]). The second is a Lagrange multiplier

\(^5\) This test is not run if the fitted model is robust to this type of misspecification.
The genspec algorithm in Stata

test for random effects (given that a random-effects model is specified), which tests the validity of said model (Breusch and Pagan 1980). Along with these tests, a Doornik–Hansen-type test for normality of the idiosyncratic portion of the error term and both in-sample and out-of-sample Chow tests (as previously discussed) are estimated.

To determine the final specification from the resulting $m$ potential terminal specifications, the algorithm uses an encompassing procedure. Each variable included in at least one terminal specification is included in the potential terminal model. This model is then tested according to step 6 of the algorithm listed in section 2.

3 The genspec command

3.1 Syntax

The syntax of the genspec command is as follows:

```
genspec depvar indepvars [if] [in] [weight] [, vce(vcetype) xt(re|fe|be) ts nodiagnostic tlimit(#) numsearch(#) nopartition noserial verbose]
```

Here `depvar` refers to the dependent variable in the general model, and `indepvars` refers to the full set of independent variables to be tested for inclusion in the final model.

3.2 Options

- `vce(vcetype)` determines the type of standard error reported in the fitted regression model and allows standard errors that are robust to certain types of misspecification. `vcetype` may be `robust`, `cluster clustvar`, `bootstrap`, or `jackknife`.
- `xt(re|fe|be)` specifies that the model is based on panel data. Users must specify whether they wish to fit a random-effects (re), fixed-effects (fe), or between-effects (be) model. `xtset` must be specified before using this option.
- `ts` specifies that the model is based on time-series data. `tsset` must be specified before using this option, and if `tsset` is specified, time-series operators may be used.
- `nodiagnostic` turns off the initial diagnostic tests for model misspecification. This should be used with caution.
- `tlimit(#)` sets the critical $t$ value above which variables are considered as important in the terminal specification. The default is `tlimit(1.96)`. 
numsearch(#) defines the number of search paths to follow in the model. The default is numsearch(5). If a large dataset is used, fewer search paths may be preferred to reduce computational time.

nopartition uses the full sample of data in all search paths and does not engage in out-of-sample testing.

noserial requests that no serial correlation test be performed if panel data are used. This option should be specified with the xt option only.

verbose requests full program output of each search path explored.

3.3 Stored results
genspec stores the following in e():

Scalars
e(fit) BIC of final specification

Macros
e(genspec) list of variables from the final specification

The full ereturn list, which includes regression results for the terminal specification, is available by typing ereturn list.

4 Performance

4.1 An example with empirical data
To illustrate the performance of genspec, we use empirical data from a well-known applied study of GETS modeling. Hoover and Perez (1999), using data from Lovell (1983), illustrate that GETS modeling can work well in recovering the true DGP in empirical applications, even when prospective variables are multicollinear. We use the Hoover and Perez (1999) dataset in the example below. A brief description of the source and nature of the data is provided in data appendix A.

We use their model 5 to provide an example of the functionality of genspec. As described in table 2, the dependent variable in model 5 is generated according to

\[ y_{5t} = -0.046 \times ggeq_t + 0.11 \times u_t \]

In the following Stata excerpt, we see that after loading the dataset and defining the full set of candidate variables (first and second lags of all independent variables and first to fourth lags of \( y_{5t} \)), the genspec algorithm searches and returns a model with only one independent variable. As desired, this final model is the true DGP, with slight sampling variation in the coefficient on ggeq due to the relatively small sample size.

However, genspec raises one warning: here the GUM does not pass the full battery of defined tests. Specifically, the GUM fails the in-sample Chow test, which suggests that the coefficients estimated over the first half of the series are statistically different.
The genspec algorithm in Stata

from those estimated over the second half. While this may indicate a structural break
signaling that the GUM may not be an appropriate model, genspec respects the GUM
entered by the user and continues to search for (and find) the true model.

. use genspec_data
(Hoover and Perez (1999) data for use in GETS modelling)
. quietly ds y* u* time, not
. local xvars `r(varlist)´
. local lags l.dcoinc l.dg l.ggeq l.ggfeq l.ggfr l.gnpq l.gpiq l.fmrra
> l.fmbase l.fm1dq l.fm2dq l.fsdj l.fyaaac l.lhc l.lhur l.mu l.mo
. genspec y5 `xvars´ `lags´ l.y5 l2.y5 l3.y5 l4.y5, ts
# of observations is > 10% of sample size. Will not run out-of-sample tests.
The in-sample Chow test rejects equality of coefficients
Respecify using nodiagnostic if you wish to continue without specification
This option should be used with caution.
The GUM fails 1 of 4 misspecification tests. Doornik-Hansen test for normality
of errors not rejected. The presence of (1 and 2 order) ARCH components is
rejected. Breusch-Pagan test for homoscedasticity of errors not rejected.

Specific Model:

<table>
<thead>
<tr>
<th>Source</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>Number of obs = 143</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>23.6849853</td>
<td>1</td>
<td>23.6849853</td>
<td>Prob &gt; F = 0.0000</td>
</tr>
<tr>
<td>Residual</td>
<td>1.6984822</td>
<td>141</td>
<td>.012045973</td>
<td>R-squared = 0.9331</td>
</tr>
<tr>
<td>Total</td>
<td>25.3834675</td>
<td>142</td>
<td>.178756814</td>
<td>Adj R-squared = 0.9326</td>
</tr>
</tbody>
</table>

| Coef. | Std. Err. | t | P>|t| | [95% Conf. Interval] |
|-------|-----------|--|---------|------------------|
| ggeq  | -.0463615 | .0010455 | -44.34 | 0.000 | -.0484284 | -.0442945 |
| _cons | -.0042157 | .0091781 | -0.46 | 0.647 | -.0223602 | .0139289 |

4.2 Monte Carlo simulation

The previous example suggests that a GETS algorithm performed well in this particular
case. However, to be confident in the functionality of the genspec command, we are
interested in testing whether this performs as empirically expected over a larger range
of models and circumstances. For this reason, we run a set of Monte Carlo simulations
based upon the empirical data described above and in data appendix A. The reason
we test the performance of genspec on these data is twofold. First, the highly multi-
collinear nature of many of these variables makes recovering the true DGP a challenge
for automated search algorithms. Second, and fundamentally, there is already a bench-
mark performance test of how a GETS algorithm should work on the data available in
Hoover and Perez’s (1999) results.

The Monte Carlo simulation is designed as follows. We draw a normally distributed
random variable for use as the u term described in table 2. Using this draw \( u^D \) (where
superscript \( D \) denotes simulated data), we generate the corresponding \( u^* \) (\( u^*D \)); then,
combining \( u^D, u^*D, \) and the true macroeconomic variables, we simulate each of our nine
different outcome variables \( y_1^D, \ldots, y_9^D \) outlined in the data appendix. Once we have one
simulation for each dependent variable, we run `genspec` with the 40 candidate variables and determine whether the true DGP is recovered. This process makes up one simulation. We then repeat this 1,000 times, observing in each case whether `genspec` identifies the true model and, if not, how many of the true variables are correctly included and how many false variables are incorrectly included.

To determine the performance of the search algorithm, we compare the performance of `genspec` with that of the benchmark performance described in table 7 of Hoover and Perez (1999). We focus on two important summary statistics: gauge and potency. Gauge refers to the percent of irrelevant variables in the final model (regardless of whether they are significant or not). The gauge shows the frequency of type I errors in the search algorithm and is analogous to power in typical statistical tests. The potency of our model refers to the percent of relevant variables in our final model (Castle, Doornik, and Hendry 2012). We would hope in most searches that potency is approximately 100% because the final model should at the very least not discard true variables. We would prefer to have a higher gauge (and more irrelevant—and perhaps insignificant—variables) if this implies that the final model includes all true variables.

Table 1 presents the performance of the `genspec` search algorithm and compares this with the benchmark levels expected. In each case, we see that `genspec` performs approximately identically to Hoover and Perez’s (1999) empirical observations. Fundamentally, the potency of `genspec` is identical to that expected with these data, which suggests that the search algorithm performs as expected in identifying true variables. We do see, however, that `genspec` is more likely to incorrectly include false variables because it has a higher gauge than benchmark performance. This is likely due to a slight difference in the battery of tests in `genspec` compared with that of Hoover and Perez’s (1999) algorithm. In `genspec`, by default, the critical value for the battery of tests is set at 5%: this increases the likelihood that a specific test is retained for the full search path. In the simulations below, Hoover and Perez (1999) report results for a critical value of 1% in the battery of tests, while the `genspec` algorithm reports results at 5% (and 1% for the critical t-value when eliminating irrelevant variables).
Table 1. Performance of `genspec` in Monte Carlo simulation

<table>
<thead>
<tr>
<th>Models</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Panel A: Algorithm performance</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average rate of inclusion of</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True variables</td>
<td>N/A</td>
<td>1.00</td>
<td>1.89</td>
<td>1.00</td>
<td>1.00</td>
<td>1.01</td>
<td>3.00</td>
<td>2.95</td>
<td>2.33</td>
<td>—</td>
</tr>
<tr>
<td>False variables</td>
<td>0.24</td>
<td>1.42</td>
<td>0.64</td>
<td>0.51</td>
<td>0.55</td>
<td>0.62</td>
<td>2.29</td>
<td>0.73</td>
<td>1.39</td>
<td>0.93</td>
</tr>
<tr>
<td>Gauge</td>
<td>0.6%</td>
<td>3.6%</td>
<td>1.6%</td>
<td>1.3%</td>
<td>1.4%</td>
<td>1.5%</td>
<td>5.7%</td>
<td>1.8%</td>
<td>3.5%</td>
<td>2.3%</td>
</tr>
<tr>
<td>Potency</td>
<td>N/A</td>
<td>100.0%</td>
<td>94.5%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>50.1%</td>
<td>100.0%</td>
<td>98.2%</td>
<td>46.6%</td>
<td>87.7%</td>
</tr>
<tr>
<td><strong>Panel B: Benchmark performance</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average rate of inclusion of</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True variables</td>
<td>N/A</td>
<td>1.00</td>
<td>1.89</td>
<td>0.99</td>
<td>1.00</td>
<td>1.01</td>
<td>2.82</td>
<td>3.00</td>
<td>2.86</td>
<td>—</td>
</tr>
<tr>
<td>False variables</td>
<td>0.29</td>
<td>2.31</td>
<td>0.39</td>
<td>0.34</td>
<td>0.32</td>
<td>0.26</td>
<td>1.23</td>
<td>0.38</td>
<td>1.20</td>
<td>0.75</td>
</tr>
<tr>
<td>Gauge</td>
<td>0.7%</td>
<td>5.7%</td>
<td>0.9%</td>
<td>0.8%</td>
<td>0.7%</td>
<td>0.6%</td>
<td>3.0%</td>
<td>0.9%</td>
<td>3.2%</td>
<td>1.8%</td>
</tr>
<tr>
<td>Potency</td>
<td>N/A</td>
<td>100.0%</td>
<td>94.7%</td>
<td>99.9%</td>
<td>100.0%</td>
<td>50.3%</td>
<td>94.0%</td>
<td>99.9%</td>
<td>57.3%</td>
<td>87.0%</td>
</tr>
</tbody>
</table>

Notes: Panel A shows the performance of the user algorithm written for Stata `genspec`, while panel B shows the benchmark algorithm of Hoover and Perez (1999), who simulate using the same data (see their Table 7 for original results). Results from each panel are from 1,000 simulations with a 2-tail critical value of 1%. The DGP for each model is described in the data appendix of this article, and each model includes a constant that is ignored when calculating the gauge and scope. Full code and simulation results for replication are available at https://sites.google.com/site/damianclarke/research.
5 Conclusion

Applied researchers are often faced with determining the appropriate set of independent variables to include in an analysis when examining a given outcome variable. This process of model selection can have important implications on the results of a given research agenda, even when the research question and methodology have been set. General-to-specific modeling offers a researcher a prescriptive, defendable, and data-driven way to resolve this issue. Although this methodology has been drawn from a considerable amount of econometric literature, nothing suggests that it should not be used by all classes of researchers interested in regression analysis.

In this article, I introduce the genspec command to Stata. It shows that this command behaves as empirically expected and is successful in recovering the true model when given a large set of potential variables to choose from. Such a modeling technique offers important benefits to a range of users who are interested in identifying an underlying model while remaining relatively agnostic or placing few restrictions on their general theory.

The genspec command is flexible, allowing the user to choose from a wide array of models using either time-series, panel, or cross-sectional data. I also discuss several empirical considerations in developing such an algorithm, in particular, the nature of the tests desired when examining the proposed general model and how to deal with model selection when choosing between multiple models.

6 Acknowledgments

Financial support from the National Commission for Scientific and Technological Research of the Government of Chile is gratefully acknowledged. I thank Bent Nielsen, Marta Dormal, George Vega Yon, and Nicolas Van de Sijpe for useful comments at various stages in the writing of this command and article. I also acknowledge H. Joseph Newton and an anonymous Stata Journal referee for valuable comments and help. This routine nests the xtserial command, which was written for Stata by David Drukker. All remaining errors and omissions are my own.

7 References


The genspec algorithm in Stata


**About the author**

Damian Clarke is a DPhil. (PhD) student in the Department of Economics at the University of Oxford.

**A Data appendix**

To test the performance of *genspec*, we use the benchmark performance of Hoover and Perez (1999). They use data from the Citibank economic database with 18 macroeconomic variables over the period 1959 quarter 1 to 1995 quarter 1. These variables include gross national product, M1, M2, labor force and unemployment rates, government purchases, and so on. They difference these data to ensure that each series is stationary.

In this article, we work with the same dataset after performing the same transformations. From these 18 underlying macroeconomic variables (and their first lags), Hoover and Perez (1999) generate artificial variables for consumption. Nine such models are generated with two different independent variables and their lags and the lags of the dependent variable. In table 2, we briefly describe these models (as laid out in table 3 of Hoover and Perez [1999]).
Table 2. Models to test the performance of `genspec`

<table>
<thead>
<tr>
<th>Model</th>
<th>DGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>$y_{1t} = 130.0 \times u_t$</td>
</tr>
<tr>
<td>Model 2</td>
<td>$y_{2t} = 130.0 \times u_t^*$</td>
</tr>
<tr>
<td>Model 3</td>
<td>$\ln(y_{3t}) = 0.395 \times \ln(y_{3t-1}) + 0.3995 \times \ln(y_{3t-2}) + 0.00172 \times u_t$</td>
</tr>
<tr>
<td>Model 4</td>
<td>$y_{4t} = 1.33 \times fm1dq_t + 9.73 \times u_t$</td>
</tr>
<tr>
<td>Model 5</td>
<td>$y_{5t} = -0.046 \times ggeq_t + 0.11 \times u_t$</td>
</tr>
<tr>
<td>Model 6</td>
<td>$y_{6t} = 0.67 \times fm1dq_t - 0.023 \times ggeq_t + 4.92 \times u_t$</td>
</tr>
<tr>
<td>Model 7</td>
<td>$y_{7t} = 1.33 \times fm1dq_t + 9.73 \times u_t$</td>
</tr>
<tr>
<td>Model 8</td>
<td>$y_{8t} = -0.046 \times ggeq_t + 0.11u_t^*$</td>
</tr>
<tr>
<td>Model 9</td>
<td>$y_{9t} = 0.67 \times fm1dq_t - 0.023 \times ggeq_t + 4.92u_t$</td>
</tr>
</tbody>
</table>

Notes: The error terms follow $u_t \sim N(0,1)$ and $u_t^* = 0.75u_t^{*1} + u_t\sqrt{7/4}$. Models involving the first-order autoregressive $u_t^*$ can be rearranged to include only $u_t$ and one lag of the dependent variable and any independent variables included in the model. The independent variable $fm1dq_t$ refers to M1 money supply, and $ggeq_t$ refers to government spending.

Each of these nine models results in one artificial consumption variable denominated $y_{nt}$. These $y_{nt}$ variables are then used as the dependent variables for a GETS model search, with 40 independent variables included as candidate variables. These 40 variables are each of the 18 macroeconomic variables in the Citibank economic dataset, the first lags of these variables, and the first to fourth lags of the $y_{nt}$ variable in question. The full-transformed dataset, including a simulated set of $u$ and $y_{nt}$ variables, is available at https://sites.google.com/site/damiancclarke/research.  

6. The untransformed original data are also available.
Robust data-driven inference in the regression-discontinuity design

Sebastian Calonico  
University of Miami  
Coral Gables, FL  
scalonico@bus.miami.edu

Matias D. Cattaneo  
University of Michigan  
Ann Arbor, MI  
cattaneo@umich.edu

Rocio Titiunik  
University of Michigan  
Ann Arbor, MI  
titiunik@umich.edu

Abstract. In this article, we introduce three commands to conduct robust data-driven statistical inference in regression-discontinuity (RD) designs. First, we present rdrobust, a command that implements the robust bias-corrected confidence intervals proposed in Calonico, Cattaneo, and Titiunik (2014d, Econometrica 82: 2295–2326) for average treatment effects at the cutoff in sharp RD, sharp kink RD, fuzzy RD, and fuzzy kink RD designs. This command also implements other conventional nonparametric RD treatment-effect point estimators and confidence intervals. Second, we describe the companion command rdbwselect, which implements several bandwidth selectors proposed in the RD literature. Following the results in Calonico, Cattaneo, and Titiunik (2014a, Working paper, University of Michigan), we also introduce rdplot, a command that implements several data-driven choices of the number of bins in evenly spaced and quantile-spaced partitions that are used to construct the RD plots usually encountered in empirical applications. A companion R package is described in Calonico, Cattaneo, and Titiunik (2014b, Working paper, University of Michigan).

Keywords: st0366, rdrobust, rdbwselect, rdplot, regression discontinuity (RD), sharp RD, sharp kink RD, fuzzy RD, fuzzy kink RD, treatment effects, local polynomials, bias correction, bandwidth selection, RD plots

1 Introduction

The regression-discontinuity (RD) design is a well-established and widely used research design in empirical work. In this design, units receive treatment on the basis of whether their value of an observed covariate is above or below a known cutoff. The key feature of the design is that the probability of receiving treatment conditional on this covariate jumps discontinuously at the cutoff, inducing “variation” in treatment assignment that is assumed to be unrelated to potential confounders. Because of its local nature, RD average treatment-effects estimators are usually constructed using local polynomial nonparametric regression, and statistical inference is based on large-sample approximations (an exception being Cattaneo, Frandsen, and Titiunik [Forthcoming]).

In this article, we discuss data-driven (that is, fully automatic) local-polynomial-based robust inference procedures in the RD design. We introduce three main commands

1. For review and further discussion, see Hahn, Todd, and Van Der Klaauw (2001), Porter (2003), Lee (2008), Imbens and Lemieux (2008), Lee and Lemieux (2010), Dinardo and Lee (2011), and Imbens and Kalyanaraman (2012).
that together offer an array of data-driven nonparametric inference procedures useful for RD empirical applications, including point estimators and confidence intervals (CIs). First, our main command, `rdrobust`, implements the bias-corrected inference procedure proposed by Calonico, Cattaneo, and Titiunik (2014d), which is robust to “large” bandwidth choices. This command also implements other classical inference procedures using local polynomial regression, as suggested in the recent econometrics literature (see references in footnote 1). This implementation offers robust bias-corrected CIs for average treatment effects at the cutoff for sharp RD, sharp kink RD, fuzzy RD, and fuzzy kink RD designs, among other possibilities.

To construct these nonparametric estimators and CIs, all of which are based on estimating a regression function in a neighborhood of the cutoff, we need one or more choices of bandwidth. Our second command, `rdbwselect`, which is called by `rdrobust`, provides different data-driven bandwidth selectors based on the recent work of Imbens and Kalyanaraman (2012) and Calonico, Cattaneo, and Titiunik (2014d). Although this command can be used as a stand-alone bandwidth selector in RD applications, its main purpose is to provide fully data-driven bandwidth choices to be used by `rdrobust`.

Finally, our third command, `rdplot`, implements the results described in Calonico, Cattaneo, and Titiunik (2014a), offering several data-driven choices of the number of bins in evenly spaced and quantile-spaced RD plots. These plots are derived either to approximate the regression function with local sample averages of the outcome variable within bins of the running variable or to depict the overall variability of the data in a disciplined and objective way. These selectors are obtained by approximating the asymptotic integrated variance and bias of certain partitioning-type estimators (see Cattaneo and Farrell [2013] for references) and are implemented nonparametrically using spacings and series estimators. We show how the results implemented in `rdplot` can be used to construct the RD plots commonly found in empirical applications. This command offers a fully automatic way of constructing several useful RD plots that can be used for both presenting the data and falsifying the design.

The rest of this article is organized as follows. In section 2, we review the methods implemented in our commands. In sections 3, 4, and 5, we describe the syntax of `rdrobust`, `rdbwselect`, and `rdplot`, respectively. In section 6, we offer an empirical illustration of our commands by investigating party advantages in U.S. Senate elections using the data and research design in Cattaneo, Frandsen, and Titiunik (Forthcoming). We illustrate several of the main features of our commands, including their compatibility with `outreg2` (Wada 2005). Finally, in section 7, we conclude. A companion R package is described in Calonico, Cattaneo, and Titiunik (2014b).

2 Review of methods

In this section, we review the methods implemented in our commands `rdrobust`, `rdbwselect`, and `rdplot`. To avoid distractions and technicalities, we do not present the regularity conditions and technical discussions underlying these methods herein, but these can be found in the references below. Also, to simplify the discussion, we focus
on the special case of the sharp RD design, where the probability of treatment changes deterministically from zero to one at the cutoff. However, we also cover sharp kink RD, fuzzy RD, and fuzzy kink RD designs. We discuss the implementation for the latter cases briefly in section 2.6 (see also sections 3 and 4 for the corresponding syntax), but we refer the reader to Calonico, Cattaneo, and Titiunik (2014d; 2014c) for further details.

For recent reviews on classical inference approaches in the RD design and comprehensive lists of empirical examples, see Imbens and Lemieux (2008), Lee and Lemieux (2010), Dinardo and Lee (2011), and references therein. Here we focus on approaches using local polynomial nonparametric estimators with data-driven bandwidth selectors and bias-correction techniques, also following the recent results in Imbens and Kalyanaraman (2012) and Calonico, Cattaneo, and Titiunik (2014d) and their supplemental appendix (Calonico, Cattaneo, and Titiunik 2014c).

2.1 Setup and notation

We focus on large-sample inference for the average treatment effect at the cutoff in the sharp RD design. For each unit $i$, the scalar random variable $Y_i$ denotes the outcome of interest, while the scalar regressor $X_i$ is the “running variable” or “score” that determines treatment assignment based on a known cutoff.

We adopt the potential-outcomes framework commonly used in the treatment-effects literature (for example, Heckman and Vytlacil [2007]; Imbens and Wooldridge [2009]). Let $\{(Y_i(0), Y_i(1), X_i') : i = 1, 2, \ldots, n\}$ be a random sample from $\{Y(0), Y(1), X'\}$, where $Y(1)$ and $Y(0)$ denote the potential outcomes with and without treatment, respectively, and treatment assignment is determined by the following known rule: unit $i$ is assigned to the treatment condition if $X_i \geq \tau$ and is assigned to the control condition if $X_i < \tau$ for some known fixed value $\tau$. Thus the observed outcome is

$$Y_i = \begin{cases} Y_i(0) & \text{if } X_i < \tau \\ Y_i(1) & \text{if } X_i \geq \tau \end{cases}$$

and the observed random sample is $\{(Y_i, X_i') : i = 1, 2, \ldots, n\}$.

We discuss and implement several data-driven inference procedures for the sharp average treatment effect at the threshold, which is given by

$$\tau = \mathbb{E}\{Y_i(1) - Y_i(0) | X_i = \tau\}$$

This popular estimand in the RD literature is nonparametrically identifiable under mild continuity conditions (Hahn, Todd, and Van Der Klaauw 2001). Specifically,

$$\tau = \mu_+ - \mu_-$$

with

$$\mu_+ = \lim_{x \uparrow \tau} \mu(x), \quad \mu_- = \lim_{x \downarrow \tau} \mu(x), \quad \mu(x) = \mathbb{E}(Y_i | X_i = x)$$

where here, and elsewhere in the article, we drop the evaluation point of functions whenever possible to simplify notation.
Robust data-driven inference in the regression-discontinuity design

Following Hahn, Todd, and Van Der Klaauw (2001) and Porter (2003), we construct a popular estimator of \( \tau \) by using kernel-based local polynomials on either side of the threshold. These regression estimators are particularly well suited for inference in the RD design because of their good properties at the boundary of the support of the regression function—see Fan and Gijbels (1996) for more details. The local polynomial RD estimator of order \( p \) is

\[
\hat{\tau}_p(h_n) = \hat{\mu}_{+, p}(h_n) - \hat{\mu}_{-, p}(h_n)
\]

where \( \hat{\mu}_{+, p}(h_n) \) and \( \hat{\mu}_{-, p}(h_n) \) denote the intercept (at \( \tau \)) of a weighted \( p \)th-order polynomial regression for only treated and only control units, respectively. More precisely,

\[
\hat{\mu}_{+, p}(h_n) = e'_0 \hat{\beta}_{+, p}(h_n) \quad \text{and} \quad \hat{\mu}_{-, p}(h_n) = e'_0 \hat{\beta}_{-, p}(h_n)
\]

with

\[
\hat{\beta}_{+, p}(h_n) = \arg \min_{\beta \in \mathbb{R}^{p+1}} \frac{1}{n} \sum_{i=1}^{n} I(X_i \geq \tau) \{ Y_i - r_p(X_i - \tau) / \beta \}^2 K_{h_n}(X_i - \tau)
\]

\[
\hat{\beta}_{-, p}(h_n) = \arg \min_{\beta \in \mathbb{R}^{p+1}} \frac{1}{n} \sum_{i=1}^{n} I(X_i < \tau) \{ Y_i - r_p(X_i - \tau) / \beta \}^2 K_{h_n}(X_i - \tau)
\]

where \( r_p(x) = (1, x, \ldots, x^p)' \), \( e_0 = (1, 0, \ldots, 0) \in \mathbb{R}^{p+1} \) is the first unit vector, \( K_{h}(u) = K(u/h) / h \) with \( K() \) a kernel function, \( h_n \) is a positive bandwidth sequence, and \( 1() \) denotes the indicator function.

Under simple regularity conditions, and assuming the bandwidth \( h_n \) vanishes at an appropriate rate, local polynomial estimators are known to satisfy

\[
\hat{\beta}_{+, p}(h_n) \rightarrow_p \beta_{+, p} \quad \text{and} \quad \hat{\beta}_{-, p}(h_n) \rightarrow_p \beta_{-, p}
\]

with

\[
\beta_{+, p} = \left( \mu_{+, (1)} \frac{\mu_{+, (2)}}{2}, \ldots, \frac{\mu_{+, (p)}}{p!} \right)'
\]

\[
\beta_{-, p} = \left( \mu_{-, (1)} \frac{\mu_{-, (2)}}{2}, \ldots, \frac{\mu_{-, (p)}}{p!} \right)'
\]

\[
\mu_{+, (s)}(x) = \lim_{x \uparrow \tau} \frac{\partial^s}{\partial x^s} \mu_{+, p}(x)
\]

\[
\mu_{+, p}(x) = \mathbb{E}\{ Y(1) | X_i = x \}
\]

\[
\mu_{-, (s)}(x) = \lim_{x \downarrow \tau} \frac{\partial^s}{\partial x^s} \mu_{-, p}(x)
\]

\[
\mu_{-, p}(x) = \mathbb{E}\{ Y(0) | X_i = x \}
\]

\( s = 1, 2, \ldots, p \), thereby offering a family of consistent estimators of \( \tau \). Among these possible estimators, the local linear RD estimator \( \hat{\tau}_1(h_n) \) is perhaps the preferred and most common choice in practice.
2.2 Overview of upcoming discussion

We now survey the results presented in the following sections to help the reader easily identify the conceptual differences between the estimators implemented by rdrobust. In particular, in sections 2.3 and 2.4, we review some of the salient asymptotic properties of RD treatment-effect estimators \( \hat{\tau}_p(h_n) \), which are based on local polynomial nonparametric estimators. We outline our discussion in these sections as follows:

1. We assess some of the main properties of \( \hat{\tau}_p(h_n) \) as point estimators in the first part of section 2.3. Specifically, we discuss a (conditional) mean-squared error (MSE) expansion of \( \hat{\tau}_p(h_n) \) that highlights its variance and bias properties. We also use this expansion to summarize some bandwidth-selection approaches tailored to minimize the leading terms in the asymptotic MSE expansion, including plug-in rules and a cross-validation (CV) approach.

2. In the rest of section 2.3, we discuss the construction of asymptotically valid CIs based on \( \hat{\tau}_p(h_n) \) for the sharp mean treatment effect \( \tau \). In particular, we discuss two distinct approaches: one based on “undersmoothing” and the other based on “bias correction”.

The first approach, which is arguably the most commonly used in practice, assumes away the bias of the estimator and constructs 100(1 - \( \alpha \)) percent CIs of the form

\[
CI_{1-\alpha,n} = \left\{ \hat{\tau}_p(h_n) \pm \Phi_{1-\frac{\alpha}{2}} \sqrt{\hat{\nu}_n} \right\}
\]

where \( \Phi_{1-\frac{\alpha}{2}} \) denotes the appropriate quantile of the Gaussian distribution (for example, 1.96 for \( \alpha = 0.975 \)), and \( \hat{\nu}_n \) denotes an appropriate choice of variance estimator. This approach is theoretically justified only if the (smoothing) leading bias of the RD estimator is “small”, which requires some form of “undersmoothing”; that is, it requires choosing a “smaller” bandwidth than the MSE-optimal one. In practice, researchers typically use the same bandwidth used to construct the RD point estimator \( \hat{\tau}_p(h_n) \), thereby ignoring the potential effects of the leading bias on the performance of these CIs.

A second approach to constructing CIs is to use bias correction. This approach is conventional in the nonparametric literature, although it is not commonly used in empirical work, because it is regarded as having inferior finite-sample properties. The resulting CIs in this case take the form

\[
CI_{1-\alpha,n}^{bc} = \left[ \left\{ \hat{\tau}_p(h_n) - \hat{\delta}_n \right\} \pm \Phi_{1-\frac{\alpha}{2}} \sqrt{\hat{\nu}_n} \right]
\]

where here the only difference is the bias-estimate \( \hat{\delta}_n \), which is introduced with the explicit goal of removing the potentially large effects of the unknown leading bias of the RD estimator, \( \hat{\tau}_p(h_n) \). This second approach to constructing CIs justifies using MSE-optimal bandwidth choices when constructing the estimators.
3. The results in Calonico, Cattaneo, and Titunik (2014d) offer alternative CIs based on bias-corrected local polynomials, which take the form

\[ \text{CI}_{1-\alpha,n}^{\text{rbc}} = \left\{ \hat{\tau}_p(h_n) - \hat{b}_n \right\} \pm \Phi^{-1}_{1-\alpha} \sqrt{\hat{\sigma}_{\text{rbc}}^2} \]

where the key difference between the “conventional” bias-corrected (CI_{bc}) and these alternative robust CIs (CI_{rbc}) is the presence of a different variance estimator, denoted here by \( \hat{\sigma}_{\text{rbc}}^2 \). This new variance formula is theoretically derived by using an alternative asymptotic approximation to the bias-corrected RD estimator \( \hat{\tau}_p(h_n) - \hat{b}_n \). The resulting CIs have some attractive theoretical properties and, as we discuss in more detail below, allow for the use of MSE-optimal bandwidth choices while offering excellent finite-sample performance. In section 2.4, we offer a heuristic discussion of these results, which are developed by Calonico, Cattaneo, and Titunik (2014d) and are the main motivation for the development of our package.

4. In section 2.5, we summarize all the statistical procedures available in our commands for sharp RD designs. In section 2.6, we briefly explain how these results are extended to sharp kink, fuzzy, and fuzzy kink RD designs, among other possibilities.

5. Finally, in section 2.7, we discuss several fully automatic approaches for constructing the plots commonly shown in RD applications. Specifically, we implement the results in Calonico, Cattaneo, and Titunik (2014a), which offer optimal choices of bin length for evenly spaced and quantile-spaced partitioning schemes for constructing local sample means for control and treatment units. These local sample means help to approximate the underlying regression functions and are usually plotted together with global polynomial estimates to summarize the empirical features of the RD design.

### 2.3 Conventional RD inference

#### Point estimators

Under appropriate regularity conditions, the treatment-effect estimator \( \hat{\tau}_p(h_n) \) admits the following MSE expansion. Let \( X_n = (X_1, X_2, \ldots, X_n)' \).

\[ \text{MSE}_p(h_n) = \mathbb{E} \left[ \left( \hat{\tau}_p(h_n) - \tau \right)^2 \right| X_n \approx h_n^{2(p+1)} B_{n,p}^2 + \frac{1}{nh_n} V_{n,p} \]

with

\[ B_{n,p} \to_p B_p \quad \text{and} \quad V_{n,p} \to_p V_p \]

where \( B_{n,p} \) and \( V_{n,p} \) represent, respectively, the leading asymptotic bias and the asymptotic variance of \( \hat{\tau}_p(h_n) \). The exact form of \( B_{n,p} \) and \( V_{n,p} \), and their asymptotic counterparts, can be found in Calonico, Cattaneo, and Titunik (2014d). This treatment-effect
estimator will be consistent if \( h_n \to 0 \) and \( nh_n \to \infty \). Moreover, the point estimator \( \hat{\tau}(h_n) \) will be optimal in an asymptotic MSE sense if the bandwidth \( h_n \) is chosen so that

\[
h_{\text{MSE},n,p} = \left\{ \frac{V_p}{2(p+1)B^2_p} \right\}^{\frac{1}{(p+3)}} n^{\frac{-1}{(p+3)}}
\]

whenever \( B_p \neq 0 \). This last assumption may be restrictive because \( B_p \mu_+^{(p+1)} - \mu_-^{(p+1)} \) may be (close to) 0 in some applications (see Imbens and Kalyanaraman [2012]; see them also for a recent review on bandwidth selection in the RD design).

Imbens and Kalyanaraman (2012) use this reasoning in providing a data-driven, asymptotically MSE-optimal, RD treatment-effect estimator. Specifically, they propose a more “robust” consistent bandwidth estimator of the form

\[
\hat{h}_{\text{IK},n,p} = \left\{ \frac{\hat{V}_{\text{IK},p}}{2(p+1)B^2_{\text{IK},p} + \hat{R}_{\text{IK},p}} \right\}^{\frac{1}{(p+3)}} n^{\frac{-1}{(p+3)}}
\]

where the additional (regularization) term \( \hat{R}_{\text{IK},p} \) is introduced to avoid small denominators in moderate-size samples. Here \( \hat{B}_{\text{IK},p} \) and \( \hat{V}_{\text{IK},p} \) (and \( \hat{R}_{\text{IK},p} \)) are nonparametric consistent estimators of their respective population counterparts, which require the choice of preliminary bandwidths, generically denoted by \( b_n \) herein. Imbens and Kalyanaraman (2012) provide a direct implementation approach for \( p = 1 \), but the preliminary bandwidths used in their construction are not optimally chosen. Thus \( \hat{h}_{\text{IK},n,p} \) can be viewed as a nonparametric first-generation plug-in rule (for example, Wand and Jones [1995]), sometimes denoted by a direct plug-in rule of order 1.

Motivated by the work of Imbens and Kalyanaraman (2012), Calonico, Cattaneo, and Titiunik (2014d) propose an alternative second-generation plug-in bandwidth selection approach. Specifically, they propose the following second-order direct plug-in rule:

\[
\hat{h}_{\text{CCT},n,p} = \left\{ \frac{\hat{V}_{\text{CCT},p}}{2(p+1)B^2_{\text{CCT},p} + \hat{R}_{\text{CCT},p}} \right\}^{\frac{1}{(p+3)}} n^{\frac{-1}{(p+3)}}
\]

This alternative bandwidth estimator has two distinct features relative to \( \hat{h}_{\text{IK},n,p} \). First, not only are the estimators \( \hat{V}_{\text{CCT},p} \) and \( \hat{B}_{\text{CCT},p} \) (and \( \hat{R}_{\text{CCT},p} \)) consistent for their population counterparts, but the preliminary bandwidths used in their constructions are consistent estimators of the corresponding population MSE-optimal bandwidths. In this sense, \( \hat{h}_{\text{CCT},n,p} \) is a direct plug-in rule of order 2. Second—motivated by finite-sample performance considerations—Calonico, Cattaneo, and Titiunik (2014d) construct an alternative estimator of \( V_p \) (denoted by \( \hat{V}_{\text{CCT},p} \) above) that does not require an additional choice of bandwidth for its construction but, as in the work of Abadie and Imbens (2006), relies instead on a fixed-matches nearest-neighbor-based “estimate” of the residuals. This construction, as well as other more traditional approaches, is discussed further below because the term \( V_p \) plays a crucial role when forming CIs for \( \tau \).
Robust data-driven inference in the regression-discontinuity design

The main bandwidth, \( h_{n,p} \), can be chosen in other ways. A popular alternative is to use cross-validation, as done by Ludwig and Miller (2007). As discussed in Imbens and Kalyanaraman (2012), one such bandwidth-selection approach can be described as follows,

\[
\hat{h}_{CV,n,p} = \arg \min_{h>0} CV_\delta(h), \quad CV_\delta(h) = \sum_{i=1}^{n} \mathbb{I}(X_{-}\{\delta\} \leq X_i \leq X_{+}\{\delta\}) \{Y_i - \hat{\mu}_p(X_i; h)\}^2
\]

where

\[
\hat{\mu}_p(x; h) = \begin{cases} 
\epsilon_0^{-1} \hat{\beta}_{+,p}(x, h) & \text{if } x \geq \tau \\
\epsilon_0^{-1} \hat{\beta}_{-,p}(x, h) & \text{if } x < \tau 
\end{cases}
\]

and

\[
\hat{\beta}_{+,p}(x, h) = \arg \min_{\beta \in \mathbb{R}^{r+1}} \sum_{i=1}^{n} \mathbb{I}(X_i \geq x)(Y_i - r_p(X_i - x)'\beta)^2 K_h(X_i - x)
\]

\[
\hat{\beta}_{-,p}(x, h) = \arg \min_{\beta \in \mathbb{R}^{r+1}} \sum_{i=1}^{n} \mathbb{I}(X_i < x)(Y_i - r_p(X_i - x)'\beta)^2 K_h(X_i - x)
\]

and, for \( \delta \in (0, 1) \), \( X_{-}\{\delta\} \) and \( X_{+}\{\delta\} \) denote the \( \delta \)th quantile of \( \{X_i : X_i < \tau\} \) and \( \{X_i : X_i \geq \tau\} \), respectively. Our bandwidth-selection command also implements this approach for completeness.

To summarize, the results discussed so far justify the following three data-driven RD treatment-effect point estimators:

\[
\hat{\tau}_p(\hat{h}_{IK,n,p}), \quad \hat{\tau}_p(\hat{h}_{CCT,n,p}), \quad \text{and} \quad \hat{\tau}_p(\hat{h}_{CV,n,p})
\]

Under appropriate conditions, these estimators may be interpreted as consistent and (asymptotically) MSE-optimal point estimators of \( \tau \).

**CIs: Asymptotic distribution**

Under appropriate regularity conditions and rate restrictions on the bandwidth sequence \( h_n \to 0 \), conventional CIs accompanying the point estimators discussed above rely on the following distributional approximation,

\[
\sqrt{n}h_n \{\hat{\tau}_p(h_n) - \tau - h_n^{p+1} B_{n,p}\} \to_d N(0, V_p), \quad V_p \frac{\sigma_+^2 + \sigma_-^2}{f}
\]

where \( \sigma_+^2 = \lim_{x \to \tau^+} \sigma^2(x) \) and \( \sigma_-^2 = \lim_{x \to \tau^-} \sigma^2(x) \) with \( \sigma^2(x) = \text{V}(Y_i|X_i = x) \), and \( f = f(\tau) \) with \( f(x) \) the density of \( X \). Therefore, an infeasible asymptotic 100(1 - \( \alpha \))-percent CI for \( \tau \) is given by

\[
\text{CI}_{1-\alpha}(h_n) = \left\{\hat{\tau}_p(h_n) - h_n^{p+1} B_{n,p}\right\} \pm \Phi^{-1}_{1-\alpha/2} \sqrt{\frac{V_p}{nh_n}}
\]

To implement this CI, we need to handle the leading bias \( (B_{n,p}) \) and the variance \( (V_p) \) of the RD estimator because they involve unknown quantities. We discuss these problems and related practical issues in the following subsections.
CIs: Variance estimators

The asymptotic variance is handled by replacing $V_p$ with a consistent estimator. A natural approach uses the conditional (on $X_n$) variance of $\hat{\tau}_p(h_n)$ as a starting point because, as mentioned above, $V_{n,p} \rightarrow_p V_p$. Here we have

$$V_{n,p} = n h_n V \{ \hat{\tau}_p(h_n) | X_n \} = V_{+,n,p} + V_{-,n,p}$$

with

$$V_{+,n,p} = h_n e_0^T \Gamma_{+,n,p}^{-1} X_{+,n,p} W_{+,n,p} \Sigma W_{+,n,p} X_{+,n,p} \Gamma_{+,n,p}^{-1} e_0$$

$$V_{-,n,p} = h_n e_0^T \Gamma_{-,n,p}^{-1} X_{-,n,p} W_{-,n,p} \Sigma W_{-,n,p} X_{-,n,p} \Gamma_{-,n,p}^{-1} e_0$$

where the exact form of these matrices is discussed in Calonico, Cattaneo, and Titiunik (2014d). Importantly, the only matrix including unknown quantities is

$$\Sigma = \begin{bmatrix}
\sigma^2(X_1) & 0 & \cdots & 0 \\
0 & \sigma^2(X_2) & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \sigma^2(X_n)
\end{bmatrix} = \mathbb{E}(\varepsilon \varepsilon^T | X_n)$$

where $\varepsilon = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)'$ with $\varepsilon_i = Y_i - \mu(X_i)$.

The “sandwich” structure of $V_{n,p}$ arises naturally from the weighted least-squares form of local polynomials, resembling the usual heteroskedasticity-robust standard-error formula in linear regression models. Therefore, and just as in linear models, implementing these standard-error estimators requires only an estimator of $\Sigma$, which reduces the problem to plugging in an estimator of $\sigma^2(X_i) = \mathbb{E}(\varepsilon_i^2 | X_i)$ for control and treatment units separately. We consider two approaches to construct such estimators: “plug-in estimated residuals” and “fixed-matches estimated residuals”. Both approaches construct an estimator of $V_{n,p}$ by removing the conditional expectation in $\Sigma$ and replacing $\varepsilon_i$ by some estimator of it.

**Plug-in Estimated Residuals.** This approach follows the standard linear models logic underlying local polynomial estimators and thus replaces $\varepsilon_i$ by

$$\tilde{\varepsilon}_{+,i} = Y_i - \tilde{\mu}_{+,p}(X_i; c_n) \quad \text{and} \quad \tilde{\varepsilon}_{-,i} = Y_i - \tilde{\mu}_{-,p}(X_i; c_n)$$

for treated and control units, respectively, where $\tilde{\mu}_{+,p}(X_i; c_n) = e_0^T \tilde{\beta}_{+,p}(x, c_n)$ and $\tilde{\mu}_{-,p}(X_i; c_n) = e_0^T \tilde{\beta}_{-,p}(x, c_n)$, and $c_n$ denotes the bandwidth used. In practice, $c_n$ is usually chosen to be $h_n$, even though this choice may not be optimal and could lead to poor finite-sample performance of the estimators. The resulting variance formula becomes the familiar Huber–Eicker–White standard-error estimator, which is robust to heteroskedasticity of unknown form. We denote this estimator by

$$\tilde{V}_{n,p} = \tilde{V}_{+,n,p} + \tilde{V}_{-,n,p}$$

where $\tilde{V}_{+,n,p}$ and $\tilde{V}_{-,n,p}$ use, respectively, $\tilde{\varepsilon}_{+,i}$ and $\tilde{\varepsilon}_{-,i}$ in their construction.
Robust data-driven inference in the regression-discontinuity design

Fixed-matches Estimated Residuals. The previous construction is intuitive and easy to implement, but it requires the additional choice of the bandwidth $c_n$ to construct the estimates of the residuals. Employing the same bandwidth choice used to construct the RD treatment-effect estimator may not lead to a variance estimator with good finite-sample properties. Calonico, Cattaneo, and Titiunik (2014d) propose a variance estimator using a different construction for the residuals as an alternative motivated by the recent work of Abadie and Imbens (2006). This estimator is constructed using a simple fixed-matches estimator for the residuals, denoted by $\hat{\varepsilon}_{+,i}$ and $\hat{\varepsilon}_{-,i}$ for treatment and control, respectively, which are unbiased but inconsistent. Nonetheless, the resulting variance estimators are consistent under appropriate regularity conditions (see Calonico, Cattaneo, and Titiunik [2014d] for more details). We denote this estimator by

$$\hat{V}_{n,p} = \hat{V}_{+,n,p} + \hat{V}_{-,n,p}$$

where $\hat{V}_{+,n,p}$ and $\hat{V}_{-,n,p}$ use, respectively, the fixed-matches estimators $\hat{\varepsilon}_{+,i}$ and $\hat{\varepsilon}_{-,i}$ in their construction.

In summary, two alternative variance estimators are i) the plug-in estimated residuals estimator $\hat{V}_{n,p}$ and ii) the fixed-matches estimator $\hat{V}_{n,p}$, both satisfying

$$\hat{V}_{n,p} \Rightarrow_{p} V_p \quad \text{and} \quad \hat{V}_{n,p} \Rightarrow_{p} V_p$$

For implementation, we use the same estimated bandwidth used in the treatment-effect estimator $\hat{\tau}_p(h_n)$ whenever needed.

CIs: Asymptotic bias

There are two main approaches to handle the leading bias ($B_{n,p}$) in the infeasible CIs, $C^{\pm}_{1-\alpha}(h_n)$: “undersmoothing” (alternatively, assuming the bias is “small”) or “bias correction”. We briefly discuss each of these ideas below.

Undersmoothing. The first approach is to “undersmooth” the estimator, that is, to choose a “small” enough bandwidth so that the bias is negligible. Theoretically, this approach simply requires selecting a bandwidth sequence $h_n \to 0$ such that

$$\sqrt{nh_n} \{\hat{\tau}_p(h_n) - \tau - h_n^{p+1} B_{n,p}\} = \sqrt{nh_n} \{\hat{\tau}_p(h_n) - \tau\} + o_p(1) \to_d N(0, V_p)$$

which is justified whenever the bandwidth $h_n$ vanishes fast enough. In practice, however, this procedure may be difficult to implement because most bandwidth selectors—such as $h_{MSE,n,p}$—will not satisfy the conditions required for undersmoothing. This fact implies that most empirical bandwidth selectors could lead to a nonnegligible leading bias in the distributional approximation, which will bias the associated CIs. Simulation evidence highlighting this potential drawback of the undersmoothing (small-bias) approach is provided in Calonico, Cattaneo, and Titiunik (2014d).

Nonetheless, in applications, it is common for researchers to simply ignore the leading bias and proceed as if $B_{n,p} \approx 0$. This approach is justified by either assuming the bias is “small” or by shrinking the bandwidth choice by some ad hoc factor (that
is, undersmoothing). We give the exact formula of the resulting CIs justified by this approach in the next subsection.

**Bias Correction.** The second approach to deal with the leading bias in the distributional approximation is to bias correct the estimator—that is, to construct an estimator of \( B_{n,p} \), which is then subtracted from the point estimate to eliminate this leading bias.

A simple way to implement this idea is to use a higher-order local polynomial to estimate the unknown derivatives in the leading bias term. Recall that \( B_p \equiv \mu_{p+1} - \mu_p \). For example, \( \mu_{p+1} \) can be estimated by using a \( q \)-th-order local polynomial \((q \geq p + 1)\) with pilot bandwidth \( b_n \), leading to the estimator \( \hat{\mu}_{p+1} = e_{p+1}^T \hat{\beta}_{p,q}(b_n) \), where \( e_j \) is the conformable \((j+1)\)th-unit vector—for example, \( e_1 = (0, 1, 0, 0)' \) if \( q = 3 \).

This is the approach we use herein to construct a bias estimate \( \hat{B}_{n,p,q} \), which depends on a preliminary bandwidth choice \( b_n \). The resulting bias-corrected estimator is

\[
\hat{\tau}_{p,q}(h_n, b_n) = \hat{\tau}_p(h_n) - h_n^{p+1} \hat{B}_{n,p,q}, \quad \hat{B}_{n,p,q} = \hat{B}_{n,p,q}(h_n, b_n), \quad p < q
\]

where the exact formula of \( \hat{B}_{n,p,q}(h_n, b_n) \) is given in Calonico, Cattaneo, and Titiunik (2014d).

Using this bias-corrected estimator and imposing appropriate regularity conditions and bandwidth restrictions, we obtain

\[
\sqrt{n h_n} \left\{ \hat{\tau}_{p,q}(h_n, b_n) - \tau \right\} = \sqrt{n h_n} \left\{ \hat{\tau}_p(h_n) - \tau - h_n^{p+1} B_{n,p} \right\} \\
\xrightarrow{d} N(0, V_p) \\
- \sqrt{n h_n} h_n^{p+1} \left( \hat{B}_{n,p,q} - B_{n,p} \right) \\
\xrightarrow{p} 0
\]

which is valid if

\[
\frac{h_n}{b_n} \to 0
\]

This result immediately justifies bias-corrected CIs, where the unknown bias in \( \hat{\tau}_p(h_n) \) is replaced by the bias estimate \( \hat{B}_{n,p,q} \). We give the exact formula of the resulting CIs in the upcoming subsection.

To implement this approach, we must make a choice of pilot bandwidth \( b_n \). As discussed above, bandwidth choices can be constructed using asymptotic MSE expansions for the appropriate estimators. This is the approach followed by Calonico, Cattaneo, and Titiunik (2014d), who propose the following MSE-optimal choice of pilot bandwidth \( b_n \) for the bias-correction estimator \( \hat{B}_{n,p,q} \). This optimal pilot bandwidth is given by

\[
b_{MSE,n,p,q} = \left\{ \frac{(2p + 3) V_{p+1,q}}{2(q-p)B_{p+1,q}^2} \right\}^{\frac{-1}{q-p+1}} n^{-\frac{1}{2(q-p+1)}}
\]

where \( V_{p,q} \) and \( B_{p,q} \) are the corresponding leading variance and bias terms arising from the MSE expansion used. [Note that this choice is not necessarily optimal for
Robust data-driven inference in the regression-discontinuity design

Calonico, Cattaneo, and Titiunik (2014d) discuss a relatively simple implementation procedure of $b_{\text{MSE},n,p,q}$, which leads to the following data-driven estimator:

$$
\hat{b}_{\text{CCT},n,p,q} = \left\{ \frac{(2p + 3)\hat{V}_{\text{CCT},p+1,q}}{2(q - p)\hat{B}_{\text{CCT},p+1,q}^2 + \hat{R}_{\text{CCT},p+1,q}} \right\}^{\frac{1}{2}}
$$

They also provide the exact form of the estimators $\hat{V}_{\text{CCT},p,q}$, $\hat{B}_{\text{CCT},p,q}$, and $\hat{R}_{\text{CCT},p,q}$. We use this pilot bandwidth estimator in our default implementation, but we also implement similar estimators constructed following the underlying logic in Imbens and Kalyanaraman (2012), denoted by $\hat{b}_{\text{IK},n,p,q}$.

CIs: Summary of classical approaches

The results discussed so far suggest the following data-driven RD treatment-effect CIs:

- **Undersmoothing (Small Bias):**
  - Plug-in estimated errors: $\hat{C}_{1-\alpha}(\hat{h}_{\text{IK},n,p})$, $\hat{C}_{1-\alpha}(\hat{h}_{\text{CCT},n,p})$, and $\hat{C}_{1-\alpha}(\hat{h}_{\text{CV},n,p})$, where
    $$
    \hat{C}_{1-\alpha}(h_n) = \left\{ \hat{\tau}_p(h_n) \pm \Phi^{-1}_{1-\frac{\alpha}{2}} \sqrt{\frac{\hat{V}_p}{nh_n}} \right\}
    $$
  - Fixed-matches estimated errors: $\hat{C}_{1-\alpha}(\hat{h}_{\text{IK},n,p})$, $\hat{C}_{1-\alpha}(\hat{h}_{\text{CCT},n,p})$, and $\hat{C}_{1-\alpha}(\hat{h}_{\text{CV},n,p})$, where
    $$
    \hat{C}_{1-\alpha}(h_n) = \left\{ \hat{\tau}_p(h_n) \pm \Phi^{-1}_{1-\frac{\alpha}{2}} \sqrt{\frac{\hat{V}_p}{nh_n}} \right\}
    $$

- **Bias Correction:**
  - Plug-in estimated errors: $\hat{C}_{1-\alpha}^{\text{bc}}(\hat{h}_n, \hat{\beta}_n)$, where
    $$
    \hat{C}_{1-\alpha}^{\text{bc}}(h_n, b_n) = \left\{ \hat{\tau}_p(h_n) - \hat{h}_n^{p+1} \hat{B}_{n,p,q} \right\} \pm \Phi^{-1}_{1-\frac{\alpha}{2}} \sqrt{\frac{\hat{V}_p}{nh_n}}
    $$
  - Fixed-matches estimated errors: $\hat{C}_{1-\alpha}^{\text{bc}}(\hat{h}_n, \hat{\beta}_n)$, where
    $$
    \hat{C}_{1-\alpha}^{\text{bc}}(h_n, b_n) = \left\{ \hat{\tau}_p(h_n) - \hat{h}_n^{p+1} \hat{B}_{n,p,q} \right\} \pm \Phi^{-1}_{1-\frac{\alpha}{2}} \sqrt{\frac{\hat{V}_p}{nh_n}}
    $$

Here $\hat{h}_n \in (\hat{h}_{\text{IK},n,p}, \hat{h}_{\text{CCT},n,p}, \hat{h}_{\text{CV},n,p})$ and $\hat{\beta}_n \in (\hat{\beta}_{\text{IK},n,p+1,q}, \hat{\beta}_{\text{CCT},n,p+1,q})$, for example.
2.4 Robust RD inference

The classical CIs discussed above may have some unappealing properties that could seriously affect their performance in empirical work. The CIs $\hat{C}_{1-\alpha}(h_n)$ and $\hat{C}_{1-\alpha}(h_n)$ require undersmoothing (or small bias), which leads to potentially large-coverage distortions otherwise. The bias-corrected CIs $\hat{C}_{1-\alpha}(h_n, b_n)$ and $\hat{C}_{1-\alpha}(h_n, b_n)$, while theoretically justified for a larger range of bandwidths, are usually regarded as performing poorly in empirical settings, which also leads to potentially large-coverage distortions in applications. Monte Carlo evidence showing some of these potential pitfalls is reported in Calonico, Cattaneo, and Titiunik (2014d).

Calonico, Cattaneo, and Titiunik (2014d) propose alternative, more robust CIs constructed using bias-corrected RD treatment-effect estimators as a starting point. Intuitively, these estimators do not perform well in finite samples because the bias estimate introduces additional variability in $\hat{\tau}_{p,q}^{bc}(h_n, b_n)$. This variability is not accounted for when forming the associated CIs, for example, $\hat{C}_{1-\alpha}(h_n, b_n)$ and $\hat{C}_{1-\alpha}(h_n, b_n)$.

Following this reasoning, Calonico, Cattaneo, and Titiunik (2014d) propose an alternative asymptotic approximation for $\hat{\tau}_{p,q}^{bc}(h_n, b_n)$ that is heuristically given by the observation that

$$\sqrt{nh_n} \{\hat{\tau}_{p,q}^{bc}(h_n, b_n) - \tau\} \rightarrow_d N(0, V_{p,q})$$

where $V_{p,q}(\rho)$ is the asymptotic variance for the bias-corrected estimator, which is different from the usual one, $V_p$. Indeed, it can be shown that $V_{p,q}^{bc}(\rho) \rightarrow V_p$ if $\rho \downarrow 0$; but, in general, $V_{p,q}^{bc}(\rho) > V_p$ under standard conditions.

It can also be shown that, under appropriate conditions,

$$\frac{\hat{\tau}_{p,q}^{bc}(h_n, b_n) - \tau}{\sqrt{V_{n,p,q}^{bc}}} \rightarrow_d N(0, 1), \quad V_{n,p,q}^{bc} = V_{n,p,q}^{bc}(h_n, b_n)$$
where the exact formula for $V_{n,p,q}^{bc}(h_n, b_n)$ is given in Calonico, Cattaneo, and Titiunik (2014d). Intuitively, this variance formula is constructed to account for the variability of both the original RD treatment-effect estimator $\hat{\tau}_p(h_n)$ and the bias-correction term, $\hat{B}_{n,p,q}$, in the distributional approximation of the Studentized statistic. Further theoretical implications of this alternative approach to nonparametric bias correction are discussed in Calonico, Cattaneo, and Farrell (2014).

This more general distributional approximation leads to the following data-driven robust CIs:

- **Robust Bias Correction:**
  - Plug-in estimated errors: $\tilde{\chi}_{1-\alpha}^{rbc}(h_n, \hat{b}_n)$, where
    \[
    \tilde{\chi}_{1-\alpha}^{rbc}(h_n, \hat{b}_n) = \left\{ \hat{\tau}_p(h_n) - h_n^{p+1} \hat{B}_{n,p,q} \right\} \pm \Phi^{-1}_{1-\frac{\alpha}{2}} \sqrt{\tilde{V}_{n,p,q}^{bc}}
    \]
  - Fixed-matches estimated errors: $\tilde{\chi}_{1-\alpha}^{rbc}(h_n, \hat{b}_n)$, where
    \[
    \tilde{\chi}_{1-\alpha}^{rbc}(h_n, \hat{b}_n) = \left\{ \hat{\tau}_p(h_n) - h_n^{p+1} \hat{B}_{n,p,q} \right\} \pm \Phi^{-1}_{1-\frac{\alpha}{2}} \sqrt{\tilde{V}_{n,p,q}^{bc}}
    \]

  As above, for example, $\hat{h}_n \in (\hat{h}_{IK,n,p}, \hat{h}_{CCT,n,p}, \hat{h}_{CV,n,p})$ and $\hat{b}_n \in (\hat{b}_{IK,n,p+1,q}, \hat{b}_{CCT,n,p+1,q}, \hat{b}_{CV,n,p})$.

The exact formulas for $\tilde{V}_{n,p,q}^{bc} = \tilde{V}_{n,p,q}^{bc}(h_n, b_n)$ and $\hat{V}_{n,p,q}^{bc} = \hat{V}_{n,p,q}^{bc}(h_n, b_n)$ are given in Calonico, Cattaneo, and Titiunik (2014d).

### 2.5 Procedures implemented for sharp RD inference

The commands `rdbwselect` and `rdrobust` implement the following procedures:

- **rdbwselect** implements three bandwidth selectors for $h_{MSE,n,p}$:
  - $\hat{h}_{IK,n,p}$: IK implementation for $p$th-order local polynomial estimator.
  - $\hat{h}_{CCT,n,p}$: CCT implementation for $p$th-order local polynomial estimator. This is the default in `rdbwselect`.
  - $\hat{h}_{CV,n,p}$: CV implementation for $p$th-order local polynomial estimator.

- **rdbwselect** implements two bandwidth selectors for $b_{MSE,n,p,q}$: $\hat{b}_{IK,n,p,q}$ and $\hat{b}_{CCT,n,p,q}$.
  - $\hat{b}_{IK,n,p,q}$: IK-analog implementation for $p$th derivative of $q$th-order local polynomial estimator.
  - $\hat{b}_{CCT,n,p,q}$: CCT implementation for $p$th derivative of $q$th-order local polynomial estimator. This is the default in `rdbwselect`. 
• **rdrobust** implements two point estimators for $\tau$:
  - $\hat{\tau}_p(h_n)$: $p$th-order local polynomial estimator. This is the default in **rdrobust**.
  - $\hat{\tau}_p^{bc}(h_n, b_n)$: $p$th-order local polynomial estimator with $q$th-order local polynomial bias correction.

• **rdrobust** implements six CIs for $\tau$:
  - $\hat{\chi}_{1-\alpha}(h_n)$: no bias correction, conventional variance, plug-in residuals.
  - $\hat{\chi}_{1-\alpha}^{bc}(h_n, b_n)$: bias correction, conventional variance, plug-in residuals.
  - $\hat{\chi}_{1-\alpha}^{bc}(h_n, b_n)$: bias correction, conventional variance, fixed-matches residuals.
  - $\hat{\chi}_{1-\alpha}^{rbc}(h_n, b_n)$: bias correction, robust variance, plug-in residuals.
  - $\hat{\chi}_{1-\alpha}^{rbc}(h_n, b_n)$: bias correction, robust variance, fixed-matches residuals. This is the default in **rdrobust**.

We give details on the syntax of **rdrobust** and **rdbwselect** in section 3 and section 4, respectively. We include a complete empirical example that illustrates these methods and commands using real data in section 6. Further details on implementation and other technical issues are discussed in Imbens and Kalyanaraman (2012) and Calonico, Cattaneo, and Titiunik (2014d).

### 2.6 Extensions to other RD designs

As already mentioned, Calonico, Cattaneo, and Titiunik (2014d) also show how to construct analog robust CIs for average treatment effects (at the cutoff) in other RD contexts, including sharp kink RD, fuzzy RD, and fuzzy kink RD designs. For further discussion on these RD contexts, see, for example, Card et al. (2014) and Dong (2011).

Our implementation also includes the following empirically relevant extensions, among other possibilities:

• **Sharp Kink RD Design.** Here the estimand involves the derivative of the underlying regression functions at the cutoff (to a known scale) as opposed to their level. Using the notation introduced above, we write the generic estimand as
  \[
  \tau^{(s)} = \mu^{(s)}_+ - \mu^{(s)}_-
  \]
  where usually $s = 1$, and the population parameter of interest is $\tau^{(1)} / \kappa$ with $\kappa$ being a known scaling factor (that is, sharp kink RD estimand). The corresponding conventional local polynomial RD estimator is, up to the known scale $\kappa$,
  \[
  \hat{\tau}_p^{(s)}(h_n) = \hat{\mu}^{(s)}_{+}(h_n) - \hat{\mu}^{(s)}_{-}(h_n)
  \]
Robust data-driven inference in the regression-discontinuity design

where \( \hat{\mu}_{+}(h_n) = \mathbf{e}'_s \hat{\beta}_{+,p}(h_n) \) and \( \hat{\mu}_{-}(h_n) = \mathbf{e}'_s \hat{\beta}_{-,p}(h_n) \), with \( s \leq p \). All the results discussed herein extend naturally to this case, and our implementations allow for this possibility using the option `deriv()` to set the derivative order and the option `scalepar()` to set the value of \( \kappa \). See sections 3 and 4 for further details.

- **Fuzzy RD Design.** Here the estimand takes the form of a ratio of two sharp RD estimands: one for the main reduced-form equation (that is, the regression of \( Y_i \) on \( X_i \)) and the other for the first-stage equation (that is, the regression of \( T_i \) on \( X_i \), where \( T_i \) denotes actual treatment status). As discussed in Calonico, Cattaneo, and Titiunik (2014d), robust bias-corrected CIs can be constructed in this case as well. In our command, CIs for the fuzzy RD estimand are implemented with the option `fuzzy()`, as discussed in sections 3 and 4.

- **Fuzzy Kink RD Design.** Finally, the results also provide robust bias-corrected CIs in the context of a fuzzy kink RD design, where the estimand of interest is the ratio of two sharp kink RD estimands: one for the main reduced-form equation and the other for the first-stage equation. In our command, CIs for the fuzzy kink RD estimand are implemented when both the `deriv()` and `fuzzy()` options are specified jointly, as discussed in sections 3 and 4.

Note that `rdrobust` conducts sharp RD inference by default. See section 3 and section 4 for details on the syntax of `rdrobust` and `rdbwselect`, respectively, to construct CIs in the other cases.

### 2.7 Data-driven RD plots

The main aspects of the RD design can be summarized in an easy-to-interpret figure, which shows how an estimated regression function behaves for control (\( X_i < \tau \)) and treated (\( X_i \geq \tau \)) units relative to some summary of the actual data. This common RD plot gives an idea of overall fit while also exhibiting graphically the sharp RD estimate. In most empirical applications, this figure is constructed using “dots” for local sample means over nonoverlapping bins partitioning a restricted support of \( X_i \), together with two smooth “global” polynomial regression curve estimates for control and treatment units separately. The binned means are usually included to capture the behavior of the “cloud of points” and to show whether there are other discontinuities in the data away from the cutoff; the two global polynomial estimates are meant to give a flexible global approximation of \( \mu_{-}(x) \) and \( \mu_{+}(x) \). An example of this kind of plot using the data analyzed by Lee (2008) is shown in figure 1.
Calonico, Cattaneo, and Titunik (2014a) study these RD plots and develop several (optimal) choices of the number of bins under two partitioning schemes: evenly spaced and quantile-spaced partitions. Here we briefly summarize the main aspects of this approach.

RD plots use two main ingredients. First, polynomial regression curves are estimated to flexibly approximate the population conditional mean functions for control and treated units, usually over a large but restricted support of the running variable. Formally, these estimates are the $p$th-order global polynomials given by

$$
\hat{\mu}_{-p,1}(x) = r_p(x)\hat{\gamma}_{-p,1}(x_l) \quad \text{and} \quad \hat{\mu}_{+p,1}(x) = r_p(x)\hat{\gamma}_{+p,1}(x_u)
$$

with

$$
\hat{\gamma}_{-p,k}(x_l) = \arg\min_{\gamma \in \mathbb{R}^{p+1}} \sum_{i=1}^{n} \mathbb{1}(x_l < X_i \leq \bar{x})\{Y_i^k - r_p(X_i)\gamma\}^2
$$

$$
\hat{\gamma}_{+p,k}(x_u) = \arg\min_{\gamma \in \mathbb{R}^{p+1}} \sum_{i=1}^{n} \mathbb{1}(\bar{x} \leq X_i < x_u)\{Y_i^k - r_p(X_i)\gamma\}^2
$$

where $r_p(x) = (1, x, \ldots, x^p)'$ and $k = 1, 2$. Here $x_l$ and $x_u$ are the lower and upper limits on the support of the running variable, and they satisfy $x_l < \bar{x} \leq x_u$. In other words, $\hat{\mu}_{-p,1}(x)$ and $\hat{\mu}_{+p,1}(x)$ are $p$th-order global polynomials over the supports $[x_l, \bar{x}]$ and $[\bar{x}, x_u]$, respectively. This ingredient of the figure is easily implemented in practice, with common choices being $p = 4$ and $p = 5$.

Second, sample means are constructed over nonoverlapping regions of the support of the running variable $X_i$, for control and treatment units separately. These sample means provide an approximation of the population regression functions, but they also help visualize the dispersion of the data, which could be used to detect other potential
discontinuities away from the cutoff (as a form of a falsification test). To implement these estimators, we construct two evenly spaced partitions for control and treatment units separately, as follows,

\begin{align*}
P_{-,n} &= \bigcup_{j=1}^{J_{-,n}} P_{-,j,n} = [x_l, \bar{x}] \quad \text{and} \quad P_{+,n} = \bigcup_{j=1}^{J_{+,n}} P_{+,j,n} = [\bar{x}, x_u] \\
P_{-,j,n} &= \begin{cases} [x_l, p_{-,1}] & j = 1 \\ [p_{-,j-1}, p_{-,j}] & j = 2, \ldots, J_{-,n} - 1 \\ [p_{-,J_{-,n}-1}, \bar{x}] & j = J_{-,n} \end{cases} \\
P_{+,j,n} &= \begin{cases} [\bar{x}, p_{+,1}] & j = 1 \\ [p_{+,j-1}, p_{+,j}] & j = 2, \ldots, J_{+,n} - 1 \\ [p_{+,J_{+,n}-1}, x_u] & j = J_{+,n} \end{cases}
\end{align*}

with \( p_{-,0} < p_{-,1} < \cdots < p_{-,J_{-,n}} \) and \( p_{+,0} < p_{+,1} < \cdots < p_{+,J_{+,n}} \). Thus \( P_n = P_{-,n} \cup P_{+,n} \) forms a (disjoint) partition of \( (x_l, x_u) \) centered at \( \bar{x} \), which is assumed to become finer as the sample size grows (that is, \( J_{-,n} \to \infty \) and \( J_{+,n} \to \infty \)).

We consider the following two partitioning schemes:

- evenly spaced (ES): \( p_{-,j} = x_l + \frac{j}{J_{-,n}} (\bar{x} - x_l) \) and \( p_{+,j} = \bar{x} + \frac{j}{J_{+,n}} (x_u - \bar{x}) \)

- quantile spaced (QS): \( p_{-,j} = X_{-,\lfloor \frac{j}{J_{-,n}} \rfloor} \) and \( p_{+,j} = X_{+,\lfloor \frac{j}{J_{+,n}} \rfloor} \)

\( X_{-,i} \) and \( X_{+,i} \) denote the \( i \)-th quantile of the control and treatment subsamples, respectively, and \( \lfloor \cdot \rfloor \) denotes the floor function. To implement the ES or QS binning in practice, we must select the lengths of the bins for control and treated units, which are determined by \( J_{-,n} \) and \( J_{+,n} \), respectively. The resulting binned means can be written as

\begin{align*}
\tilde{\mu}_-(x; J_{-,n}) &= \frac{1}{J_{-,n}} \sum_{j=1}^{J_{-,n}} \mathbb{1}_{P_{-,j,n}}(x) \tilde{\gamma}_{-,j} \\
\tilde{\mu}_+(x; J_{+,n}) &= \frac{1}{J_{+,n}} \sum_{j=1}^{J_{+,n}} \mathbb{1}_{P_{+,j,n}}(x) \tilde{\gamma}_{+,j}
\end{align*}

with \( N_{-,j} = \sum_{i=1}^{n} \mathbb{1}_{P_{-,j,n}}(X_i) \), \( N_{+,j} = \sum_{i=1}^{n} \mathbb{1}_{P_{+,j,n}}(X_i) \), and \( \mathbb{1}_P(x) = \mathbb{1}(x \in P) \). These estimators are a simple version of the “nonparametric partitioning estimators”; see, for example, Cattaneo and Farrell (2013) and references therein.

Calonico, Cattaneo, and Titiunik (2014a) develop asymptotic expansions of the integrated variance and square-bias of the partitioning estimators under ES and QS partitioning, and they obtain two distinct (optimal) choices for \( J_{-,n} \) and \( J_{+,n} \) depending on the explicit goal chosen. They derive an integrated mean squared-error (IMSE)-optimal choice, which is explicitly tailored to produce binned sample means that “trace out” the underlying regression function. This choice is useful for falsification purposes (for
example, identifying potential discontinuities at values of the running variable different from the RD cutoff). The selectors of the number of bins for control and treatment units, respectively, take the form

\[ J_{\cdot,n}^* = \left\lfloor C_\cdot n^{1/3} \right\rfloor \quad \text{and} \quad J_{\cdot,n}^* = \left\lfloor C_\cdot n^{1/3} \right\rfloor \]

where the exact form of the constants depends on the partitioning scheme used, ES or QS, and \( \left\lfloor \cdot \right\rfloor \) denotes the ceiling operator.

Calonico, Cattaneo, and Titiunik (2014a) also propose a different rule for selecting the number of bins used in the ES and QS RD plots that is explicitly tailored to approximate the underlying variability of the raw data. This approach, which leads to a substantially larger number of bins than the IMSE-optimal choice, is useful to depict the data in a disciplined and objective way. The corresponding selectors for control and treatment units, respectively, are given by

\[ J_{\cdot,n}^* = \left\lfloor C_\cdot \frac{n}{\log(n)^2} \right\rfloor \quad \text{and} \quad J_{\cdot,n}^* = \left\lfloor C_\cdot \frac{n}{\log(n)^2} \right\rfloor \]

where here the specific constants are different from those in the IMSE-optimal choices, and they also vary depending on the partitioning scheme employed.

For either method, feasible plug-in rules can be constructed using preliminary estimators for the unknown constants \( C_\cdot \) that enter each of the number-of-bins selectors. Specifically, Calonico, Cattaneo, and Titiunik (2014a) propose implementations combining spacings and series estimation techniques, with the implementation varying depending on whether the outcome variable is continuously distributed or not. All details regarding these selectors, and the resulting data-driven RD plots constructed using them, can be found in Calonico, Cattaneo, and Titiunik (2014a).

Our command \texttt{rdplot} implements eight distinct data-driven RD plots that vary in the partitioning scheme used (ES or QS), the objective desired (approximating the regression function or depicting the variability of the data), and the nonparametric procedure used for implementation (spacings estimators or polynomial series estimators). We discuss the syntax details of \texttt{rdplot} in section 5, and we give an empirical example of this command in section 6.

3 The \texttt{rdrobust} command

In this section, we describe the syntax of the command \texttt{rdrobust} to conduct point estimation and robust inference for mean treatment effects in the RD design.

3.1 Description

\texttt{rdrobust} provides an array of local-polynomial-based inference procedures for mean treatment effects in the RD design. The user must specify the dependent and running
variable. This command permits fully data-driven inference by using the companion command \texttt{rdbwselect}, which can also be used as a stand-alone command. We describe \texttt{rdbwselect} below.

### 3.2 Syntax

\begin{verbatim}
rdrobust depvar runvar [ if ] [ in ] [ , c(cutoff) p(pvalue) q(qvalue) 
  deriv(dvalue) fuzzy(fuzzyvar) kernel(kernelfn) h(hvalue) b(bvalue) 
  rho(rhovalue) scalepar(scaleparvalue) bwselect(bwmethod) 
  scaleregul(scaleregulvalue) delta(deltavalue) cvgrid_min(cvgrid_minvalue) 
  cvgrid_max(cvgrid_maxvalue) cvgrid_length(cvgrid_lengthvalue) cvplot 
  vce(vcemethod) matches(nummatches) level(level) all ]
\end{verbatim}

where \texttt{depvar} is the dependent variable, and \texttt{runvar} is the running variable (also known as the score or forcing variable).

### 3.3 Options

- \texttt{c(cutoff)} specifies the RD cutoff. The default is \texttt{c(0)}.
- \texttt{p(pvalue)} specifies the order of the local polynomial to be used to construct the point estimator. The default is \texttt{p(1)} (local linear regression).
- \texttt{q(qvalue)} specifies the order of the local polynomial to be used to construct the bias correction. The default is \texttt{q(2)} (local quadratic regression).
- \texttt{deriv(dvalue)} specifies the order of the derivative of the regression functions to be estimated. The default is \texttt{deriv(0)} (sharp RD, or fuzzy RD if \texttt{fuzzy()} is also specified). Setting \texttt{deriv(1)} results in estimation of a kink RD design (up to scale), or fuzzy kink RD if \texttt{fuzzy()} is also specified.
- \texttt{fuzzy(fuzzyvar)} specifies the treatment status variable used to implement fuzzy RD estimation (or fuzzy kink RD if \texttt{deriv(1)} is also specified). The default is sharp RD design; hence, this option is not used. For fuzzy RD designs, bandwidths are estimated using sharp RD bandwidth selectors for the reduced-form outcome equation.
- \texttt{kernel(kernelfn)} specifies the kernel function used to construct the local polynomial estimators. \texttt{kernelfn} may be \texttt{triangular}, \texttt{epanechnikov}, and \texttt{uniform}. The default is \texttt{kernel(triangular)}.
- \texttt{h(hvalue)} specifies the main bandwidth, $h_n$, to be used to construct the RD point estimator. If not specified, this is computed by the companion command \texttt{rdbwselect}.
- \texttt{b(bvalue)} specifies the pilot bandwidth, $b_n$, used to construct the bias-correction estimator. If not specified, this is computed by the companion command \texttt{rdbwselect}.
rho(rhovalue) specifies the value of $\rho$ so that the pilot bandwidth, $b_n$, equals $b_n = h_n/\rho$. The default is rho(1) if $h_n$ is specified but $b_n$ is not.

scalepar(scaleparvalue) specifies the scaling factor for the RD parameter of interest. This option is useful when the population parameter of interest involves a known multiplicative factor (for example, sharp kink RD). The default is scalpar(1) (no scaling).

bwselect(bwmethod) specifies the bandwidth selection procedure to be used. By default, it computes both $h_n$ and $b_n$, unless $\rho$ is specified, in which case it computes only the $h_n$ and sets $b_n = h_n/\rho$. bwmethod may be one of the following:

- CCT for the bandwidth selector proposed by Calonico, Cattaneo, and Titiunik (2014d). The default is bwselect(CCT).
- IK for the bandwidth selector proposed by Imbens and Kalyanaraman (2012) (available for only sharp RD design).
- CV for the cross-validation method proposed by Ludwig and Miller (2007) (available for only sharp RD design).

scaleregul(scaleregulvalue) specifies the scaling factor for the regularization terms of CCT and IK bandwidth selectors. Setting scaleregulvalue(0) removes the regularization term from the bandwidth selectors. See companion command rdbwselect for more details. The default is scaleregul(1).

delta(deltavalue) specifies the quantile that defines the sample used in the cross-validation procedure. This option is used only if bwselect(CV) is specified. The default is delta(0.5), that is, the median of the control and treated subsamples.

cvgrid_min(cvgrid_minvalue) specifies the minimum value of the bandwidth grid used in the cross-validation procedure. This option is used only if bwselect(CV) is specified. See companion command rdbwselect for more details.

cvgrid_max(cvgrid_maxvalue) specifies the maximum value of the bandwidth grid used in the cross-validation procedure. This option is used only if bwselect(CV) is specified. See companion command rdbwselect for more details.

cvgrid_length(cvgrid_lengthvalue) specifies the bin length of the (evenly spaced) bandwidth grid used in the cross-validation procedure. This option is used only if bwselect(CV) is specified. See companion command rdbwselect for more details.

cvplot generates a graph of the cross-validation objective function. This option is used only if bwselect(CV) is specified. See companion command rdbwselect for more details.
Robust data-driven inference in the regression-discontinuity design

\texttt{vce(vcemethod)} specifies the procedure used to compute the variance–covariance matrix estimator. \texttt{vcemethod} may be one of the following:

- \texttt{nn} for nearest-neighbor matches residuals using \texttt{matches()}. This is the default option (with \texttt{matches(3)}, see below).
- \texttt{resid} for estimated plug-in residuals using $h_n$ bandwidth.

\texttt{matches(nummatches)} specifies the number of matches in the nearest-neighbor-based variance–covariance matrix estimator. This option is used only when nearest-neighbor matches residuals are used. The default is \texttt{matches(3)}.

\texttt{level(level)} is the confidence level for CIs. The default is \texttt{level(95)}.

\texttt{all} specifies that \texttt{rdrobust} report three different procedures:

i) conventional RD estimates with a conventional variance estimator;
ii) bias-corrected RD estimates with a conventional variance estimator; and
iii) bias-corrected RD estimates with a robust variance estimator.

4 The \texttt{rdbwselect} command

Here we describe the syntax for \texttt{rdbwselect}. This command implements the different bandwidth selection procedures for the local-polynomial regression-discontinuity estimators discussed above.

4.1 Description

\texttt{rdbwselect} implements several bandwidth selection procedures currently available for the RD design. The user must specify the dependent and running variable.

4.2 Syntax

\texttt{rdbwselect depvar runvar [ if ] [ in ] \[ , c(cutoff) p(pvalue) q(qvalue) deriv(dvalue) rho(rhovalue) kernel(kernelfn) bwselect(bwmethod) scaleregul(scaleregulvalue) delta(deltavalue) cvgrid_min(cvgrid_minvalue) cvgrid_max(cvgrid_maxvalue) cvgrid_length(cvgrid_lengthvalue) cvplot vce(vcemethod) matches(nummatches) all ]}

where \texttt{depvar} is the dependent variable, and \texttt{runvar} is the running variable (also known as the score or forcing variable).

4.3 Options

\texttt{c(cutoff)} specifies the RD cutoff. The default is \texttt{c(0)}.
\( p(p) \) specifies the order of the local polynomial to be used to construct the point estimator. The default is \( p(1) \) (local linear regression).

\( q(q) \) specifies the order of the local polynomial to be used to construct the bias correction. The default is \( q(2) \) (local quadratic regression).

driv(deriv) specifies the order of the derivative of the regression functions to be estimated. The default is \( \text{deriv}(0) \) (sharp RD, or fuzzy RD if \text{fuzzy}() is also specified). Setting \( \text{deriv}(1) \) results in estimation of a kink RD design (up to scale), or fuzzy kink RD if \text{fuzzy}() is also specified.

\( \rho(\rho) \) sets the pilot bandwidth, \( b_n \), equal to \( h_n \{ \rho \} \), where \( h_n \) is computed using the method and options chosen below.

\text{kernel}(kernelfn) specifies the kernel function used to construct the local polynomial estimators. Options are \text{triangular}, \text{epanechnikov}, and \text{uniform}. The default is \text{kernel}(\text{triangular}).

\text{bwselect}(bwmethod) specifies the bandwidth selection procedure to be used. By default, it computes both \( h_n \) and \( b_n \), unless \( \rho \) is specified, in which case it computes only \( h_n \) and sets \( b_n = h_n / \rho \). \text{bwmethod} may be one of the following:

- \text{CCT} for the bandwidth selector proposed by Calonico, Cattaneo, and Titiunik (2014d). The default is \text{bwselect}(\text{CCT}).

- \text{IK} for the bandwidth selector proposed by Imbens and Kalyanaraman (2012) (available for only sharp RD design).

- \text{CV} for the cross-validation method proposed by Ludwig and Miller (2007) (available for only sharp RD design).

\text{scaleregul}(scaleregulvalue) specifies the scaling factor for the regularization terms of \text{CCT} and \text{IK} bandwidth selectors. Setting \text{scaleregulvalue}(0) removes the regularization term from the bandwidth selectors. The default is \text{scaleregul}(1).

\text{delta}(deltavalue) specifies the quantile that defines the sample used in the cross-validation procedure. This option is used only if \text{bwselect}(\text{CV}) is specified. The default is \text{delta}(0.5), that is, the median of the control and treated subsamples.

\text{cvgrid_min}(cvgrid_minvalue) specifies the minimum value of the bandwidth grid used in the cross-validation procedure. This option is used only if \text{bwselect}(\text{CV}) is specified.

\text{cvgrid_max}(cvgrid_maxvalue) specifies the maximum value of the bandwidth grid used in the cross-validation procedure. This option is used only if \text{bwselect}(\text{CV}) is specified.

\text{cvgrid_length}(cvgrid_lengthvalue) specifies the bin length of the (evenly spaced) bandwidth grid used in the cross-validation procedure. This option is used only if \text{bwselect}(\text{CV}) is specified.
cvplot generates a graph of the cross-validation objective function. This option is used only if bwselect(CV) is specified.

vce(vcemethod) specifies the procedure used to compute the variance–covariance matrix estimator. This option is used only if the bwselect(CCT) or bwselect(IK) bandwidth procedure is used. vcemethod may be one of the following:

nn for nearest-neighbor matches residuals using matches(). This is the default (with matches(3), see below).

resid for estimated plug-in residuals using $h_n$ bandwidth.

matches(nummatches) specifies the number of matches in the nearest-neighbor-based variance–covariance matrix estimator. This option is used only when nearest-neighbor matches residuals are used. The default is matches(3).

all implements all three bandwidth selection procedures; see bwselect() above.

5 The rdplot command

In this section, we describe the syntax of the rdplot command, which implements different data-driven RD plots using either evenly spaced or quantile-spaced binning. The number of bins on these plots is chosen either i) to approximate the underlying regression function (IMSE-optimal selectors) or ii) to mimic the underlying variability of the raw data. For further details on these approaches, see Calonico, Cattaneo, and Titiumik (2014a).

5.1 Description

rdplot implements several selectors of the number of bins used to construct RD plots using either an evenly spaced or a quantile-spaced partitioning of the support of the running variable $X_i$. Two distinct methods are implemented: one to approximate the regression function and one to display the variability of the data in an objective way. The user must specify the dependent and running variables.

5.2 Syntax

rdplot depvar runvar [if] [in] [ , c(cutoff) p(pvalue) numbinl(numbinlvalue) numbinr(numbinrvalue) binselect(binmethod) lowerend(xlvalue) upperend(xuvalue) scale(scalevalue) scalel(scalelvalue) scaler(scalervalue) generate(idname meanxname meanyname) graph_options(gphopts) hide ]

where depvar is the dependent variable, and runvar is the running variable (also known as the score or forcing variable).
5.3 Options

c(cutoff) specifies the RD cutoff. The default is c(0).

p(pvalue) specifies the order of the global polynomial used to approximate the population conditional mean functions for control and treated units. The default is p(4).

numbinl(numbinlvalue) specifies the number of bins used to the left of the cutoff, denoted \( J^- \). If not specified, \( J^- \) is estimated using the method and options chosen below.

numbinr(numbinrvalue) specifies the number of bins used to the right of the cutoff, denoted \( J^+ \). If not specified, \( J^+ \) is estimated using the method and options chosen below.

binselect(binmethod) specifies the procedure to select the number of bins. This option is available only if \( J^- \) and \( J^+ \) are not set manually. \( binmethod \) may be one of the following:

- es specifies the IMSE-optimal evenly spaced method using spacings estimators.
- espr specifies the IMSE-optimal evenly spaced method using polynomial regression.
- esmv specifies the mimicking variance evenly spaced method using spacings estimators; the default.
- esmvpr specifies the mimicking-variance evenly spaced method using polynomial regression.
- qs specifies the IMSE-optimal quantile-spaced method using spacings estimators.
- qspr specifies the IMSE-optimal quantile-spaced method using polynomial regression.
- qsmv specifies the mimicking-variance quantile-spaced method using spacings estimators.
- qsmvpr specifies the mimicking-variance quantile-spaced method using polynomial regression.

lowerend(xlvalue) specifies the lower bound for \( indepvar \) to the left of the cutoff. The default is the minimum value in sample.

upperend(xuvalue) specifies the upper bound for \( indepvar \) to the right of the cutoff. The default is the maximum value in sample.

scale(scalevalue) specifies a multiplicative factor to be used with the optimal number of bins selected. Specifically, the number of bins used for the treatment and control groups will be \( \text{scale}(scalevalue) \times J^+ \) and \( \text{scale}(scalevalue) \times J^- \), where \( J^- \) and \( J^+ \) denote the optimal numbers of bins originally computed for each group. The default is \( \text{scale}(1) \).
Robust data-driven inference in the regression-discontinuity design

scalel(scalevalue) specifies a multiplicative factor to be used with the optimal number of bins selected to the left of the cutoff. The number of bins used will be scalel(scalevalue) \times \hat{J}_{-n}. The default is scalel(1).

scaler(scalervalue) specifies a multiplicative factor to be used with the optimal number of bins selected to the right of the cutoff. The number of bins used will be scaler(scalervalue) \times \hat{J}_{+n}. The default is scaler(1).

generate(idname meanxname meanyname) generates new variables storing the results:

idname specifies the name of a new generated variable with a unique bin id that identifies the chosen bins. This variable indicates the bin (between lowerend() and upperend()) to which each observation belongs. Negative natural numbers are assigned to observations to the left of the cutoff, and positive natural numbers are assigned to observations to the right of the cutoff.

meanxname specifies the name of a new generated variable (of the same length as idname) with the middle point of the running variable within each chosen bin.

meanyname specifies the name of a new generated variable (of the same length as idname) with the sample mean of the outcome variable within each chosen bin.

graph_options(gphopts) specifies graphical options to be passed on to the underlying graph command.

hide omits the RD plot.

6 Illustration of methods

We illustrate our commands using an extract of the dataset constructed by Cattaneo, Frandsen, and Titiunik (Forthcoming); the dataset comes from a study on party advantages in U.S. Senate elections for the period 1914–2010. In particular, we focus here on the RD effect of the Democratic party winning a U.S. Senate seat on the vote share obtained in the following election for that same seat. The unit of observation is the state. This empirical illustration is analogous to the one presented by Lee (2008) for U.S. House elections.

The dataset rdrobust.rdsenate contains two variables: vote and margin. The variable margin ranges from −100 to 100 and records the Democratic party’s margin of victory in the statewide election for a given U.S. Senate seat, which is defined as the vote share of the Democratic party minus the vote share of its strongest opponent. The variable vote ranges from 0 to 100 and records the Democratic vote share in the following election for the same seat (6 years later). Thus observations for years 2008 and 2010 are missing vote. When margin is above zero, the Democratic party wins the election for that seat; otherwise, it loses. As is usual in the literature, we exploit this discontinuity in incumbency status that occurs at margin = 0 to estimate the incumbency advantage of parties with an RD design.
The dataset has a total of 1,297 complete observations. We load the database and present basic summary statistics of these two variables.

```
. use rdrobust_rdsenate.dta
. summarize vote margin
```

<table>
<thead>
<tr>
<th>Variable</th>
<th>Obs</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>vote</td>
<td>1297</td>
<td>52.66627</td>
<td>18.12219</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>margin</td>
<td>1390</td>
<td>7.171159</td>
<td>34.32488</td>
<td>-100</td>
<td>100</td>
</tr>
</tbody>
</table>

To further explore the available data, we use `rdplot` to construct an automatic plot of the RD design.

```
. rdplot vote margin,
   > graph_options(title(RD Plot - Senate elections data)
   >                      ytitle(Vote share in election at time t+1)
   >                      xtitle(Vote share in election at time t))
```

<table>
<thead>
<tr>
<th>Number of bins for RD estimates. Method: Mimicking Variance evenly spaced using spacings estimators.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cutoff c = 0</td>
</tr>
<tr>
<td>Number of observations</td>
</tr>
<tr>
<td>Polynomial order</td>
</tr>
<tr>
<td>Chosen scale</td>
</tr>
<tr>
<td>Selected bins</td>
</tr>
<tr>
<td>Bin length</td>
</tr>
<tr>
<td>IMSE-optimal bins</td>
</tr>
<tr>
<td>Mimicking Variance bins</td>
</tr>
<tr>
<td>Relative to IMSE-optimal:</td>
</tr>
<tr>
<td>Implied scale</td>
</tr>
<tr>
<td>WIMSE variance weight</td>
</tr>
<tr>
<td>WIMSE bias weight</td>
</tr>
</tbody>
</table>

Figure 2 is constructed using the default options in the command `rdplot`, which produce an RD plot that has evenly spaced bins that mimic the underlying variability of the data and is implemented using spacings estimators. Using the notation introduced above, we see that the number of optimal bins for control and treatment units is $\hat{J}_{-,n} = 15$ and $\hat{J}_{+,n} = 35$, respectively, implying bin lengths of 6.661 and 2.856 percentage points, respectively. The global polynomial is constructed using a 4th-degree polynomial $[p = 4$ for $\mu_{-,p,1}(x)$ and $\mu_{+,p,1}(x)]$. The output table also reports the IMSE-optimal number of bins (first row of the middle panel) and the multiplicative factor (scale) associated with the selected number of bins (taking the IMSE-optimal value as a reference). This is shown in the last row of the first panel. Finally, the bottom panel includes the IMSE weights that correspond to the selected choice of the number of bins—see Calonico, Cattaneo, and Titiunik (2014a, sec. 3.3.1) for additional details.
Next, we construct an alternative RD plot using evenly spaced bins selected to trace out the underlying regression function (that is, IMSE-optimal selectors) and implemented using spacings estimators. The resulting plot is given in figure 3, which shows how the (local) binned sample means indeed seem to approximate the underlying regression function well (taking the global polynomial fit as a benchmark). This does not provide evidence of potential discontinuities away from the cutoff in the underlying regression functions.

```
. rdplot vote margin, binselect(es)
> graph_options(title(RD plot - Senate elections data)
> ytitle(Vote share in election at time t+1)
> xtitle(Vote share in election at time t))
```

Number of bins for RD estimates.
Method: IMSE-optimal evenly spaced method using spacings estimators.

<table>
<thead>
<tr>
<th>Cutoff c = 0</th>
<th>Left of c</th>
<th>Right of c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of observations</td>
<td>595</td>
<td>702</td>
</tr>
<tr>
<td>Polynomial order</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Chosen scale</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>Selected bins</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Bin length</td>
<td>12.490</td>
<td>11.107</td>
</tr>
<tr>
<td>IMSE-optimal bins</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Mimicking Variance bins</td>
<td>15</td>
<td>35</td>
</tr>
<tr>
<td>Relative to IMSE-optimal:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implied scale</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>WIMSE variance weight</td>
<td>0.500</td>
<td>0.500</td>
</tr>
<tr>
<td>WIMSE bias weight</td>
<td>0.500</td>
<td>0.500</td>
</tr>
</tbody>
</table>
The last two figures show different features of the underlying data and research design. Figure 2 gives a disciplined scatterplot of the data, which seeks to represent the overall variability of the raw data. This figure uses more than the IMSE-optimal number of bins (that is, it undersmooths the nonparametric partitioning estimator), which induces more variability in the underlying binned sample means. This gives a visual “cloud of points” on top of the polynomial approximation to the regression function. On the other hand, figure 3 uses the IMSE-optimal number of bins specifically tailored to produce an estimator that approximates the underlying regression function well. In this approach, the optimal number of bins is selected to balance squared bias and variance to approximate the underlying conditional expectation globally.

Finally, to illustrate some of the other features of the command \texttt{rdplot}, we present figure 4. Here the RD plot is constructed using the quantile-spaced binning approach. In this case, the number of optimal bins for control and treatment units is $\hat{J}_{-n} = 28$ and $\hat{J}_{+n} = 49$, respectively.
Robust data-driven inference in the regression-discontinuity design

```stata
* rdplot vote margin, binselect(qsmv)
> graph_options(title(RD plot - Senate elections data)
>       ytitle(Vote share in election at time t+1)
>       xtitle(Vote share in election at time t))
```

Number of bins for RD estimates.
Method: Mimicking Variance quantile spaced using spacings estimators.

<table>
<thead>
<tr>
<th>Cutoff c = 0</th>
<th>Left of c</th>
<th>Right of c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of observations</td>
<td>595</td>
<td>702</td>
</tr>
<tr>
<td>Polynomial order</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Chosen scale</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>Selected bins</td>
<td>28</td>
<td>49</td>
</tr>
<tr>
<td>Bin length</td>
<td>3.569</td>
<td>2.040</td>
</tr>
<tr>
<td>IMSE-optimal bins</td>
<td>21</td>
<td>16</td>
</tr>
<tr>
<td>Mimicking Variance bins</td>
<td>28</td>
<td>49</td>
</tr>
<tr>
<td>Relative to IMSE-optimal:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implied scale</td>
<td>1.333</td>
<td>3.062</td>
</tr>
<tr>
<td>WIMSE variance weight</td>
<td>0.297</td>
<td>0.034</td>
</tr>
<tr>
<td>WIMSE bias weight</td>
<td>0.703</td>
<td>0.966</td>
</tr>
</tbody>
</table>

Figure 4. Automatic RD plot using a quantile-spaced approach
Next, we conduct fully data-driven RD treatment-effects estimation and inference. The command `rdrobust`, using its default options, gives the following output:

```
  . rdrobust vote margin
Preparing data.
  Computing bandwidth selectors.
  Computing variance-covariance matrix.
  Computing RD estimates.
Estimation completed.
  Sharp RD estimates using local polynomial regression.

Cutoff c = 0 Left of c Right of c Number of obs = 1297
  Number of obs 343 310
  Order loc. poly. (p) 1
  Order bias (q) 2
  BW loc. poly. (h) 16.794 16.794
  BW bias (b) 27.437 27.437
  rho (h/b) 0.612 0.612


| Method   | Coef. Std. Err. | z | P>|z| | [95% Conf. Interval] |
|----------|-----------------|---|------|---------------------|
| Conventional | 7.4253         | 1.4964 | 4.9656 | 0.000 | 4.49446, 10.3561 |
| Robust    | -               | -     | 4.2675 | 0.000 | 4.06975, 10.9833 |
```

These results contain a variety of information organized into two panels. The upper panel of the output table contains a summary of the main choices selected to construct the RD treatment-effect estimators, while the lower panel includes the main estimation results. Specifically, using the notation introduced above, this table shows the following:

1. The total number of observations is 1,297, with effective 343 control and 310 treated units (given the bandwidth \( h_n \) chosen; see below). The estimation is conducted using a local linear \((p = 1)\) estimator with a local-quadratic \((q = 2)\) bias-correction estimate, with a triangular kernel. The variance estimators are the robust ones proposed by Calonico, Cattaneo, and Titiunik (2014d), computed using three nearest neighbors.

2. The bandwidth selection procedure is the one proposed by Calonico, Cattaneo, and Titiunik (2014d) leading to

\[
\hat{h}_{CCT,n,p} = 16.794 \quad \text{and} \quad \hat{b}_{CCT,n,p+1,q} = 27.437
\]

with \( p = 1 \) and \( q = 2 \).

3. The point estimator and robust CI are

\[
\hat{\tau}_p (\hat{h}_{CCT,n,1}) = 7.4253, \quad \hat{\tau}^{rc}_{1-\alpha} (\hat{h}_{CCT,n,1}, \hat{b}_{CCT,n,2,2}) = [4.06975, 10.9833]
\]

with \( \alpha = 0.05 \).
Robust data-driven inference in the regression-discontinuity design

The command `rdrobust` also offers a more detailed output, which includes all the point estimators, variance estimators, and CIs discussed in section 2. These results are retrieved by including the option `all`, as follows:

```
. rdrobust vote margin, all
Preparing data.
Computing bandwidth selectors.
Computing variance-covariance matrix.
Computing RD estimates.
Estimation completed.
```

Sharp RD estimates using local polynomial regression.

<table>
<thead>
<tr>
<th>Cutoff c = 0</th>
<th>Left of c</th>
<th>Right of c</th>
<th>Number of obs = 1297</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>343</td>
<td>310</td>
<td></td>
</tr>
<tr>
<td>Number of obs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Order loc. poly. (p)</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Order bias (q)</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>BW loc. poly. (h)</td>
<td>16.794</td>
<td>16.794</td>
<td></td>
</tr>
<tr>
<td>BW bias (b)</td>
<td>27.437</td>
<td>27.437</td>
<td></td>
</tr>
<tr>
<td>rho (h/b)</td>
<td>0.612</td>
<td>0.612</td>
<td></td>
</tr>
</tbody>
</table>

Outcome: vote. Running variable: margin.

| Method             | Coef. Std. Err. | z    | P>|z|  | [95% Conf. Interval] |
|--------------------|-----------------|------|------|----------------------|
| Conventional       | 7.4253          | 1.4954 | 4.9656 | 0.000 | 4.49446 10.3561    |
| Robust             | -               | -    | 4.2675 | 0.000 | 4.06975 10.9833    |

All estimates.

| Method             | Coef. Std. Err. | z    | P>|z|  | [95% Conf. Interval] |
|--------------------|-----------------|------|------|----------------------|
| Conventional       | 7.4253          | 1.4954 | 4.9656 | 0.000 | 4.49446 10.3561    |
| Bias-corrected     | 7.5265          | 1.4954 | 5.0333 | 0.000 | 4.59569 10.4574    |
| Robust             | 7.5265          | 1.7637 | 4.2675 | 0.000 | 4.06975 10.9833    |

This detailed output contains an additional table located at the bottom, relative to the default output. Using our notation and the options specified in the case above, this extra table follows the structure of table 1:

| Method             | Coef. Std. err. | z    | P>|z|  | [95% Conf. Interval] |
|--------------------|-----------------|------|------|----------------------|
| Conventional       | \( \hat{\tau}_p(h_n) \) | \( \sqrt{\hat{V}_p/nh_n} \) | \( \cdot \) | \( \cdot \) | \( \hat{C}_1-\alpha(h_n) \) |
| Bias corrected     | \( \hat{\tau}_{p,q}(h_n,b_n) \) | \( \sqrt{\hat{V}_{p,q}/nh_n} \) | \( \cdot \) | \( \cdot \) | \( \hat{C}_{1-\alpha}(h_n,b_n) \) |
| Robust             | \( \hat{\tau}_{p,q}(h_n,b_n) \) | \( \sqrt{\hat{V}_{p,q,n,p,q}} \) | \( \cdot \) | \( \cdot \) | \( \hat{C}_{1-\alpha}(h_n,b_n) \) |

Note: In the output above, \( p = 1, q = 2, h_n = \hat{h}_{CCT,n,p}, b_n = \hat{b}_{CCT,n,p+1,q}, \) and \( \alpha = 0.05. \)
Finally, we explore all the bandwidth selection procedures contained in our package. Specifically, we can use our companion `rdbwselect` to compare the CCT bandwidth selectors with the IK and CV bandwidth selectors:

```
    . rdbwselect vote margin, all
    Computing CCT bandwidth selector.
    Computing IK bandwidth selector.
    Computing CV bandwidth selector.

    Bandwidth estimators for RD local polynomial regression
    Cutoff c = 0
     Left of c     Right of c
     Number of obs  595     702
    Order loc. poly. (p)  1     1
    Order bias (q)  2     2
    Range of margin 99.921 99.964

    Number of obs = 1297
    NN matches = 3
    Kernel type = Triangular
    Min BW grid = 0.69039
    Max BW grid = 99.92107
    Length BW grid = 4.96153

    Method   h      b       rho
    CCT  16.79369 27.43745 .612072
    IK  15.75038 16.47286 .956141
    CV  35.42113 NA       NA
```

Here we used the option `all`, which computes the three bandwidth selectors briefly discussed above. In the case of \( h_n \), these choices range from 16.79369 to 35.42113. In the case of \( b_n \), we obtain \( \hat{b}_{CCT,n,2.2} = 27.43745 \) and \( \hat{b}_{IK,n,2.2} = 16.47286 \) for the CCT and IK methods, respectively. Notice that the option CV is currently not available for derivative estimation. To further understand the performance of the CV approach, we include a graph of the CV objective function over the grid being considered in figure 5. This is done using the option `cvplot` as follows (in this example, we also changed the grid features to obtain a better plot and to show this additional functionality in action):

```
    . rdbwselect vote margin, bwselect(CV) cvplot cvgrid_min(10) cvgrid_max(80)
    Computing CV bandwidth selector.

    Bandwidth estimators for RD local polynomial regression
    Cutoff c = 0
     Left of c     Right of c
     Number of obs  595     702
    Order loc. poly. (p)  1     1
    Order bias (q)  2     2
    Range of margin 99.921 99.964

    Number of obs = 1297
    NN matches = 3
    Kernel type = Triangular
    Min BW grid = 10.00000
    Max BW grid = 80.00000
    Length BW grid = 3.50000

    Method   h    b     rho
    CV  34.5  NA     NA
```
As discussed above, our commands have many other options. For example, for the main command `rdrobust`, we have the following additional examples (output is not provided to conserve space):

1. `rdrobust vote margin, kernel(uniform)`
   Estimation using uniform kernel.

2. `rdrobust vote margin, bwselect(IK)`
   Estimation using the IK bandwidth selectors.

3. `rdrobust vote margin, bwselect(CV)`
   Estimation using the CV bandwidth selector (and $\rho = 1$).

4. `rdrobust vote margin, h(15) rho(0.8)`
   Estimation using $h_n = 15$ and $b_n = h_n/\rho = 15/0.8 = 18.75$.

5. `rdrobust vote margin, p(2) q(4)`
   Estimation using $p = 2$ and $q = 4$.

6. `rdrobust vote margin, vce(resid)`
   Estimation using plug-in residuals estimates in the variance–covariance estimator.
Finally, our commands can also be used to conduct inference in other RD design settings. For example, let’s assume y is the output variable, t is the treatment status variable, and x is the running variable:

1. \texttt{rdrobust y x, deriv(1)}  
   Estimation for sharp kink RD.

2. \texttt{rdrobust y x, fuzzy(t)}  
   Estimation for fuzzy RD.

3. \texttt{rdrobust y x, fuzzy(t) deriv(1)}  
   Estimation for fuzzy kink RD.

### 6.1 Generating tables with \texttt{outreg2}

The output generated by \texttt{rdrobust} can be used to construct tables with available Stata packages because the main estimates and chosen parameters are stored as \texttt{e()} results. This can facilitate the creation of tables and postestimation handling of numerical results, but it should not be used to perform joint or other postestimation hypothesis tests among the coefficients in the stored matrices.

To illustrate how to produce output tables with \texttt{outreg2} (Wada 2005), we consider the following three estimation specifications:

\begin{verbatim}
. rdrobust vote margin, kernel(uniform)  
. rdrobust vote margin, bwselect(CV)  
. rdrobust vote margin, p(2) q(3)
\end{verbatim}

Then we can easily obtain the following table in \texttt{LaTeX} format:

<table>
<thead>
<tr>
<th>VARIABLES</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD_Estimate</td>
<td>7.403</td>
<td>7.167</td>
<td>7.943</td>
</tr>
<tr>
<td>Observations</td>
<td>489</td>
<td>1013</td>
<td>828</td>
</tr>
<tr>
<td>Robust 95% CI</td>
<td>[4.16 ; 11.46]</td>
<td>[4.42 ; 10.59]</td>
<td>[4.03 ; 11.99]</td>
</tr>
<tr>
<td>Kernel Type</td>
<td>Uniform</td>
<td>Triangular</td>
<td>Triangular</td>
</tr>
<tr>
<td>BW Type</td>
<td>CCT</td>
<td>CV</td>
<td>CCT</td>
</tr>
<tr>
<td>Conventional Std. Error</td>
<td>1.640</td>
<td>1.103</td>
<td>1.818</td>
</tr>
<tr>
<td>Conventional p-value</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Robust p-value</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Order Loc. Poly. (p)</td>
<td>1.000</td>
<td>1.000</td>
<td>2.000</td>
</tr>
<tr>
<td>Order Bias (q)</td>
<td>2.000</td>
<td>2.000</td>
<td>3.000</td>
</tr>
<tr>
<td>BW Loc. Poly. (h)</td>
<td>11.095</td>
<td>35.421</td>
<td>24.328</td>
</tr>
<tr>
<td>BW Bias (b)</td>
<td>22.790</td>
<td>35.421</td>
<td>35.572</td>
</tr>
</tbody>
</table>

This table is constructed using the following command after each specification:

\begin{verbatim}
outreg2 using table1, addstat(Conventional Std. Err., e(se_cl),  
Conventional p-value, e(pv_cl), Robust p-value, e(pv_rb), Order Loc.  
Poly. (p), e(p), Order Bias (q), e(q), BW Loc. Poly. (h), e(h_bw), BW  
Bias (b), e(b_bw)) addtext(Robust`e(level)`% CI, `e(ci_rb)`), Kernel  
Type, `e(kernel)`), BW Type, `e(bwsel)`), Observations, `e(N)` noobs  
nomaster nomotes adec(3)
\end{verbatim}
Robust data-driven inference in the regression-discontinuity design

The appropriate additional options (for example, `replace`, `append`, `tex`, etc.) can be used as needed. Additional statistics can also be included. The complete list of estimation results can be obtained by running `ereturn list` after `rdrobust`.

Finally, we can also construct tables in a more standard format if we use `outreg2` after running `rdrobust` with the `all` option. For example, for the same specifications as before, we obtain the following table in \LaTeX\ format:

```
<table>
<thead>
<tr>
<th>VARIABLES</th>
<th>(1) vote</th>
<th>(2) vote</th>
<th>(3) vote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional</td>
<td>7.403***</td>
<td>7.167***</td>
<td>7.943***</td>
</tr>
<tr>
<td></td>
<td>(1.640)</td>
<td>(1.103)</td>
<td>(1.818)</td>
</tr>
<tr>
<td>Bias-corrected</td>
<td>7.810***</td>
<td>7.502***</td>
<td>8.006***</td>
</tr>
<tr>
<td></td>
<td>(1.640)</td>
<td>(1.103)</td>
<td>(1.818)</td>
</tr>
<tr>
<td>Robust</td>
<td>7.810***</td>
<td>7.502***</td>
<td>8.006***</td>
</tr>
<tr>
<td></td>
<td>(1.861)</td>
<td>(1.575)</td>
<td>(2.031)</td>
</tr>
<tr>
<td>Observations</td>
<td>489</td>
<td>1,013</td>
<td>828</td>
</tr>
</tbody>
</table>
```

Standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1

Our companion replication file (`rdrobust_illustration.do`) includes the exact syntax of all the examples discussed above.

7 Conclusion

In this article, we discussed the implementation of several data-driven local-polynomial-based (robust) inference procedures in the RD design. We introduced the commands `rdrobust`, `rdbwselect`, and `rdplot`, which together offer an array of data-driven non-parametric inference methods for performing empirical work in RD applications. Our implementations cover average treatment effects at the cutoff in the sharp RD, sharp kink RD, fuzzy RD, and fuzzy kink RD designs, among other possibilities.

A companion R package is also available—see Calonico, Cattaneo, and Titunik (2014b) for a description.

8 Acknowledgments

We specially thank David Drukker for detailed comments and suggestions that greatly improved our implementation. We also received very useful comments from Richard Anderson, Sutirtha Bagchi, Devin Caughey, Pablo Celhay, Jose Galdo, Andrew Hall, Marko Klašnja, Tae Hoon Kim, Benjamin Lutz, Zhuan Pei, László Sándor, Jeff Smith, Ugo Troiano, and a referee on previous versions of this article. The authors gratefully acknowledge financial support from the National Science Foundation through grant SES-1357561.
9 References


Robust data-driven inference in the regression-discontinuity design


About the authors

Sebastian Calonico is an assistant professor of economics at the University of Miami.

Matias D. Cattaneo is an associate professor of economics at the University of Michigan.

Rocío Titiunik is an assistant professor of political science at the University of Michigan.
adjcatlogit, ccrlogit, and ucrlogit: Fitting ordinal logistic regression models

Morten W. Fagerland
Oslo Centre for Biostatistics and Epidemiology
Research Support Services
Oslo University Hospital
Oslo, Norway
morten.fagerland@medisin.uio.no

Abstract. In this article, I present three commands that perform adjacent-category logistic regression (adjcatlogit), constrained continuation-ratio logistic regression (ccrlogit), and unconstrained continuation-ratio logistic regression (ucrlogit) for ordered response data.

Keywords: st0367, adjcatlogit, ccrlogit, ucrlogit, ordinal models, ordered regression, logistic models, adjacent category, continuation ratio

1 Introduction

Ordinal response regression models are used to describe the relationship between an ordered categorical response (dependent) variable and one or more explanatory (independent) variables. The literature on ordinal regression models includes general textbooks such as those by McCullagh and Nelder (1989); Agresti (2010); and Hosmer, Lemeshow, and Sturdivant (2013). It also includes review articles such as those by Greenland (1994) and Ananth and Kleinbaum (1997). There are several models we can use. When choosing a model, we must first select a link function, which describes the functional relationship between the dependent variable and the independent variables. We then must choose which response categories to compare.

The main commands for ordinal regression are ologit and oprobit. ologit fits proportional-odds logistic regression models, also called parallel-lines models. The link function is the logit transformation, which is also used in (ordinary) binary logistic regression (logit or logistic) and multinomial logistic regression (mlogit). oprobit fits ordered probit regression models, where the link function is the standard normal cumulative distribution function. Also available is the stereotype logistic regression model (slogit), which is a compromise between the multinomial and ordered logistic regression models (see [R] slogit).

In this article, I focus on ordinal models that use the logit link, that is, ordinal logistic regression models. Logistic models have several advantages, including mathematical flexibility and ease of use. Also the exponential form of the regression coefficients can be interpreted as odds ratios (ORs) (Hosmer, Lemeshow, and Sturdivant 2013). Several different logistic models are possible. The proportional odds model (ologit) compares two
sets of response categories: an equal or smaller response versus a larger response. The
adjacent category model compares each response category with the next larger response
category. The constrained and unconstrained continuation-ratio models compare each
response category with all lower response categories. The unconstrained continuation-
ratio model defines \( c-1 \) regression coefficients for each independent variable—where \( c \)
is the number of response categories—whereas the other models describe the effect of each
independent variable using a single regression coefficient. When choosing a model for
a particular problem, users should consider which model provides the most informative
comparisons for the subject matter as well as the desired amount of model flexibility.

Here I describe three commands for adjacent-category logistic regression (\texttt{adjcatlogit}),
constrained continuation-ratio logistic regression (\texttt{ccrlogit}), and uncon-
strained continuation-ratio logistic regression (\texttt{ucrlogit}).

2 Model definitions

Let the dependent variable \( Y \) take on \( c \) possible values \( 1, 2, \ldots, c \), and let
\( \mathbf{x} = (x_1, x_2, \ldots, x_p) \) denote a vector of \( p \) independent variables. The conditional prob-
ability of a response equal to category \( j \) given \( \mathbf{x} \) is denoted by \( \pi_j = P(Y = j | \mathbf{x}) \) for
\( j = 1, 2, \ldots, c \). Each model is defined through the \( c-1 \) logits (logit equations) \( g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_{c-1}(\mathbf{x}) \), which relate a set of intercepts (\( \alpha \)s) and regression coefficients (\( \beta \)s)
to the probability of the response categories. I define the logits for each model in
sections 2.1–2.3.

2.1 The adjacent-category logistic regression model

The adjacent category model compares each response category (except the first) with
the next larger response category.

\[
g_j(\mathbf{x}) = \log \left\{ \frac{P(Y = j + 1 | \mathbf{x})}{P(Y = j | \mathbf{x})} \right\} = \alpha_j + \mathbf{\beta}' \mathbf{x} \quad j = 1, \ldots, c - 1
\]

The regression coefficients, \( \mathbf{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)' \), are constant across the logits, whereas
the intercepts (\( \alpha_j \)) are not. For an independent variable \( x_k \), we may interpret \( \exp(\beta_k) \)
as the OR for a one-unit increase in \( x_k \), comparing response category \( j+1 \) with response
category \( j \). Thus we have the following equalities,

\[
\exp(\beta_k) = \text{OR}(2, 1) = \text{OR}(3, 2) = \cdots = \text{OR}(c, c - 1)
\]

where \( \text{OR}(a, b) \) denotes the OR for comparing category \( a \) with category \( b \). As shown in
Hosmer, Lemeshow, and Sturdivant (2013, 290–291), we also have that

\[
\log \{ \text{OR}(j, 1) \} = (j - 1) \times \{ \text{OR}(2, 1) \} \quad j = 3, \ldots, c \quad (1)
\]

Equation (1) provides the rationale for fitting the adjacent category model via a con-
strained multinomial model. The necessary constraints are \( \beta_{jk} = (j - 1) \times \beta_{2k} \), for
Formulas for the conditional probabilities of the adjacent category model were derived in Fagerland and Hosmer (2014) and are given as

$$\pi_1 = \frac{1}{1 + \theta}$$

and

$$\pi_{j+1} = \frac{\theta_j}{1 + \theta} \quad j = 1, \ldots, c - 1$$

where $\theta = \sum_{k=1}^{c-1} \theta_k$ and

$$\theta_j = \exp \left\{ \sum_{k=1}^{j} g_k(x) \right\} \quad j = 1, \ldots, c - 1$$

### 2.2 The constrained continuation-ratio logistic regression model

The constrained continuation-ratio model compares each response category with all lower response categories.

$$g_j(x) = \log \left\{ \frac{P(Y = j|x)}{P(Y < j|x)} \right\} = \alpha_j + \beta^T x \quad j = 2, \ldots, c$$

As with the adjacent category model, the regression coefficients are constant across the logits, and we may describe the relationship between an independent variable $x_k$ and the dependent variable $Y$ by a single coefficient or OR. The following interpretations apply:

$$\exp(\beta_k) = \text{OR}(2,1) = \text{OR}(3,1 \ldots 2) = \cdots = \text{OR}(c,1 \ldots c - 1)$$

Note that we can obtain another version of the constrained continuation-ratio model by substituting the denominator in (2) with $P(Y > j|x)$ and possibly changing the sign of $\beta^T x$. The resulting model can be fit with the user-written program ocratio (Wolfe 1998). The two models are, however, not equivalent (Hosmer, Lemeshow, and Sturdivant 2013, 291). Both models can be fit with a generalized linear model formulation.
Fitting ordinal logistic regression models

Fagerland and Hosmer (2014) derived formulas for the conditional probabilities of the constrained continuation-ratio model as defined by \texttt{ocratio}. The formulas for the model in (2) are similar and given as

\[
\pi_c = \frac{e^{g_c(x)}}{1 + e^{g_c(x)}}
\]

\[
\pi_j = \frac{\gamma_{j+1} \times e^{g_j(x)}}{1 + e^{g_j(x)}} \quad j = 2, \ldots, c - 1
\]

and

\[
\pi_1 = 1 - \sum_{k=2}^{c} \pi_k
\]

where

\[
\gamma_j = 1 - \sum_{k=j}^{c} \pi_k \quad j = 3, \ldots, c
\]

2.3 The unconstrained continuation-ratio logistic regression model

The unconstrained continuation-ratio model is equal to the constrained model in section 2.2, except that we let the regression coefficients vary across the logits

\[
g_j(x) = \log \left\{ \frac{P(Y = j|x)}{P(Y < j|x)} \right\}
\]

\[
= \alpha_j + \beta_j^T x \quad j = 2, \ldots, c
\]

where \( \beta_j = (\beta_{j1}, \beta_{j2}, \ldots, \beta_{jp})' \) for \( j = 2, \ldots, c \). We now have \( c - 1 \) coefficients or ORs to describe the effect of an independent variable on the response. For independent variable \( x_k \), we have the following:

\[
\exp(\beta_{2k}) = \text{OR}(2,1)
\]

\[
\exp(\beta_{3k}) = \text{OR}(3,1 \ldots 2)
\]

\[
\vdots
\]

\[
\exp(\beta_{ck}) = \text{OR}(c,1 \ldots c - 1)
\]
The number of parameters in this model is the same as the number of parameters in the multinomial logistic regression model. The unconstrained continuation-ratio model may be fit using \( c - 1 \) binary logistic regression models, where the binary response variables are defined as

\[
Y_j^* = \begin{cases} 
1 & \text{if } Y = j \\
0 & \text{if } Y < j \\
\text{missing} & \text{if } Y > j
\end{cases}
\]

for \( j = 2, \ldots, c \).

The expressions for the conditional probabilities of the unconstrained continuation-ratio model are equal to those for the constrained continuation-ratio model in section 2.2 for given values of the logits \( g_2(x), \ldots, g_c(x) \); however, the evaluations of \( g_j(x_i) \) for a particular observation \( i \) are different for the two models.

3 The estimation commands

3.1 adjcatlogit (adjacent-category logistic regression)

Syntax

\[ \text{adjcatlogit } \text{depvar} [ \text{indepvars} ] [ \text{if} ] [ \text{in} ] [ , \text{level(#)} \text{ or listconstraints} ] \]

\text{indepvars} may contain factor variables.

Syntax for predict

\[ \text{predict } \{ \text{newvarname } | \text{newvarlist} \} [ \text{if} ] [ \text{in} ] [ , \text{pr } \text{xb outcome(outcome)} ] \]

Description

\text{adjcatlogit} fits adjacent-category logistic regression models of ordinal variable \text{depvar} on the independent variables \text{indepvars}. The actual values taken on by the dependent variable are irrelevant, except that larger values are assumed to correspond to “higher” outcomes.

Options for adjcatlogit

\text{level(#)} specifies the confidence level, as a percentage, for the confidence interval (CI). The default is \text{level(95)} or as set by \text{set level}. 
or reports the estimated coefficients transformed to ORs, that is, \( \exp(\beta) \) rather than \( \beta \).
Standard errors and CIs are similarly transformed. This option affects how results are displayed, not how they are estimated. or may be specified at estimation or when replaying previously estimated results.

listconstraints requests that a list of the constraints used by mlogit to fit the model be displayed.

**Options for predict**

pr calculates the predicted probabilities. This is the default. If you do not also specify the outcome() option, you specify one or c new variables, where c is the number of categories of the dependent variable. If you specify one new variable (and no outcome() option), outcome(#1) is assumed. If you specify the outcome() option, you must specify one new variable.

xb calculates the linear prediction. You specify one new variable (and no outcome() option). The contributions of the estimated constants are ignored in the calculations.

outcome(outcome) specifies the outcome for which the predicted probabilities are to be calculated. outcome() should contain either one value of the dependent variable or one of #1, #2, ... with #1 meaning the first category of the dependent variable, #2 meaning the second category, etc.

**Remarks**

adjcatlogit fits the adjacent category model using constrained multinomial logistic regression (mlogit), where the lowest category of the dependent variable is used as the reference category. The constraint free command is used to select free (unused) constraint numbers. The constraints are not dropped after estimation, so the number of free constraints is reduced each time an adjacent category model is estimated. The maximum number of constraints in Stata is 1,999, which is sufficient for a large number of estimations with adjcatlogit. If there are not enough free constraints, adjcatlogit will exit and give the error no free constraints. If that happens, constraint drop can be used to increase the number of free constraints.
Stored results

`adjcatlogit` stores the following in `e()`:

Scalars
- `e(N)` number of observations
- `e(k_cat)` number of categories
- `e(k_exp)` number of auxiliary parameters
- `e(df_m)` model degrees of freedom
- `e(df_0)` degrees of freedom, constant-only model
- `e(r2_p)` pseudo-$R$-squared
- `e(ll)` log likelihood
- `e(ll_0)` log likelihood, constant-only model
- `e(chi2)` $\chi^2$
- `e(p)` significance

Macros
- `e(cmd)` `adjcatlogit`
- `e(cmdline)` command as typed
- `e(constraints)` list of constraints
- `e(depvar)` name of dependent variable
- `e(title)` title in estimation output
- `e(chi2type)` Wald or LR; type of model chi-squared test
- `e(properties)` `b V`
- `e(predict)` program used to implement `predict`

Matrices
- `e(b)` coefficient vector
- `e(cat)` category values
- `e(V)` variance–covariance matrix of the estimators

Functions
- `e(sample)` marks estimation sample

3.2 `ccrlogit` (constrained continuation-ratio logistic regression)

Syntax

```
ccrlogit depvar [ indepvars ] [ if ] [ in ] [, level(#) or ]
```

`indepvars` may contain factor variables.

Syntax for predict

```
predict {newvarname | newvarlist} [ if ] [ in ] [, pr xb outcome(outcome) ]
```

Description

`ccrlogit` fits constrained continuation-ratio logistic regression models of ordinal variable `depvar` on the independent variables `indepvars`. The actual values taken on by the dependent variable are irrelevant, except that larger values are assumed to correspond to “higher” outcomes.
Options for `ccrlogit`

`level(#)` specifies the confidence level, as a percentage, for the CI. The default is `level(95)` or as set by `set level`.

`or` reports the estimated coefficients transformed to ORs, that is, \( \exp(\beta) \) rather than \( \beta \). Standard errors and CIs are similarly transformed. This option affects how results are displayed, not how they are estimated. `or` may be specified at estimation or when replaying previously estimated results.

Options for `predict`

`pr` calculates the predicted probabilities. This is the default. If you do not also specify the `outcome()` option, you specify one or `c` new variables, where `c` is the number of categories of the dependent variable. If you specify one new variable (and no `outcome()` option), `outcome(#1)` is assumed. If you specify the `outcome()` option, you must specify one new variable.

`xb` calculates the linear prediction. You specify one new variable (and no `outcome()` option). The contributions of the estimated constants are ignored in the calculations.

`outcome(outcome)` specifies the outcome for which the predicted probabilities are to be calculated. `outcome()` should contain either one value of the dependent variable or one of `#1, #2, ...` with `#1` meaning the first category of the dependent variable, `#2` meaning the second category, etc.

Remarks

`ccrlogit` fits the constrained continuation-ratio model using a generalized linear model (`glm`).
Stored results

ccrlogit stores the following in e():

Scalars
- e(N) number of observations
- e(k_cat) number of categories
- e(k_exp) number of auxiliary parameters
- e(df_m) model degrees of freedom
- e(df_0) degrees of freedom, constant-only model
- e(r2_p) pseudo-$R^2$
- e(ll) log likelihood
- e(ll_0) log likelihood, constant-only model
- e(chi2) $\chi^2$
- e(p) significance

Macros
- e(cmd) ccrlogit
- e(cmdline) command as typed
- e(depvar) name of dependent variable
- e(title) title in estimation output
- e(chi2type) Wald or LR; type of model chi-squared test
- e(properties) b V
- e(predict) program used to implement predict

Matrices
- e(b) coefficient vector
- e(cat) category values
- e(V) variance–covariance matrix of the estimators

Functions
- e(sample) marks estimation sample

3.3 ucrlogit (unconstrained continuation-ratio logistic regression)

Syntax

ucrlogit depvar [ indepvars ] [ if ] [ in ] [, level(#) or ]

indepvars may contain factor variables.

Syntax for predict

predict { newvarname | newvarlist } [ if ] [ in ] [, pr xb outcome(outcome) ]

Description

ucrlogit fits unconstrained continuation-ratio logistic regression models of ordinal variable depvar on the independent variables indepvars. The actual values taken on by the dependent variable are irrelevant, except that larger values are assumed to correspond to “higher” outcomes.
Options for ucrlogit

`level(#)` specifies the confidence level, as a percentage, for the CI. The default is `level(95)` or as set by `set level`.

`or` reports the estimated coefficients transformed to ORs, that is, $\exp(\beta)$ rather than $\beta$. Standard errors and CIs are similarly transformed. This option affects how results are displayed, not how they are estimated. `or` may be specified at estimation or when replaying previously estimated results.

Options for predict

`pr` calculates the predicted probabilities. This is the default. If you do not also specify the `outcome()` option, you specify one or $c$ new variables, where $c$ is the number of categories of the dependent variable. If you specify one new variable (and no `outcome()` option), `outcome(#1)` is assumed. If you specify the `outcome()` option, you must specify one new variable.

`xb` calculates the linear predictions. If you do not also specify the `outcome()` option, you specify one or $c$ new variables, where $c$ is the number of categories of the dependent variable. If you specify one new variable (and no `outcome()` option), `outcome(#1)` is assumed. If you specify the `outcome()` option, you must specify one new variable.

`outcome(outcome)` specifies the outcome or logit for which the predicted probabilities or linear predictions, respectively, are to be calculated. `outcome()` should contain either one value of the dependent variable or one of #1, #2, ... with #1 meaning the first category of the dependent variable or the first logit, #2 meaning the second category or the second logit, etc.

Remarks

`ucrlogit` fits the unconstrained continuation-ratio model using $c - 1$ binary logistic regression models (`logit`), where $c$ is the number of categories of the dependent variable.
Stored results

ucrlogit stores the following in e():

Scalars
- \( e(N) \): number of observations
- \( e(k_{cat}) \): number of categories
- \( e(k_{exp}) \): number of auxiliary parameters
- \( e(df_m) \): model degrees of freedom
- \( e(df_0) \): degrees of freedom, constant-only model
- \( e(r_2,p) \): pseudo-\( R^2 \)-squared
- \( e(ll) \): log likelihood
- \( e(ll_0) \): log likelihood, constant-only model
- \( e(chi2) \): \( \chi^2 \)
- \( e(p) \): significance

Macros
- \( e(cmd) \): ucrlogit
- \( e(cmdline) \): command as typed
- \( e(depvar) \): name of dependent variable
- \( e(title) \): title in estimation output
- \( e(chi2type) \): Wald or LR; type of model chi-squared test
- \( e(properties) \): b V
- \( e(predict) \): program used to implement predict

Matrices
- \( e(b) \): coefficient vector
- \( e(cat) \): category values
- \( e(V) \): variance-covariance matrix of the estimators

Functions
- \( e(sample) \): marks estimation sample

4 Examples

Here we use the well-known low birthweight (lbw.dta) dataset accessible in Stata through the `webuse` command.

```
. webuse lbw
(Hosmer & Lemeshow data)
```

This dataset is described in Hosmer, Lemeshow, and Sturdivant (2013, 24). In short, the dataset contains the birthweight and selected risk factors of low birthweight (birthweight less than 2,500 grams) of 189 children–mother pairs. The continuous variable \( bwt \) contains the birthweight measured in grams. We form an ordinal variable \( bwt4 = 1, 2, 3, 4 \), such that higher values of \( bwt4 \) represent lower birthweight.
Fitting ordinal logistic regression models

. generate bwt4 = .
. replace bwt4 = 1 if bwt > 3500
. replace bwt4 = 2 if bwt <= 3500 & bwt > 3000
. replace bwt4 = 3 if bwt <= 3000 & bwt > 2500
. replace bwt4 = 4 if bwt <= 2500
. tabulate bwt4

<table>
<thead>
<tr>
<th>bwt4</th>
<th>Freq.</th>
<th>Percent</th>
<th>Cum.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>46</td>
<td>24.34</td>
<td>24.34</td>
</tr>
<tr>
<td>2</td>
<td>46</td>
<td>24.34</td>
<td>48.68</td>
</tr>
<tr>
<td>3</td>
<td>38</td>
<td>20.11</td>
<td>68.78</td>
</tr>
<tr>
<td>4</td>
<td>59</td>
<td>31.22</td>
<td>100.00</td>
</tr>
<tr>
<td>Total</td>
<td>189</td>
<td>100.00</td>
<td></td>
</tr>
</tbody>
</table>

It may seem counterintuitive to define the categories of `bwt4` in the opposite direction of the underlying continuous variable. We choose this coding so that higher category values indicate more unfavorable outcomes, which is consistent with the usual way such variables are coded; for example, level of pain: 1 = none, 2 = mild, 3 = moderate, and 4 = severe. It is also consistent with how dichotomous variables are usually coded; for example, 0 = no disease, 1 = disease, 0 = not exposed, and 1 = exposed.

4.1 Examples using adjcatlogit

We start by fitting an adjacent-category logistic regression model of `bwt4` on `smoke` (smoking status during pregnancy: 0 = no, 1 = yes); `race` (1 = white, 2 = black, 3 = other); `lwt` (mother’s weight in pounds at last menstrual period); `ht` (history of hypertension: 0 = no, 1 = yes); and `ui` (presence of uterine irritability: 0 = no, 1 = yes). The option `or` requests that ORs rather than coefficients be displayed.
Smoking during pregnancy, the races “black” or “other”, hypertension, and presence of uterine irritability all indicate increased risk of higher values of \( bwt4 \) and thus of lower birthweight. A high value of mother’s weight, on the other hand, reduces this risk, although the estimated OR for lwt is quite small. The estimated effect of smoke on \( bwt4 \) may be interpreted as

\[
\hat{\text{OR}}(2, 1) = \hat{\text{OR}}(3, 2) = \hat{\text{OR}}(4, 3) = \exp(\hat{\beta}_k) = 1.74 \ (95\% \text{ CI } [1.25, 2.40])
\]

with similar interpretations for the other estimated effects.

As explained in sections 2.1 and 3.1, \texttt{adjcatlogit} uses constrained multinomial logistic regression (\texttt{mlogit}) to fit the adjacent category model. A list of the constraints is obtained by specifying the \texttt{listconstraints} option, as follows:

\[
\texttt{. adjcatlogit bwt4 smoke i.race lwt ht ui, or listconstraints}
\]

Constraints used with \texttt{mlogit}:


(output omitted)
Fitting ordinal logistic regression models

Factor variables may be specified in the usual manner.

```
. adjcatlogit bwt4 smoke##race lwt ht ui, or
```

Adjacent-category logistic regression

```
Number of obs = 189
LR chi2( 8) = 49.72
Prob > chi2 = 0.0000
Log likelihood = -234.79047 Pseudo R2 = 0.0958
```

|       | Odds Ratio | Std. Err. | z     | P>|z|    | [95% Conf. Interval] |
|-------|------------|-----------|-------|--------|---------------------|
| bwt4  |            |           |       |        |                     |
| smoke |            |           |       |        |                     |
| smoker|            |           |       |        |                     |
|       | 2.166554   | .4724645 | 3.55  | 0.000  | 1.413014 3.321946   |
| race  |            |           |       |        |                     |
| black |            |           |       |        |                     |
|       | 2.476267   | .7381052 | 3.04  | 0.002  | 1.380633 4.441366   |
| other |            |           |       |        |                     |
|       | 1.968733   | .4246554 | 3.14  | 0.002  | 1.289979 3.004629   |
| smoke##race |   |           |       |        |                     |
| smoker##black | | |       |        |                     |
|       | .9131809   | .44709   | -0.19 | 0.853  | .3497934 2.383977   |
| smoker##other | | |       |        |                     |
|       | .4471343   | .1705224 | -2.11 | 0.035  | .2117471 .9441881   |
| lwt   |            |           |       |        |                     |
|       | .9923672   | .0027256 | -2.79 | 0.005  | .9870394 .9977237   |
| ht    |            |           |       |        |                     |
|       | 1.771864   | .5593936 | 1.81  | 0.070  | .9543298 3.297747   |
| ui    |            |           |       |        |                     |
|       | 1.792218   | .3952526 | 2.65  | 0.008  | 1.163238 2.761299   |
| _anc  |            |           |       |        |                     |
| cons1 |            |           |       |        |                     |
|       | 1.647005   | .7491717 | 1.10  | 0.273  | .6753203 4.0168    |
| cons2 |            |           |       |        |                     |
|       | 1.068094   | .7350177 | 0.10  | 0.924  | .2772352 4.115009   |
| cons3 |            |           |       |        |                     |
|       | 1.633601   | 1.409166 | 0.57  | 0.569  | .3012234 8.859377   |

Following estimation, we can test for the overall effect of the interaction between `smoke` and `race`.

```
. test 1.smoke#2.race = 1.smoke#3.race = 0
```

( 1) [bwt4]1.smoke#2.race - [bwt4]1.smoke#3.race = 0
( 2) [bwt4]1.smoke#2.race = 0

```
chi2( 2) = 4.57
Prob > chi2 = 0.1020
```
The conditional probabilities of each outcome category can be obtained by predict in the usual manner, as follows:

```
predict p1-p4
```

Likewise for the linear prediction, one can type

```
predict xbeta, xb
```

### 4.2 Example using ccrlogit

We now fit a constrained continuation-ratio model using the same variables as in section 4.1.

```
crlogit bwt4 smoke i.race lwt ht ui, or
```

```
Constrained continuation-ratio logistic regression

Number of obs = 189
LR chi2( 6) = 44.22
Prob > chi2 = 0.0000
Log likelihood = -237.54456 Pseudo R2 = 0.0851

bwt4|  Odds Ratio   Std. Err.     z  P>|z|   [95% Conf. Interval]
-----|-------------|--------------|-----|------|------------------
bwt4|           |
smoke| 2.286725   .5662308     3.34 0.001  1.407481    3.715228
race |           |
  black| 4.081572   1.45312     3.95 0.000  2.031342    8.201096
  other| 2.086064   .5470363     2.80 0.005  1.247771    3.487718
lwt  | .9883587   .0040084   -2.89 0.004   .980535   .9962464
ht   | 2.411021   1.201645     1.77 0.077   .9077464   6.403789
ui   | 2.39648    .7904512     2.65 0.008   1.255495   4.573486

_cons |
  cons1| 2.423381   1.495945     1.43 0.152   .7227326   8.125792
  cons2| .7897336   .4673476    -0.40 0.690   .247604   2.518858
  cons3| .7136462   .4102232    -0.59 0.557   .2313073   2.201794

_cons
```

The estimated effect of `smoke` in this case is

$$\hat{OR}(2, 1) = \hat{OR}(3, 1 \ldots 2) = \hat{OR}(4, 1 \ldots 3) = \exp(\hat{\beta}_k) = 2.29 \ (95\% \ CI [1.41, 3.72])$$

which means that the odds of smokers are estimated to be 2.29 times that of nonsmokers for higher values of `bwt4`, that is, lower birthweight. The estimated OR is larger than that of the adjacent category model ($\hat{OR} = 1.74$) because the comparisons for the constrained continuation-ratio model include several response categories, whereas the adjacent category model compares only adjacent response categories.
### 4.3 Example using ucrlogit

The unconstrained continuation-ratio model is similar to the multinomial logistic model in that the effect of each independent variable on the response is described by \( c - 1 \) parameters.

```stata
. ucrlogit bwt4 smoke i.race lwt ht ui, or
Unconstrained continuation-ratio logistic regression
Number of obs = 189
LR chi2(18) = 56.75
Prob > chi2 = 0.0000
Log likelihood = -231.27942 Pseudo R2 = 0.1093

| bwt4 | Odds Ratio | Std. Err. | z    | P>|z| | 95% Conf. Interval |
|------|------------|-----------|------|------|-------------------|
| 2    |            |           |      |      |                   |
| smoke| 2.40644    | 1.243078  | 1.70 | 0.089| 0.8743305 6.623299|
| race | 7.091104   | 6.725815  | 2.07 | 0.039| 1.104997 45.50581|
| black| 3.540889   | 1.810036  | 2.47 | 0.013| 1.300152 9.643407|
| other| 1.001661   | 0.0083208 | 0.20 | 0.842| 0.9854846 1.018103|
| lwt  | 2.8945     | 2.681902  | 1.14 | 0.255| 0.662888 17.84375|
| ht   | 8.01311    | 5.096874  | 1.00 | 0.317| 0.0163281 45.50581|
| ui   | 2.8845     | 2.681902  | 1.14 | 0.255| 0.662888 17.84375|
| _cons| 0.3330399  | 0.401554  | -0.91| 0.362| 0.0313459 3.538435|

| 3    |            |           |      |      |                   |
| smoke| 1.564183   | 0.990885  | 1.00 | 0.317| 0.6514121 3.755948|
| race | 3.602625   | 2.276972  | 2.03 | 0.043| 1.043838 12.43383|
| black| 0.9405772  | 0.4551569 | -0.13| 0.899| 0.3643283 2.428295|
| other| 0.983936   | 0.007965  | -2.02| 0.044| 0.9686581 0.995361|
| lwt  | 1.114572   | 1.325293  | 0.09 | 0.927| 0.1083873 11.46141|
| ht   | 2.314626   | 1.597222  | 1.40 | 0.161| 0.7150425 7.422551|
| ui   | 2.246423   | 2.51009   | 0.72 | 0.469| 0.2514103 20.07244|
| _cons| 2.246423   | 2.51009   | 0.72 | 0.469| 0.2514103 20.07244|

| 4    |            |           |      |      |                   |
| smoke| 2.817403   | 1.105908  | 2.64 | 0.008| 1.305356 6.080917|
| race | 3.758631   | 1.959795  | 2.54 | 0.011| 1.352705 10.44375|
| black| 2.526023   | 1.087054  | 2.15 | 0.031| 1.08675 5.871446|
| other| 0.9834361  | 0.0066887 | -2.46| 0.014| 0.9704134 0.9966336|
| lwt  | 1.490237   | 4.483259  | 2.71 | 0.007| 1.676009 25.13302|
| ht   | 2.471801   | 1.106213  | 2.02 | 0.043| 1.028189 5.942297|
| ui   | 1.054066   | 0.9884219 | 0.06 | 0.955| 0.1677556 6.623063|
```

Logit 2 compares bwt4==2 with bwt4 < 2
Logit 3 compares bwt4==3 with bwt4 < 3
Logit 4 compares bwt4==4 with bwt4 < 4
The estimated effects of smoke on bwt4 are now

\[ \hat{OR}(2, 1) = \exp(\hat{\beta}_{2k}) = 2.41 \text{ (95\% CI [0.87, 6.62])} \]
\[ \hat{OR}(3, 1 \ldots 2) = \exp(\hat{\beta}_{3k}) = 1.56 \text{ (95\% CI [0.65, 3.76])} \]
\[ \hat{OR}(4, 1 \ldots 3) = \exp(\hat{\beta}_{4k}) = 2.82 \text{ (95\% CI [1.31, 6.08])} \]

As with the adjacent category and constrained continuation-ratio models, the odds of higher values of bwt4 (lower birthweight) are estimated to be higher for smokers than for nonsmokers. Although the confidence intervals of the three ORs for smoke are wide and largely overlapping, the point estimates are quite different, as are those for the other independent variables, which supports the idea that the effects are dependent on the category. It may seem that the constrained continuation-ratio model in section 4.2, where the effects are constrained across the logits, does not fit the data well.

Note that the results for logit 4 are identical to the results obtained with a binary logistic regression model using a dichotomized dependent variable defined as 0 = birthweight > 2500 grams, 1 = birthweight \( \leq \) 2500 grams.

The linear predictions obtained by predict with the option xb after estimation with ucrlogit are specific for each logit. Thus you may request either the linear prediction for one logit using the outcome option,

```
   . predict xb4, xb outcome(4)
```

or the linear predictions for all logits in one command,

```
   . predict xb1-xb4, xb
```

The linear prediction for logit 1 (the reference category) is 0.

### 5 Concluding remarks

I have presented the estimation commands adjcatlogit, ccrlogit, and ucrlogit, which calculate three ordinal logistic regression models: the adjacent category, the constrained continuation-ratio, and the unconstrained continuation-ratio models, respectively. The models can be used as alternatives to the proportional odds model (ologit), for instance, when the proportional odds assumption does not hold or when the comparisons between response categories for these models are more informative for the problem at hand. The continuation-ratio models are particularly useful for the analysis of sequential processes (Agresti 2010, 96–97), where \( \hat{Y} \) measures the number of attempts to attain a binary outcome.

Models estimated with adjcatlogit, ccrlogit, and ucrlogit are all equal to the binary logistic regression model (logit or logistic) if applied to a binary dependent variable.
Further model-building options for ordered response data are provided by the `gologit2` command (Williams 2006). `gologit2` fits generalized ordered logistic models that include an unconstrained model with the same number of parameters as the multinomial and unconstrained continuation-ratio models, the proportional odds model, and the partial-proportional odds model. The partial-proportional odds model allows for a subset of the regression coefficients to be constrained across the logits, thus providing a compromise between the restrictive constrained models and the unconstrained models, which often estimate more parameters than necessary.

Wolfe (1998) previously published the command (`ocratio`) for the constrained continuation-ratio model. As discussed in section 2.2, the model implemented by `ocratio` is not equivalent to the model implemented by `ccrlogit`. The model formulation in `ccrlogit` is equal to the recommended version of the constrained continuation-ratio model in Hosmer, Lemeshow, and Sturdivant (2013).

### 6 References


### About the author

Morten W. Fagerland is a senior researcher in biostatistics at the Oslo Centre for Biostatistics and Epidemiology. His research interests include the application of statistical methods in medical research, analysis of categorical data and contingency tables, and comparisons of statistical methods using Monte Carlo simulations.
Collecting and organizing Stata graphs

Joseph D. Wolfe  Shawn Bauldry
University of Alabama at Birmingham  University of Alabama at Birmingham
Birmingham, AL  Birmingham, AL
jdwolfe@uab.edu  sbauldry@uab.edu

Abstract. Stata includes a powerful set of tools for constructing a wide array of graphs. Stata’s graphing capabilities are well suited for describing exploratory or preliminary data analyses as well as producing publication-quality graphics. Currently, however, Stata does not have a built-in suite of commands for constructing various types of files (for example, HTML, TeX, or RTF files) to display multiple graphs. Such files can be invaluable for organizing and facilitating the interpretation of the numerous graphs needed throughout an analysis or in the final stage of a project. In this article, we provide an overview of two commands, graphsto and graphout, designed to organize and process multiple graphs across various file types.

Keywords: gr0060, graphsto, graphout, graph, graph export, HTML, RTF, TeX

1 Introduction

For many Stata users, graphs play an essential role in all aspects of an analysis. Graphs are often used to explore univariate and bivariate distributions of data, to display postestimation results for diagnostics, to explore the implications of models, and to efficiently and elegantly display the results of an analysis. Stata provides a powerful set of tools for constructing a variety of graphs for all of these uses (Mitchell 2012a,b).

Stata, however, does not currently have a built-in suite of commands for managing and reviewing the many graphs that emerge at various phases of a research project. It would be helpful, for instance, to have one file that contains all the preliminary descriptive graphs (for example, univariate or bivariate distributions or matrix scatterplots) so that users can refer to a single location to easily access, compare, and interpret the graphs. Similarly, it would be helpful to maintain a file that includes a series of graphs displaying postestimation diagnostics (for example, residual plots) or collects all the graphs to be included in an article.

Stata allows users to save a graph as one file in various formats (for example, PNG, EPS, WMF, and PDF) via the graph export command. However, to review and compare graphs outside Stata, users must manually move saved graphs to an external word processor like Microsoft Word or create links to the graphs in programs like LATEX. When a user is interested in only a few graphs, this is not a difficult procedure. However, when users want to review more than a few graphs, the process can become cumbersome and tedious.

A few ad-hoc procedures for constructing files with multiple Stata graphs are currently available online. In addition, Lo Magno (2013) developed a set of procedures that...
Collecting and organizing Stata graphs

allows users to transfer graphs obtained in Stata directly into Word documents, but these procedures are limited to Word running on Windows-based systems. Finally, users can also write their own Stata code to produce markup language code (Gini and Pasquini 2006). None of these approaches, however, help users to collect multiple Stata graphs in varying document types.

In this article, we present the two commands that are capable of quickly organizing Stata graphs and saving them to a single file for viewing. The first command, `graphsto`, saves users’ graphs and marks them for `graphout`. After users have created all their graphs, they can use `graphout` to create a new HTML, RTF, or TeX file containing links to the graphs. When users open this file, the graphs will be displayed and numbered. In the remainder of this article, we provide an overview of these two commands along with examples illustrating their use.

2 Description and syntax

Together, `graphsto` and `graphout` allow users to quickly create an RTF, an HTML, or a TeX file that can display a variety of graph formats. For exploratory research, for example, users often create HTML files that display PNG graphs. To share graphs with colleagues, on the other hand, users tend to use PNG graphs to create an RTF file, which can then be converted to a PDF file to embed graphs. Note that `graphout` will create HTML files with PNG graph formats only; RTF files on a personal computer can display TIF, EMF, WMF, PNG, and EPS files. RTF files on Macs can create RTF files that can display EPS, PNG, and PDF graph formats. TeX files can display multiple graph formats depending on the TeX package used.

The `graphsto` command is a wrapper for the `graph export` command, which outputs graphs as a variety of file types, for example, PNG, EPS, and PDF (see [G-2] `graph export` for a full list). `graphsto` exports graphs as the output format specified by `.suffix` and collects the names of the graphs as specified by `graphname`. The user can also use `graphsto` to add a title or note. The syntax for `graphsto` is as follows:

```
graphsto graphname.suffix [, options]
```

Table 1 describes the optional arguments for `graphsto`.

<table>
<thead>
<tr>
<th>options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title(string)</td>
<td>add a title above graph</td>
</tr>
<tr>
<td>note(string)</td>
<td>add a note below graph</td>
</tr>
<tr>
<td>replace</td>
<td>overwrite an existing file</td>
</tr>
<tr>
<td>directory</td>
<td>list the graph names stored in the global macro</td>
</tr>
<tr>
<td>clear</td>
<td>clear the graph names stored in the global macro</td>
</tr>
</tbody>
</table>
Users can also clear stored graph names from memory by typing `graphsto clear`, and they can obtain a list of the stored graph names by typing `graphsto dir`.

The `graphout` command constructs an RTF, an HTML, or a TeX file (as specified by `.suffix`) that contains links to the graphs stored by `graphsto`. The new file will display the specified graphs as long as the links are not broken. To embed the graphs, the user should create a PDF copy of the file containing the links to the graphs. The syntax for `graphout` is as follows:

```
graphout using filename.suffix [, options]
```

where `filename` is the name of the file that will contain links to the graphs that have been stored with `graphsto`. Table 2 describes the options for `graphout`.

### Table 2. Options for `graphout`

<table>
<thead>
<tr>
<th>options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output (HTML, RTF, and TeX)</td>
<td></td>
</tr>
<tr>
<td><code>replace</code></td>
<td>overwrite an existing file</td>
</tr>
<tr>
<td><code>append</code></td>
<td>append the output to an existing file</td>
</tr>
<tr>
<td>Size (HTML and TeX only)</td>
<td></td>
</tr>
<tr>
<td><code>height(string)</code></td>
<td>adjust the height of graphs</td>
</tr>
<tr>
<td><code>width(string)</code></td>
<td>adjust the width of graphs</td>
</tr>
<tr>
<td><code>scale(numlist)</code></td>
<td>adjust the scale of graphs</td>
</tr>
<tr>
<td>Layout (RTF, HTML, and TeX)</td>
<td></td>
</tr>
<tr>
<td><code>alignment(string)</code></td>
<td>center, right-align, or left-align the graphs</td>
</tr>
<tr>
<td>Text (RTF, HTML, and TeX)</td>
<td></td>
</tr>
<tr>
<td><code>nocount</code></td>
<td>do not add “Figure #” to each graph</td>
</tr>
<tr>
<td><code>basecount(numlist)</code></td>
<td>specify the starting # for “Figure #”</td>
</tr>
<tr>
<td>Advanced (TeX only)</td>
<td></td>
</tr>
<tr>
<td><code>placement(string)</code></td>
<td>specify the placement of a float in TeX file</td>
</tr>
<tr>
<td><code>fbarrier(#)</code></td>
<td>add “\FloatBarrier” to every graph in TeX file</td>
</tr>
<tr>
<td><code>document</code></td>
<td>make a stand-alone TeX document</td>
</tr>
<tr>
<td><code>label(string)</code></td>
<td>add “\label{string}” to TeX files</td>
</tr>
</tbody>
</table>
3 Examples

3.1 Basic example

The following example illustrates using `graphsto` and `graphout` to create an RTF document containing one graph. For this example, we create a scatterplot of `price` and `mpg` using Stata’s built-in dataset `auto.dta`. We use `graphsto` to title the graph and export it to a PNG format. This command also stores the name of the graph, which we specified here as `Ex1`. Because we use `graphsto`, it is not necessary to specify the title of the graph in `graphout`.

```
    . graphsto, clear
    . sysuse auto
    . graph twoway scatter price mpg, scheme(sj)
    . graphsto Ex1.png, replace title(Price by MPG)
    . graphout using Ex1.rtf, replace
```

Executing this code produces an RTF file with the specified title and a link to the following graph.

![Graph Image]

For the links to the graphs in the RTF file (and in all the other files formats) to work, the graphs must remain in the same file directory as the RTF file. If a user moves the RTF file to a different directory without moving the graphs as well, then the links will fail and no graphs will appear. This is important to remember when working with collaborators. It is possible, however, to save a copy of the original RTF, HTML, or TeX file with the links to the graphs as a PDF file that contains all the graphs and does not require the original graph files to be in the same directory.
3.2 Collecting graphs for exploratory analysis

During the preliminary phase of a project, we often find ourselves constructing numerous graphs to explore the data. For this type of exploratory analysis, creating HTML files can be especially useful. The files can be opened in the user’s favorite web browser, and the user can easily review updates to the graphs by clicking on the refresh button. For the following example, we again use Stata’s `auto.dta`. We construct histograms by `price` and `mpg` and then a scatterplot by `price` and `mpg`. We also illustrate two formatting features that are options for `graphout`: centering the graphs on the page and resizing the graphs while maintaining the same scale. There are also options to adjust the height and width of the graphs directly.

```stata
.sgraph, clear
.svusue auto
.sgraph price, scheme(sj)
.sgraph Ex2_1.png, replace title(Histogram for price)
.sgraph mpg, scheme(sj)
.sgraph Ex2_2.png, replace title(Histogram for mpg)
.sgraph twoway scatter price mpg, scheme(sj)
.sgraph Ex2_3.png, replace title(Scatterplot price by mpg)
.graphout using Ex2.html, replace alignment(center) scale(0.5)
```

Executing this code produces an HTML file with links to the following graphs.
Collecting and organizing Stata graphs

Figure 1: Histogram for price

Figure 2: Histogram for mpg

Figure 3: Scatterplot price by mpg
3.3 Constructing graphs in loops

Here we illustrate a more advanced use of `graphsto` and `graphout`. Our third example involves quickly constructing numerous graphs using loops. This procedure can make exploratory or diagnostic analyses, for instance, much more efficient than examining the graphs one by one. For this example, we examine scatterplots that include linear fits for `price` and six different variables potentially associated with `price`. After we construct each graph, we use `graphsto` to export the graph along with a title. After the loop, we use `graphout` to save all the collected graphs in one HTML file with the graphs left-aligned and resized to the specified height. For HTML files, the `height()` and `width()` options are automatically specified by pixels. For TEX files, users can specify the units for height or width (for example, `2in` for 2 inches or `5cm` for 5 centimeters).

```
. graphsto, clear
. sysuse auto
. foreach var of varlist mpg rep78 headroom trunk weight length {
   2.   graph twoway scatter price `var' || lfit price `var', scheme(sj)
   3.   graphsto Ex3_`var'.png, replace title(Scatterplot price by `var')
   4. }
. graphout using Ex3.html, replace alignment(left) height(500)
```

3.4 Adding a graph to a table of regression estimates

Our fourth example involves appending a graph to a set of model estimates stored using Jann’s `eststo` and `esttab` suite of commands (Jann 2005, 2007). The append feature of `graphout` allows a user to add a graph (or set of graphs) to an already existing file. This feature introduces many interesting possibilities. For instance, a user can add a diagnostic graph to a TEX file that contains a table of regression estimates. The following example illustrates this using Stata’s `auto.dta`.

We estimate a regression model with the mile per gallon (`mpg`), repair record in 1978 (`rep78`), weight (`weight`), and length (`length`) predicting the price of a car, and we store the results using the `eststo` command. We are interested in examining the Studentized residuals, a common diagnostic for identifying potential outliers. Following our regression, we use `predict` to obtain the Studentized residuals, and we then create a histogram and box plot to examine the distribution. We combine the histogram and box plot into one graph with the two plots side by side. Finally, we use the `esttab` command to create a TEX file with the regression results and then use `graphout` to append the Studentized residual plots to the regression table.

```
. eststo
. sysuse auto
. foreach var of varlist mpg rep78 headroom trunk weight length {
   2.   eststo
   3.   est `var'
   4. }
. predict stdres, std
. graph twoway histogram stdres || boxplot stdres, scheme(sj)
. graphsto Ex3_`var'.png, replace
. esttab, replace
. graphout using Ex3.html, replace alignment(left) height(500)
```
Collecting and organizing Stata graphs

```
. graphsto, clear
. sysuse auto
. qui regress price mpg rep78 weight length
. eststo m1
. predict rstu, rstu
. tempfile g1 g2
. histogram rstu, scheme(sj) nodraw saving(`g1')
. graph box rstu, scheme(sj) nodraw saving(`g2')
. graph combine "`g1'" "`g2'", scheme(sj)
. graphsto Ex4.png, replace title(Studentized residuals)
. esttab m1 using Ex4.tex, replace b(%9.3f) se(%9.3f) nogap > compress title("Price regression") alignment(center)
. graphout using Ex4.tex, append scale(0.25)
```

Executing the above code produces a TeX file with code for the regression results and a link to the graph. The following figure illustrates the compiled TeX code for this example:
J. D. Wolfe and S. Bauldry

Table 1: Price regression

<table>
<thead>
<tr>
<th></th>
<th>Coefficient</th>
<th>Standard Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>price</td>
<td>-106.712</td>
<td>(81.158)</td>
</tr>
<tr>
<td>mpg</td>
<td>910.986**</td>
<td>(304.527)</td>
</tr>
<tr>
<td>rep78</td>
<td>4.960***</td>
<td>(1.120)</td>
</tr>
<tr>
<td>length</td>
<td>-115.018**</td>
<td>(38.565)</td>
</tr>
<tr>
<td>_cons</td>
<td>11934.507*</td>
<td>(5774.178)</td>
</tr>
</tbody>
</table>

N = 69

Standard errors in parentheses
*p < 0.05, ** p < 0.01, *** p < 0.001

Figure 1: Studentized residuals
4 Conclusion

Stata has excellent graphing capabilities for tasks ranging from describing exploratory analysis to producing publication-quality figures. The commands we describe in this article, `graphsto` and `graphout`, are designed to help users collect, organize, and present multiple Stata graphs. The commands allow users to construct or append graphs to three different file types (HTML, RTF, and TeX). The examples illustrate some of the benefits of this capability to manage and view multiple Stata graphs in one file.

5 Acknowledgments

The `graphout` and `graphsto` programs are modeled on and meant to complement Ben Jann’s `eststo` and `esttab` packages (Jann 2005, 2007). We are also grateful for Eng’s article on file filters (Eng 2007). The idea for `graphout` came from an email on Statalist written by Maartin Buis. His email explained how to add graphs to an HTML file and inspired the `graphout` and `graphsto` programs.

6 References


About the authors

Joseph D. Wolfe is an assistant professor in the department of sociology at the University of Alabama at Birmingham.

Shawn Bauldry is an assistant professor in the department of sociology at the University of Alabama at Birmingham. His methodological research interests focus on the development of structural equation models and approaches to handling missing data.
Speaking Stata: Design plots for graphical summary of a response given factors

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

Abstract. Design plots, as defined in this article, show summaries of a response variable given the classes or distinct levels of numeric or string variables presented as influencing factors. Any summarize results can be plotted using statsby as an engine to produce summaries for groups of observations defined by classes and their cross-combinations. graph dot is used by default, but graphs may readily be recast using graph hbar or graph bar. Such plots offer scope for detailed yet concise data exploration and reporting.

Keywords: gr0061, designplot, design plots, graphics, grmeanby, statsby, summarize

1 Introduction

In this article, I introduce and explain a new command, designplot, that produces a graphical summary of a numeric response variable given one or more factors. The term “factor” in this context means that any (numeric or string) variable concerned will be treated in terms of its distinct classes or levels as they occur in the data. Use of Stata’s factor-variable syntax is neither explicit nor implicit.

The focus is, therefore, on a Stata program for a particular kind of graph. The choice of a name for such graphs was, in a sense, backward. A program was written to produce graphs that otherwise would be difficult to produce except through several intricate commands. A Stata program always requires a distinct name. That name may be arbitrary within a few syntactic rules, but it is natural to prefer a name that is memorable, even catchy. I borrowed a name that is used in statistical literature for a graph that looks quite different in practice, but in principle shows statistical summaries of the same kind.

Design plots (as here defined) offer a diversity of uses, ranging from simple exploratory overviews to multiscale breakdowns deserving detailed scrutiny. In this article, I discuss what is possible with the new command and relate the ideas behind designplot to previous literature.
2 Examples

To begin, we consider the example in figure 1, produced by

```
.sysuse auto
.designplot mpg foreign rep78
```

![Design plot showing mean miles per gallon (mpg) of all cars and for distinct levels of origin (foreign or domestic) and repair record, singly and jointly]

This command produces a plot showing the mean of \texttt{mpg} for all observations in Stata’s \texttt{auto.dta}; for the classes defined by the values of \texttt{foreign} and also the classes of \texttt{rep78}; and for the classes defined by the cross-combinations of values of \texttt{foreign} and \texttt{rep78} occurring in the data.

The graph shows several features of the data. There are indications that mean \texttt{mpg} differs according to whether cars are foreign (from outside the United States) and according to their repair record. There is a hint that the relation between \texttt{mpg} and repair record, an ordered scale, may not be simple. As usual, appearances such as these may be side effects of variations in other predictors not shown.

Because the plot also includes results from cross-combinations of the two categorical variables, we get other clues as to what is occurring. We see that there are no foreign cars with repair record 1 or 2; like almost any other Stata command, \texttt{designplot} cannot show results for data that do not exist. Furthermore, whereas foreign cars have higher \texttt{mpg} than domestic, given repair record 3 or 4, the reverse is true for cars with repair record 5.

Hence, the plot provides some detail on how a response varies with predictors, presented in terms of their distinct levels. \texttt{designplot} takes whatever is offered as pre-
dictors, including string variables as well as numeric. There is no error in presenting a predictor with many distinct levels, but the plot is unlikely to be helpful.

As demonstrated in the first example, `designplot` by default shows means for all the data and for whatever detailed breakdowns (one-way, two-way, and so forth) are possible given the predictors specified. Options give scope for showing other summary statistics as calculated by `summarize` (see `[R] summarize`) and for restricting the results shown in the plot.

By default, the graph is produced by `graph dot` (see `[G-2] graph dot`). Optionally, `graph hbar` (see `[G-2] graph hbar`) or `graph bar` (see `[G-2] graph bar`) may be used instead. Also by default, a `ytitle()` appears at the bottom of the graph when a single summary statistic is shown; if two or more statistics are shown, a legend appears instead. A description of the response variable being shown appears as `t1title()` at the top of the graph, again by default.

Let’s look at a different example. The ship *R.M.S. Titanic* sank in the North Atlantic in 1912 with much loss of life. The disaster continues to receive attention in many styles, from books and movies to the statistical approach central here. On the last front, we make no attempt to survey contributions beyond noting the pioneer graphical work of Bron (1912), which seems little known within statistical science.

Dawson (1995) gives an accessible dataset on the fate of those on board (note the small qualifications in his article about the accuracy of the data). Here we read Dawson’s version of the data into Stata, specifying whether individuals survived together with various possible predictors. The variable names differ slightly from Dawson’s because we follow a convention that (0,1) indicator or dummy variables should be named for whatever is coded 1. The mean of *survived* is precisely the response of interest as the fraction or proportion surviving. Figure 2 is a first version of our design plot.

```stata
. infix class 1-9 adult 10-18 male 19-27 survived 28-36 using \> http://www.amstat.org/publications/jse/datasets/titanic.dat.txt, clear (2201 observations read)
. label define class 0 crew 1 first 2 second 3 third
. label define adult 1 adult 0 child
. label define male 1 male 0 female
. label define survived 1 yes 0 no
. foreach v in class adult male survived {
  2. label values `v´ `v´
  3. }
. designplot survived class adult male, maxway(2) ysize(7)
> ylabel(0 .25 "25" .5 "50" .75 "75" 1 "100", angle(h)) ytitle(% survived)
> yscale(alt) t1title(""")
```
Figure 2. Design plot showing percent survived from the *Titanic* in relation to class, age, and gender

The option choices in this last `designplot` command not only improve the graph as compared with the defaults, but also show how you can additionally exploit the options of the underlying `graph` command (specifically here, `graph dot`):

1. The graph stops short of showing the three-way breakdown (with categories such as “male adults in first class”) by using the option `maxway(2)`. The graph still deserves greater height as compared with the default, obtained with `yscale(7)`.
2. Fraction or proportion is a natural scale. If you prefer to show percents on the $y$ axis, you need to change only the axis labels and the axis title.

3. Partly to show that it can be done, the $y$ axis is placed at the top of the graph in a manner common with tables but less conventional with graphs. Its title is made more informative, as \textit{\% survived} rather than \textit{mean}. The \texttt{ytitle()} should, therefore, be suppressed. See Cox (2012a) if desired for more discussion on axis conventions and choices.

The graphics syntax here shows a small clash of conventions. The option \texttt{ysize()} for controlling the vertical size of the graph echoes the usual convention that the $y$ axis is vertical. The options \texttt{ylabel()} and \texttt{ytitle()} echo a convention peculiar to \texttt{graph dot} and its siblings \texttt{graph bar} and \texttt{graph hbar}: the $y$ axis is the axis showing numeric summaries, regardless of its orientation. This convention is adopted to ease experimentation. In particular, the single change from \texttt{bar} to \texttt{hbar}, or vice versa, is sufficient to move between one command and another without making any changes to options. Less well known is that \texttt{graph dot} has an undocumented \texttt{vertical} option.

In problems like this, many researchers would prefer a bar chart. \texttt{designplot} has a special option to make this easier. The \texttt{recast()} option is inspired by the option of the same name for \texttt{twoway}. Either option recasts a \texttt{graph} command to an equivalent. In \texttt{designplot}, you can recast from \texttt{graph dot} to \texttt{graph hbar} or \texttt{graph bar}. \texttt{hbar} is far more useful because the categorical axis labeling of \texttt{graph dot} rarely works well if transposed to vertical. Note that \texttt{recast()} here will not recast your graph to any \texttt{twoway} type; as said, the name is inspired by a \texttt{twoway} option, but it is not the same option.

If we \texttt{recast(hbar)}, we can add whatever small flourishes are permitted by \texttt{graph hbar}. Suppose we would like to show the percents as numeric labels at the top of each bar. For this, it is easiest to multiply the binary response by 100 first to change its mean to percent terms. We need a little more space to show such labels for those bars at or near 100\%. Figure 3 is the result.

\begin{verbatim}
. generate survived2 = 100 * survived
. designplot survived2 class adult male, maxway(2) ysize(7)
  > ylabel(0(25)100, angle(h)) ytitle(\% survived) yscale(alt) recast(hbar)
  > blabel(total, format(\%2.0f) size(medsmall)) yscale(r(0 110)) t1title(""")
\end{verbatim}
In this graph, the size of the numeric labels was determined by experiment. It is arguable that the axis labels and ticks are now redundant given the bar labels. In a moment, we shall see how to remove them.

Figure 3 exemplifies a simple strategy: to blur or even ignore a conventional distinction between graphs and tables (Cox 2008).

Focusing on percent survival is a good idea, but we still should keep track of how many people were in each category. The count or number of observations is one of several summaries available from `summarize`, so a bar chart of frequencies is easy to use.
within the same framework. Note that a response variable `yvar` must still be specified, even though it is not evident on the graph. In this graph, we omit axis labels and ticks. We also omit the count for all observations by using the `minway()` option. The small amount of extra space needed for the `y` axis was again determined by experiment. Figure 4 is the result.

```stata
. designplot survived class adult male, statistics(count) minway(1) maxway(2)
   > ysize(7) yscale(alt) recast(hbar) blabel(total, format(%2.0f) size(medsmall))
   > yscale(r(0 2300)) ylabel(, nolabels noticks) t1title(""

Figure 4. Design plot in the form of a bar chart, showing frequencies of people on the Titanic by class, age, gender, and two-way combinations of those categories.
```
3 Origins

designplot is based on an eclectic combination of ideas. Readers are warmly invited to inform the author of other similar or related work.

1. The existing Stata command `grmeanby` (see [R `grmeanby`]) shows means (or, optionally, medians) of a response variable given one or more other variables. The scope of `grmeanby` is identical to that of `designplot` insofar as the other variables could be string variables as well as numeric variables. As recorded by Gould (1993) and in the manual entry, `grmeanby` was inspired by examples in Chambers and Hastie (1992). `grmeanby` is based on direct use of `summarize`.

2. Freeny and Landwehr (1992) gave the name “design plot” to plots similar to those in Chambers and Hastie (1992), and that name is associated with software implementations outside Stata, notably in S, S-Plus, and R. The name is also consistent with the S syntax detailed by Chambers and Hastie (1992, 546–547). In these implementations, plots show results from fitting linear models, specifically analyses of variance. The name evokes the idea of an underlying experimental design, but the command here clearly may be applied to any data, including observational data in any sense of that term. The graph shown by Zuur, Ieno, and Smith (2007, 37) is an example from the applied literature. See also Crawley (2013) for more detail on a wider-ranging implementation in R.


Graphs of types 3 and 4 commonly show effects and residuals scaled to be comparable in terms of variability.

5. Graphically, these displays share a possible problem: points may need to be plotted close to each other, creating difficulties especially if any text labels occlude each other or need to be abbreviated. Three out of four examples in Chambers and Hastie (1992) show this, as does the example in [R `grmeanby`]. Several examples in Hoaglin, Mosteller, and Tukey (1991) avoid the problem only by jittering points apart. Harrell (2001) used a different display based on dot charts or dot plots (in the sense of Cleveland [1984, 1994]) that avoids this problem. Conversely, a dot chart representation will work well with, say, 10 entries, but not with 100 or more.
6. On a simpler level, tables or graphs reporting survey results often show two or more separate breakdowns of some sample. Examples are shown by Tufte (2001, 179) and (more trivially) Cox (2008), among many others.

7. The `statsby` command with its `subsets` option provides an easy framework for calculation and assembly of summary statistics for zero-, one-, two-way and higher breakdowns of a dataset. Cox (2010) illustrated its exploitation for graphics. More will be said later on how the term “way” is used with `designplot`.

The term “design plot” is adopted here as a simple, memorable name and given its earlier and widespread use to show similar information. These are positive features. On the other hand, the connotation of experimental design will often be inappropriate. The use of dot-chart (or, optionally, bar-chart) form also distinguishes the results of this command from others published as design plots. People who like the plots and dislike the name are free to use other terminology, or none at all. Not every kind of graph needs a distinct name, but clearly every graph program does.

This lack of standardization is not new:

“Most or all features of statistical computation—computer hardware, software systems, coding, languages, symbols, terminology, procedures— have much to gain from elimination of pointless variations, redundancies and confusion. Yet pointlessness is not always easy to judge. The only quite satisfying rule of standardization is that you adopt my standards.” (Anscombe 1981, 3)

To summarize in Stata terms: `designplot` is a generalization and recasting of `grmeanby`, using `summarize` to produce summaries, `statsby` to provide machinery for multiway breakdowns, and `graph dot` (or `graph bar` or `graph hbar`) to plot the graph rather than `twoway`.

4 Discussion

designplot creates a new dataset of `summarize` results that uses default variable names (`.stat1` and so forth) for each statistic and uses `.way`, `.group`, and `.entry` to describe the results. If the number of observations is not one of the statistics requested, a variable with default name `.nobs` is added anyway, on the grounds that it will often be interesting or useful. The original dataset will be restored after the graph is drawn, but the results set may be saved for other use with the `saveresults()` option.

We can now epitomize how `designplot` differs from what is readily available through (for example) `graph dot`. There are two main differences. First, `graph dot` and its siblings are more restricted in offering only one-way or two-way or three-way breakdowns given, respectively, one or two or three “factors” as arguments to `over()` or `by()` options. Second, they do not give scope for saving results for separate graphing or tabulation.
Similarly, `designplot` is more general than `grmeanby`, which allows means or medians and one-way breakdowns only.

Consider again the example of figure 1. This example produces a plot that displays the following:

1. the mean of `mpg` for all observations, which may be called a “zero-way” breakdown;
2. the means for all the classes defined by the values of `foreign` and also of `rep78`, which may be called “one-way” breakdowns, as is often done in statistical literature; and
3. the means for all the classes defined by the cross-combinations of values of `foreign` and `rep78` occurring in the data, which similarly may be called a “two-way” breakdown, again as is often done.

In general, specifying one or more factors gives scope for various breakdowns, but the number of (cross-)combinations may grow rapidly and the resulting graph might be too complicated to be readable or useful. Thus `designplot` also offers options to restrict the scope of what is plotted.

Missing values require a special note. `designplot` may be applied when users want to show summaries for missing values of the factors. The recommended approach, however, is to clone the variable concerned and use new codes to show missings explicitly. This is mainly because values of . or empty strings would not show up well on graphs. (Missings would be problematic otherwise, given their use by `statsby` to denote all the data.) The help for `designplot` includes a detailed example in which `rep78` for `auto.dta` is cloned and missings are recoded to 6, with value labels to make matters clear.

Some users may wish to add reference lines for (for example) the overall mean (or, optionally, median) in the style of `grmeanby`. This is easy with a prior calculation. The examples in the help include a typical sequence.

The extension likely to be of greatest interest is to move beyond predefined categorical variables that arrive as part of a dataset to intervals defined by the researcher, subdividing the range of counted or measured variables. There is no syntax in `designplot` for this because various methods might be useful. Typically, an extra line of code is required to create a new variable before `designplot` is called.

A method very popular in some quarters is to identify quantile-based bins that contain approximately equal frequencies. `xtile` (see [D] `pctile`) is the usual command of choice here. (Note that researchers are often disappointed by the failure of `xtile` to produce exactly equal frequencies. This is the case whenever the sample size is not a multiple of the number of groups desired, as when 42 can at best be divided into two groups of 10 and two of 11. But the major reason for unequal frequencies is the existence of tied values. Sometimes results better than those of `xtile` can be obtained by using a different inequality at bin boundaries or, equivalently, by binning a negated version of the variable. If this issue is interesting or important to your work, see the comments of Cox [2012b].)
An alternative that deserves greater use by comparison is just to define bins of equal width. On cosmetic grounds, we might have a preference for nice round numbers, where “nice” is a little hard to define but easy to recognize. The functions \texttt{floor()} and \texttt{ceil()} can crack such a problem (Cox 2003).

The capacity of \texttt{designplot} to show frequencies of various unions and intersections of classes or sets makes it an alternative to Venn diagrams. Venn diagrams are popular partly because people recall from early courses (say, in probability) how they make simple problems even simpler. Unfortunately, Venn diagrams in general are very hard to draw usefully. Edwards (2004) gives a definitive account. While he rightly explains how clever tricks make drawing arbitrarily complicated Venn diagrams possible at all, it is difficult to avoid concluding that the results are often too bizarre to be useful statistically.

A yet further possibility is that \texttt{designplot} could be applied to cope with multiple response variables. As with researcher-defined binning of counted or measured variables, coping with a different data structure can be delegated to a \texttt{reshape long} of the dataset so that several variables are stacked into one. Returning to \texttt{auto.dta}, we want to get a plot of skewness and kurtosis for all numeric variables. Figure 5 is the result.

```stata
. sysuse auto, clear  
   (1978 Automobile Data)  
. rename (price-foreign) (num=)  
. reshape long num, i(make) j(variable) string  
   (note: j = displacement foreign gear_ratio headroom length mpg price rep78 > trunk turn weight)  
   Data wide -> long  
   Number of obs. 74 -> 814  
   Number of variables 12 -> 3  
   j variable (11 values) -> variable  
   xij variables: numdisplacement numforeign ... numweight -> num  
. designplot num variable, statistics(skewness kurtosis) minway(1) > t1title(auto dataset) yline(0, lcolor(gs12) lwidth(vthin))  
   > yline(3, lcolor(gs12) lwidth(vthin) lpattern(dash))  
   > entryopts(sort(1) descending)
```
The ease with which the dataset can be restructured in just two lines is offered as grounds for not complicating the syntax, let alone the code, of `designplot`. We add two vertical reference lines. Gaussian (normal) distributions, often used as reference distributions even when we do not expect to observe them in practice, have skewness 0 and kurtosis 3. The skewness and kurtosis of a mix of variables with quite different units of measurement and magnitude would have no meaning; hence, the option calls `minway(1)`. So the interest is just in one group of results. `entryopts()` is a handle to pass options, here to sort the individual entries on the first “variable” plotted or the results for skewness.

5 The `designplot` command

5.1 Syntax

```
designplot yvar xvarlist [ if ] [ in ] [ weight ] [ , statistics(statistics) maxway(#) minway(#) saveresults(filename[, save_options]) prefix(prefix) recast(bar|hbar) {variablelabels|variablenames} alllabel(text) entryopts(over_subopts) groupopts(over_subopts) graph_options ]
```

`aweights` and `fweights` are allowed; see [U] 11.1.6 `weight`.

Figure 5. Design plot showing skewness and kurtosis of numeric variables in `auto.dta`
5.2 Options

**statistics(Statistics)** specifies statistics calculated by `summarize` to be calculated. The default is the mean (only). One or more statistics may be specified. Note that no allowance is made in graphics for different statistics being on different scales, so the user may need to exercise discretion over what is specified. The names allowed include the names of the r-class results as visible after `summarize`, `detail` or as documented in [R] `summarize`. Thus `p50` specifies the median available as `r(p50)`.

Allowed synonyms also include the following (any synonyms specified will be echoed to the `ytitle()` or legend):

1. **n** or **count** or any abbreviation of **frequency** for N.
2. **minimum** for **min** and **maximum** for **max**.
3. **total** for **sum**.
4. **median** for **p50**.
5. **SD** for **sd**.
6. Any abbreviation of **variance** or **Variance** for Var.
7. **skew** for **skewness** and **kurt** for kurtosis.

Note that if just `statistics(N)` is specified, the `year` specified is immaterial so long as it is nonmissing whenever values of `xvarlist` are nonmissing.

**maxway(#)** specifies the maximum “way” to be plotted. See the earlier explanation on breakdowns that are called zero-way, one-way, two-way, and so forth. Thus `maxway(1)` by itself specifies that only zero-way and one-way breakdowns be shown.

**minway(#)** specifies the minimum “way” to be plotted. See the earlier explanation on breakdowns that are called zero-way, one-way, two-way, and so forth. Thus `minway(1)` by itself specifies that the zero-way breakdown not be shown.

**saveresults(filename[ , save_options])** saves the results as a Stata dataset. Options of `save` may be specified, most usefully `replace`. The dataset will include notes on the `designplot` command issued and (if defined) the filename and its date for the (saved) dataset.

**prefix(prefix)** is an occasionally used option. `designplot` creates a dataset of results with variable names such as `stati` and so forth. If these names clash with existing variable names, this option may be used to add a prefix to all such names to remove the clash.

**recast(hbar | bar)** specifies that the graph be drawn using `graph hbar` or `graph bar`. The default is `graph dot`. People fond of bar charts are advised to try `graph hbar` for greater readability of axis information. Note for experienced users: although the
option name is suggested by another `recast()` option, this is not a back door to recasting to a `twoway` plot.

`variablelabels` specifies that one-way breakdowns be labeled by the corresponding variable labels or by the corresponding variable names if no variable label is defined. The default is, or should be, an invisible label (precisely, an instance of `char(160)`).

`variablenames` specifies that one-way breakdowns be labeled by the corresponding variable names. The default is, or should be, an invisible label (precisely, an instance of `char(160)`). The reason for using this option rather than `variablelabels` is likely to be that variable labels would take up too much space.

Only one of `variablelabels` and `variablenames` may be specified.

`alllabel(text)` specifies text to label results for all observations used. The default is `alllabel(all)`.

`entryopts(over_subopts)` specifies `over_subopts` of `graph dot`, `graph hbar`, or `graph bar`, used to tune the corresponding call to an `over()` option that affects the display of individual entries in the graph. Users unsure of what this means may find it helpful to inspect the source code or, alternatively, to just modify a graph using the Graph Editor. Useful examples are `entryopts(sort(1))` and `entryopts(sort(2) descending)`, where (1), (2), etc., indicate the first, second, etc., statistic specified.

`groupopts(over_subopts)` specifies `over_subopts` of `graph dot`, `graph hbar`, or `graph bar`, used to tune the corresponding call to an `over()` option that affects the display of groups of entries in the graph. Users unsure of what this means may find it helpful to inspect the source code or, alternatively, to just modify a graph using the Graph Editor.

`graph_options` are other options allowed with `graph dot`, `graph hbar`, or `graph bar`. Note that, among other defaults, `title()` is used to display information on `yvar`.

## 6 Conclusions

The design of design plots was the outcome of an irregular but repetitive personal path. Over the last 20 years or so—for example, in repeated readings of Harrell (2001)—I have often encountered graphs I liked that were loosely or even closely similar to those here. Over that period, `grmeanby` was available as a Stata command offering one solution, but the need was for something more general.

`designplot` is offered with a suggested variety of uses. It builds on versatile commands: `summarize`, `statsby`, and `graph dot` and its siblings. The way they come together is distinctive. `designplot` could be useful in exploration, even if its graphs are never made public, and in reporting, either for one response variable or for several.
7 Acknowledgments

Elizabeth Allred, William Dupont, and Frank Harrell supplied various kinds of encouragement.

8 References


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 was an author of several inserts in the Stata Technical Bulletin and is an editor of the Stata Journal. His previous Speaking Stata articles on graphics have been collected as Speaking Stata Graphics (College Station, TX: Stata Press, 2014).
Stata tip 121: Box plots side by side

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

Box plots are a standard plot type in statistical graphics and, as such, are popular with Stata users. The official Stata commands `graph box` and `graph hbox` are identical except that `graph box` draws box plots with the response (or outcome) scale on the vertical axis and `graph hbox` draws plots with the response scale on the horizontal axis. Contrary to the usual mathematical convention, the response axis is always regarded as the y axis for these commands so that options such as `ytitle()`, `ylabel()`, and `yscale()` always apply to the axis with the response variable. The manual entry [G-2] `graph box` gives much more detail and pertinent references. For a wider discussion of box plots, including how to draw box plots and related plots with `graph twoway`, see Cox (2009, 2013).

The greatest value of box plots is for comparing distributions of related variables or distributions of single variables for different groups of observations. This tip focuses on how and which data are plotted side by side. I explain the default appearance and structure of side-by-side box plots and how to tune or even to reverse that default.

To make this question concrete, we read in some data and then plot some graphs. As often happens, the code here is a cleaned-up version of what was done in preparing the tip, with afterthoughts and second guesses turned into anticipations of useful ideas.

```
.set scheme sj
.sysuse citytemp
.local title "Mean temperatures (°F)"
.label var tempjan "January"
.label var tempjul "July"
```

The `citytemp` dataset distributed with Stata contains temperature data for various U.S. cities. These are given in degrees Fahrenheit, a scale on which water freezes at 32°F and water boils at 212°F. Most countries of the world use the Celsius (formerly centigrade) scale °C, for which water freezes at 0°C and boils at 100°C. The degree symbol can be shown as a text symbol, as explained in the help for `text`. If this functionality is not available in your Stata, you can use the trick explained in Cox (2004). One way or another, we create a local macro indicating units of measurement for use in later graphs. Because we plan to use a graph title explaining that we are showing temperatures, the month names suffice as variable labels.
Some examples of box plots are shown in figures 1 and 2. Figures 1a and 2a show vertical box plots for January and July temperatures in various regions of the United States, while figures 1b and 2b show corresponding horizontal box plots. The commands are as follows:

```stata
. graph box tempjan tempjuly, over(region) ytitle('title')
    ylabel(14 32 50 68 86, angle(h))
. graph hbox tempjan tempjuly, over(region) ytitle('title') ylabel(14(18)86)
. graph box tempjan tempjuly, by(region, rows(1) compact note('**'))
    ytitle('title') ylabel(14(18)86, angle(h))
. graph hbox tempjan tempjuly, by(region, cols(1) compact note('**'))
    ytitle('title') ylabel(14(18)86)
```

![Box plots for January and July temperatures in various regions](image)

Figure 1. Box plots for January and July temperatures of various U.S. cities using the `over()` option to compare different regions.
The axis labels \(14(18)86\) may seem a strange choice to U.S. readers, but \(32^\circ\) F is a key threshold, while differences of \(18^\circ\) F between labeled ticks match differences of \(10^\circ\) C. Figure 1 uses the \texttt{over()} option to compare different regions, while figure 2 uses the \texttt{by()} option to compare regions. In broad terms, the \texttt{by()} option is more flexible but produces more scaffolding. The scaffolding is sometimes helpful in indicating the subdivisions of the graph clearly but sometimes less helpful in that it may take up valuable space. Users aware of both syntaxes can make an informed choice.

What is less well known is that the \texttt{by()} option can be tuned so that results resemble those of the \texttt{over()} option. This trick may be applied more widely than just to box plots. Appropriate incantations tweak the position and appearance of the subtitles of the component graphs. It is convenient, but not essential, to define those incantations with local macros for repeated use in later commands. Note the clock notation for position, which places subtitles for vertical box plots at 12 o’clock and those for horizontal plots at 9 o’clock. Figure 3 shows the results.
Despite these minor variations, the design common to all the plots so far is that different variables are placed closest (on the inside, as it were) and groups of observations, as defined by the distinct values of the variable specified in `over()` or `by()`, are placed more broadly. What is to be done if the opposite order is wanted? Suppose that the contrast between January and July (Northern Hemisphere winter and summer) is thought less interesting than the contrasts between different regions. We then need regions, not months, to be next to each other.

For the opposite order, we need a different data structure, which can be obtained through the `reshape` command. If `reshape` is new to you, refer to the online help and manual entry. In this example, `reshape` stacks different variables into one variable that is subdivided by a group variable indicating where the groups came from. This is an easy change of data structure to envisage and one that is often needed.

Figure 3. Box plots for January and July temperatures of various U.S. cities using the `by()` option to compare different regions, but with panel titles shown differently.
The `citytemp` data lack an identifier variable naming the observations, here cities. We do need an identifier for `reshape`, but the observation number will work well.

```
. generate id = _n
```

In a very large dataset, we would make such an identifier of `long` storage type. Some judicious renaming of variables can also be a good idea:

```
. rename (tempjan tempjul) (tempJanuary tempJuly)
. reshape long temp, i(id) j(month) string
```

Now the combined variable `temp` can be grouped by `region`, as before, and also by `month`, a new variable created by `reshape`. We can choose which variable goes on the inside. In this example, we already suspect that comparing temperatures by region may be more interesting than comparing by month. With many other datasets (for example, medical results compared by sex and age group), you may need to experiment to see what works best. Comparisons between subtle effects of interest and starker but well-known effects often recur. Figure 4 shows the results of this example.

```
. graph box temp, over(region) by(month, rows(1) compact note(""))
   > ytitle(\'title\') ylabel(14(18)86, angle(h) grid) \'incant1\'
. graph hbox temp, over(region) by(month, cols(1) compact note(""))
   > ytitle(\'title\') ylabel(14(18)86, grid) \'incant2\'
```

![Figure 4. Box plots for January and July temperatures of various U.S. cities using both over() and by() options to compare different regions and months](image)

Figure 4. Box plots for January and July temperatures of various U.S. cities using both `over()` and `by()` options to compare different regions and months.
References


Software Updates


A bug in *acd* concerning the handling of missing values in the expression (*exp*) has been corrected. The help file is unchanged.